Advanced Cryptography — Final Exam
Solution

Serge Vaudenay

21.6.2016

- 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 Σ Protocol in an Group of Exponent 2

Given an integer s, we consider an Abelian group G (with multiplicative notations) such that for all $x \in G$, we have $x^2 = 1$. We assume that there are deterministic polynomial time algorithms to compute the order n of G, to multiply and to compare two group elements. More precisely, given x and y we can compute xy and say whether $x = y$. For $x = (x_1, \ldots, x_m, y) \in G^{m+1}$ and $w = (w_1, \ldots, w_m) \in \{0,1\}^m$, we consider the following relation:

$$R(x; w) \Longleftrightarrow y = x_1^{w_1} \cdots x_m^{w_m}$$

We consider the following protocol:

$$\begin{array}{rcl}
\textbf{Prover} & & \textbf{Verifier} \\
w & x & \\
 & & \\
\text{pick } r = (r_1, \ldots, r_m) \in_U \{0,1\}^m & & \\
a = x_1^{r_1} \cdots x_m^{r_m} & \xrightarrow{a} & \\
 & \xleftarrow{e} & \text{pick } e \in_U \{0,1\} \\
z = r \oplus ew & \xrightarrow{z} & \text{check } x_1^{z_1} \cdots x_m^{z_m} = ay^e
\end{array}$$

where $\oplus$ denotes the XOR operation (exclusive OR) between two bits and $\in_U$ denotes a random selection with uniform distribution and fresh coins.

Q.1 Following the terminology of Σ protocols, show that the above protocol has *special soundness.*

Given x, a, and the answers z_0 and z_1 to $e = 0$ and $e = 1$, we compute $w = z_0 \oplus z_1$. We obtain $y = x_1^{w_1} \cdots x_m^{w_m}$. So, we have an extractor for special soundness.

Q.2 Following the terminology of Σ protocols, show that the above protocol is *special HVZK* (special honest verifier zero-knowledge).

We first observe that in the proposed protocol, z is uniformly distributed when $e = 0$ (this is just $r = (r_1, \ldots, r_m)$). When $e = 1$, this is the XOR between r and w. Since r is uniform and independent from w, z is uniformly distributed in this case as well.
Next, we observe that we can always compute a from z, e, and y.
Given x and e, we pick $z \in \{0,1\}^m$ uniformly at random and deduce a from the value y inside x. Clearly, the obtained (a, e, z) has a correct distribution with e imposed. So, we have a perfect simulator for special HVZK.

Q.3 Show that the proposed protocol is a Σ protocol.

The protocol follows the structure of Σ protocols:
- *the verifier is polynomially bounded;*
- *it has 3 moves initiated by the prover;*
- *the challenge message from the verifier is selected at random from a set independently from the first message from the prover;*
- *the acceptance condition only depends on (x, a, e, z).*

Furthermore, we have prover special soundness and special HVZK. So, we have a Σ protocol.

This exercise was quite simple. Nevertheless, we spotted two frequent mistakes.

- *The extractor and the simulator for Σ protocols work with transcripts instead of views.*
- *For the extractor, we cannot assume that the answers z_0 and z_1 to $e = 0$ and $e = 1$ are $z_0 = r$ and $z_1 = r \oplus w$ and simply answer that $z_0 \oplus z_1$ is the witness. Instead, we should show that $z_0 \oplus z_1$ is a valid witness from the properties of z_0 and z_1: given that $x^{z_e} = ay^e$, then $x^{z_1 - z_0} = y$.*

Also: for the simulator, it is important to show that the output has the same distribution as a natural transcript.

2 On Generator Generation in Diffie-Hellman Problems

In the Computational Diffie-Hellman (CDH) Problem and the Decisional Diffie-Hellman (DDH) Problem, there is a security parameter (integer) s as input to a probabilistic polynomial-time (PPT) algorithm $\mathsf{Gen}(1^s) \to (q, \mathsf{params}, g)$ to generate a prime number q together with an element g and some parameters params. The values q and params define a group $G = (q, \mathsf{params})$ of order q in which g is a generator. We denote $G = \langle g \rangle$ and $x \in G$ to mean that g generates G and x belongs to G. We assume multiplicative notations in the group. We assume we have two deterministic polynomial-time algorithms MUL and EQ such that for all $x, y \in G$, we have $\mathsf{MUL}(G, x, y) = xy$ and $\mathsf{EQ}(G, x, y) = 1_{x=y}$.

Q.1 Show that we can design deterministic polynomial-time algorithms UN, INV, and POW such that for all $x \in G$ and $e \in \mathbf{Z}$, we have $\mathsf{UN}(G, x) = 1$, $\mathsf{INV}(G, x) = x^{-1}$, and $\mathsf{POW}(G, x, e) = x^e$.
CAUTION: be careful with the $e = 0$ and $e < 0$ cases.

> *We define $\mathsf{POW}(G, x, e)$ for e positive by using the square-and-multiply algorithm using MUL.*
> *Then, we can define $\mathsf{UN}(G, x) = \mathsf{POW}(G, x, q)$ as $x^q = 1$. Note that it is necessary to have one element group element x to compute 1. This is the case when we have, for instance, a generator.*
> *We can also define $\mathsf{INV}(G, x) = \mathsf{POW}(G, x, q-1)$ as $x^{q-1} \times x = 1$.*
> *Finally, we can define $\mathsf{POW}(G, x, 0) = \mathsf{UN}(G, x)$ and $\mathsf{POW}(G, x, e) = \mathsf{POW}(G, x, e \bmod q)$ for any $e \in \mathbf{Z}$.*
> *COMMENT after correction: we cannot just say that $\mathsf{UN}(G, x)$ just answer 1 as this "1" is not necessarily the integer 1 that we know. Here, we want to output the neutral element from the group and we need to reconstruct it from the given parameters.*
> *In the correction, we have seen several times some non-polynomial solutions. For instance, compute $\mathsf{POW}(G, x, e)$ using $e - 1$ multiplications or finding $\mathsf{INV}(G, x)$ by exhaustive search. These answers are not acceptable.*

In this exercise, we look at the influence on the g generation by Gen in the Gen-CDH and Gen-DDH problems. We assume a first PPT algorithm $\mathsf{Setup}(1^s) \to (q, \mathsf{parms})$ to generate the group $G = (q, \mathsf{parms})$ and we assume that from G we can extract a generator $g = \mathsf{Generator}(G)$ using a deterministic polynomial-time algorithm $\mathsf{Generator}$. We define two generating algorithms.

$\mathsf{GenFixed}(1^s; \rho)$:
1: run $\mathsf{Setup}(1^s; \rho) \to G = (q, \mathsf{parms})$
2: run $g = \mathsf{Generator}(G)$
3: **output** (q, parms, g)

We call $\mathsf{GenFixed}$ *the setup with fixed generator g.*

$\mathsf{GenRand}(1^s; \rho)$:

1: split ρ into two independent sequences ρ_1 and ρ_2
2: run $\mathsf{Setup}(1^s; \rho_1) \to G = (q, \mathsf{parms})$
3: run $g = \mathsf{Generator}(G)$
4: generate $a \in \mathbf{Z}_q^*$ with uniform distribution from ρ_2
5: set $h = \mathsf{POW}(G, g, a)$
6: **output** (q, parms, h)

We call GenRand *the setup with random generator* h.

The DDH problem specifies two distributions with parameter s:

Source $S_0(\mathsf{Gen})$:

1: pick a large enough sequence of independent fair coin flips ρ
2: run $\mathsf{Gen}(1^s; \rho) \to (q, \mathsf{parms}, g)$
3: pick $x, y \in \mathbf{Z}_q$ with uniform distribution (x, y, and ρ are independent)
4: set $X = g^x$, $Y = g^y$, $Z = g^{xy}$
5: **output** $(q, \mathsf{parms}, g, X, Y, Z)$

Source $S_1(\mathsf{Gen})$:

1: pick a large enough sequence of independent fair coin flips ρ
2: run $\mathsf{Gen}(1^s; \rho) \to (q, \mathsf{parms}, g)$
3: pick $x, y, z \in \mathbf{Z}_q$ with uniform distribution (x, y, z, and ρ are independent)
4: set $X = g^x$, $Y = g^y$, $Z = g^z$
5: **output** $(q, \mathsf{parms}, g, X, Y, Z)$

The Gen-DDH problem consists of distinguishing $S_0(\mathsf{Gen})$ and $S_1(\mathsf{Gen})$. We stress that the DDH problem is relative to Gen; this is why we call it *the* Gen-*DDH problem*. We define the advantage

$$\mathsf{Adv}^{\mathsf{Gen\text{-}DDH}}(\mathcal{A}) = \Pr[\mathcal{A}(S_0(\mathsf{Gen})) = 1] - \Pr[\mathcal{A}(S_1(\mathsf{Gen})) = 1]$$

The Gen-DDH is hard if and only if for all PPT distinguisher $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{Gen\text{-}DDH}}(\mathcal{A})$ is negligible.

Q.2 Given a GenRand-DDH distinguisher $\mathcal{A}$, construct a GenFixed-DDH distinguisher $\mathcal{B}$ with the same advantage and a similar complexity.

We define $\mathcal{B}$ as follows:

$\mathcal{B}(q, \mathsf{params}, g, X, Y, Z)$*:*

1: *set* $G = (q, \mathsf{params})$
2: *pick* $a \in \mathbf{Z}_q^*$ *with uniform distribution*
3: *set* $h = \mathsf{POW}(G, g, a)$, $X' = \mathsf{POW}(G, X, a)$, $Y' = \mathsf{POW}(G, Y, a)$, $Z' = \mathsf{POW}(G, Z, a)$
4: *run* $b = \mathcal{A}(G, h, X', Y', Z')$
5: ***output*** b

Clearly, with the sources $S_0(\mathsf{GenFixed})$ *and* $S_1(\mathsf{GenFixed})$*, the input* (q, params, g) *to* $\mathcal{B}$ *follows the distribution of* GenFixed*. So,* (q, params, h) *is identical to the distribution induced by* GenRand*. Hence, the input* $(q, \mathsf{params}, h, X, Y, Z)$ *of* $\mathcal{A}$ *in Step 4 follows either source* $S_0(\mathsf{GenRand})$ *or* $S_1(\mathsf{GenRand})$*. Therefore, the advantage of* $\mathcal{B}$ *equals the advantage of* $\mathcal{A}$*.*

The complexity overhead in $\mathcal{B}$ *corresponds to Step 2 and Step 3 which are polynomial.*

COMMENT after correction: some students just answered that $\mathcal{A}(G, g, X, Y, Z)$ *will work as it is a special case. We only know that* $\mathcal{A}$ *works for random inputs so it is not guaranteed that it works for the special input* g*. This is the main problem in this exercise which requires to randomize inputs. We did not accept these answers. The same problem occurred in the next questions.*

Q.3 Show that if the GenFixed-DDH is hard, then the GenRand-DDH problem is hard.

Any PPT GenRand*-DDH distinguisher has the same advantage of some PPT* GenFixed*-DDH distinguisher. If the* GenFixed*-DDH is hard, it must be negligible. So, the* GenRand*-DDH problem is hard.*

Unfortunately, we have no implication in the other direction for the DDH problem, but there is for the CDH problem.

The computational Diffie-Hellman (CDH) problem has instances defined by the following source:

Source $S(\mathsf{Gen})$:

1: pick a large enough sequence of independent fair coin flips ρ
2: run $\mathsf{Gen}(1^s; \rho) \rightarrow (q, \mathsf{params}, g)$
3: pick $x, y \in \mathbf{Z}_q^*$ with uniform distribution (x, y, and ρ are independent)
4: set $X = g^x$, $Y = g^y$ {the solution to the problem is g^{xy}}
5: **output** $(q, \mathsf{params}, g, X, Y)$

Given a CDH solver $\mathcal{A}$, we define

$$\mathsf{Succ}^{\mathsf{Gen\text{-}CDH}}(\mathcal{A}) = \Pr[\mathcal{A}(S(\mathsf{Gen})) = g^{xy}]$$

The Gen-CDH is hard if and only if for all PPT solver $\mathcal{A}$, $\mathsf{Succ}^{\mathsf{Gen\text{-}CDH}}(\mathcal{A})$ is negligible.

Q.4 Given a GenRand-CDH solver $\mathcal{A}$, construct a GenFixed-CDH solver $\mathcal{B}$ with similar complexity and

$$\mathsf{Succ}^{\mathsf{GenRand\text{-}CDH}}(\mathcal{A}) = \mathsf{Succ}^{\mathsf{GenFixed\text{-}CDH}}(\mathcal{B})$$

We define $\mathcal{B}$ as follows:

$\mathcal{B}(q, \mathsf{params}, g, X, Y)$*:*

set $G = (q, \mathsf{params})$
pick $a \in \mathbf{Z}_q^*$ *with uniform distribution*
set $g' = \mathsf{POW}(G, g, a)$
set $X' = \mathsf{POW}(G, X, a)$
set $Y' = \mathsf{POW}(G, Y, a)$
run $Z' = \mathcal{A}(G, g', X', Y')$
set $Z = \mathsf{POW}(G, Z', 1/a \bmod q)$
output Z

Just like for DDH, we show that $(q, \mathsf{params}, g', X', Y')$ *follows the distribution of* $S(\mathsf{GenRand})$ *with solution* Z^a*, where* Z *is the solution to the* GenFixed-*CDH problem. So, the probability of success is preserved. Similarly, the complexity overhead is small.*

COMMENT after correction: we have seen several pearls for which we gave a penalty: $g^{xy} = g^{xya^2} g^{a^{-2}}$, $(g^{axy})^{-a} = g^{xy}$, *or* $(g^x)^u = g^{u^x}$ *!!!*

Q.5 Given a GenFixed-CDH solver $\mathcal{A}$, we denote

$$\varepsilon_\rho = \Pr[\mathcal{A}(S(\mathsf{GenFixed})) = g^{xy} | \rho]$$

the probability of success when ρ in S is fixed. So, GenFixed always returns the same group and generator (due to ρ being fixed). Only x, y, and possible coins used by $\mathcal{A}$ remain random.

Given a GenFixed-CDH solver $\mathcal{A}$, show that we can construct an algorithm Mu such that <u>for any</u> integer x and y (i.e., not only for random x and y) and any ρ, we have

$$\mathsf{Mu}(q, \mathsf{params}, g, g^x, g^y) = g^{xy}$$

for $\mathsf{GenFixed}(1^s; \rho) \to (q, \mathsf{params}, g)$ with probability at least ε_ρ over the distribution of x, y, and possible coins by $\mathcal{A}$.

We just have to randomize g^x and g^y. There is a particular case when one of the two is equal to 1. We can check it using EQ. *In this case, the algorithm should answer* 1 *and it is correct with probability 1.*
In other cases, we pick a and b in $\mathbf{Z}_q^*$ uniformly at random and run $\mathcal{A}(q, \mathsf{params}, g, g^{ax}, g^{by}) = Z$ then output $Z^{\frac{1}{ab}}$. Now, g^{ax} and g^{by} are independent and uniformly distributed in $G - \{1\}$. So they are correctly distributed and $\mathcal{A}(q, \mathsf{params}, g, g^{ax}, g^{by}) = g^{abxy}$ with probability $\mathsf{Succ}^{\mathsf{GenFixed\text{-}CDH}}(\mathcal{A})$. So, $Z^{\frac{1}{ab}} = g^{xy}$ with this probability.

$\mathsf{Mu}(q, \mathsf{params}, g, X, Y)$: $\{$*say* $X = g^x$ *and* $Y = g^y\}$

```
set G = (q, params) and u = UN(G, g)
if EQ(G, X, u) = 1 or EQ(G, Y, u) = 1 then
    output u
end if
pick a, b ∈ Z_q^* with uniform distribution
compute X^a = POW(G, X, a) and Y^b = POW(G, Y, b)
run Z = A(q, params, g, X^a, Y^b) {we should have Z = g^{abxy}}
output Z^{1/(ab)}
```

Q.6 Show that we can construct an algorithm In such that *for any* integer x and any ρ, we have

$$\mathsf{In}(q, \mathsf{params}, g, g^x) = g^{\frac{1}{x}}$$

for $\mathsf{GenFixed}(1^s; \rho) \to (q, \mathsf{params}, g)$ with probability at least ε_ρ^w for $w = \mathcal{O}(\log q)$.

We have $g^{\frac{1}{x}} = g^{x^{q-2}}$.
Using Mu *from the previous question we compute $g^{x^{q-2}}$ with a square-and-multiply algorithm. If w denotes the number of squares or multiplications to perform, we have $w = \mathcal{O}(\log q)$.*
Once ρ is fixed, all w operations are independent. Since they each succeed with probability ε_ρ, the overall process succeeds with probability at least ε_ρ^w.
COMMENT after correction: several times we say incorrect answers such as $\mathsf{In}(g, g^x) = \mathsf{Mu}(g^x, g^x, g)$ of other answers where the generator given as input to g was not the fixed one g. However, we constructed g to work for this very particular fixed g and our main problem is to construct one which works on average for a random g.

Q.7 Given a GenFixed-CDH solver $\mathcal{A}$, construct a GenRand-CDH solver $\mathcal{B}$ with similar complexity and

$$\mathsf{Succ}^{\mathsf{GenRand\text{-}CDH}}(\mathcal{B}) \geq \left(\mathsf{Succ}^{\mathsf{GenFixed\text{-}CDH}}(\mathcal{A})\right)^{\mathcal{O}(\log q)}$$

Let Mu *and* In *be the algorithms from the previous questions.*

$\mathcal{B}(q, \mathsf{params}, h, X, Y)$*:* $\{$*say* $h = g^a$*,* $X = g^{ax}$*,* $Y = g^{ay}\}$

1: *set* $G = (q, \mathsf{params})$
2: *run* $g = \mathsf{Generator}(G)$
3: *run* $h' = \mathsf{In}(G, g, h)$ $\{$*we should have* $h' = g^{\frac{1}{a}}\}$
4: *run* $A = \mathsf{Mu}(G, g, X, Y)$ $\{$*we should have* $A = g^{a^2xy}\}$
5: *run* $B = \mathsf{Mu}(G, g, h', A)$ $\{$*we should have* $B = g^{axy}\}$
6: *run* $Z = \mathsf{Mu}(G, g, h', B)$ $\{$*we should have* $Z = g^{xy}\}$
7: ***output*** Z

Clearly, this works with probability at least $E_\rho(\varepsilon_\rho^{w+3})$*. By using the Jensen inequality, we obtain* $E_\rho(\varepsilon_\rho^{w+3}) \geq E_\rho(\varepsilon_\rho)^{w+3} = \left(\mathsf{Succ}^{\mathsf{GenFixed\text{-}CDH}}(\mathcal{A})\right)^{w+3}$*. Then, we use* $w = \mathcal{O}(\log q)$*.*

We could also have computed $h' = g^{\frac{1}{a^2}}$ *directly (by* $g^{a^{q-2}}$*) and obtained* $B = g^{xy}$ *to save one* Mu *operation.*

COMMENT after correction: in the copies given during the exam, the question was given with

$$\mathsf{Succ}^{\mathsf{GenFixed\text{-}CDH}}(\mathcal{B}) \geq \left(\mathsf{Succ}^{\mathsf{GenRand\text{-}CDH}}(\mathcal{A})\right)^{\mathcal{O}(\log q)}$$

The mistake was corrected during the exam and written on the black board.

Q.8 Show that GenFixed-CDH is hard if and only if GenRand-CDH is hard.

If GenFixed-*CDH is hard, any* GenRand-*CDH solver has a probability of success equal to the one of a* GenFixed-*CDH solver (due to Q.4). So, it must be negligible. Hence,* GenRand-*CDH is hard.*

If GenRand-*CDH is hard, any* GenFixed-*CDH solver has a probability of success bounded by the one of a* GenRand-*CDH solver raised to the power* $\frac{1}{\mathcal{O}(\log q)}$ *(due to Q.7). Since* GenFixed *returns* q*,* $\log q$ *must be polynomially bounded. So, this power must be negligible. Hence,* GenFixed-*CDH is hard.*

COMMENT after correction: We gave $2/3$ *of the grade if one direction was missing. We gave a penalty when nothing was said about the negligible advantage. We gave no points to answers such as "if we can easily solve* GenRand *then we can easily solve* GenFixed *as it is a particular case".*

3 Equivalent PRF Notions

We consider a function family f_s which depends on a security parameter s. Given s, the function f_s takes a key $k \in \mathcal{K}_s$ and an input $x \in \mathcal{X}_s$. It produces an output $y = f_s(k, x) \in \mathcal{Y}_s$. To have lighter notations, from now on the subscript s is omitted. We further write the input k of f as a subscript to write $f_k(x) = f(k, x)$. We say that the function family f is a pseudorandom function (PRF) if it can be computed in polynomial time (in terms of s) and if for every probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, the function $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRF}}$ (this is a function in terms of s) is a negligible function where

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRF}} = \Pr[\Gamma_0^{\mathsf{PRF}}(\mathcal{A}) = 1] - \Pr[\Gamma_1^{\mathsf{PRF}}(\mathcal{A}) = 1]$$

and $\Gamma_b^{\mathsf{PRF}}(\mathcal{A})$ is defined with a bit b as follows:

Game $\Gamma_b^{\mathsf{PRF}}(\mathcal{A})$:

pick $s \in \mathcal{K}$ at random
set ρ to a long enough sequence of random coins
set $i = 1$
$(q, x_i) \leftarrow \mathcal{A}(;\rho)$
while $q \neq \mathsf{final}$ **do**
    if $x_i \in \{x_1, \ldots, x_{i-1}\}$ **then**
        abort {it is not allowed to repeat a query}
    end if
    if $b = 0$ **then**
        set $y_i = f_s(x_i)$
    else
        set $y_i \in \mathcal{Y}$ at random
    end if
    $i \leftarrow i + 1$
    $(q, x_i) \leftarrow \mathcal{A}(y_1, \ldots, y_{i-1}; \rho)$
end while
output x_i

Here, $\mathcal{A}$ returns a pair (q, x). The string q is either query or final. If $q = \mathsf{query}$, it means that $\mathcal{A}$ wants to query $f_s(x)$ and continue. If $q = \mathsf{final}$, it means that $\mathcal{A}$ is done and returning a bit x as a final output.

We recall that a function $\mathsf{Adv}(s)$ is negligible is for all $c > 0$, we have $\mathsf{Adv}(s) = \mathcal{O}(s^{-c})$ when $s \to +\infty$.

In this exercise, we consider another notion defined by the following game:

Game $\Gamma_b^{\mathsf{prePRF}}(\mathcal{A})$:

pick $s \in \mathcal{K}$ at random
set ρ to a long enough sequence of random coins
set $i = 1$ and unset flag
$(q, x_i) \leftarrow \mathcal{A}(;\rho)$
while $q \neq \mathsf{final}$ **do**

if $x_i \in \{x_1, \ldots, x_{i-1}\}$ **then**
  abort {it is not allowed to repeat a query}
end if
if $q = \mathsf{challenge}$ and flag is set **then**
  abort {it is not allowed to make two challenges}
end if
if $q = \mathsf{challenge}$ **then**
  set flag {$\mathcal{A}$ is making a challenge}
end if
if $q = \mathsf{challenge}$ and $b = 1$ **then**
  set $y_i \in \mathcal{Y}$ at random
else
  set $y_i = f_s(x_i)$
end if
$i \leftarrow i + 1$
$(q, x_i) \leftarrow \mathcal{A}(y_1, \ldots, y_{i-1}; \rho)$
end while{we must have $q = \mathsf{final}$}
output x_i

Essentially, $\mathcal{A}$ always plays with f with $q = \mathsf{query}$ and at some point uses only once a special $q = \mathsf{challenge}$. For this "challenge", the response which is returned to him is $f_s(x)$ if $b = 0$ or something random if $b = 1$. An equivalent way consists of saying that $\mathcal{A}^{f_k(\cdot)}$ works in two phases, playing with a $f_k(\cdots)$ oracle. In between the two phases, it makes a challenge which is answer by $f_k(\cdots)$ or at random.

We define

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{prePRF}} = \Pr[\Gamma_0^{\mathsf{prePRF}}(\mathcal{A}) = 1] - \Pr[\Gamma_1^{\mathsf{prePRF}}(\mathcal{A}) = 1]$$

and we say that the function family f is a prePRF if it can be computed in polynomial time (in terms of s) and if for every PPT algorithm $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{prePRF}}$ is negligible.

The objective of this exercise is to show that PRF and prePRF are equivalent security notions.

Q.1 Given a prePRF adversary $\mathcal{A}$ and a bit b, we construct a PRF adversary $\mathcal{B}_b$ as follows:

$\mathcal{B}_b(y_1, \ldots, y_{i-1}; \rho)$:
if $i = 1$ **then**
  set seq_x and seq_y to the empty sequence {first execution of $\mathcal{B}_b$}
else
  set $\mathsf{seq}_y \leftarrow (\mathsf{seq}_y, y_{i-1})$ {y_{i-1} is the answer to the previous query}
end if
run $(q, x) = \mathcal{A}(\mathsf{seq}_y; \rho)$
if $x \in \mathsf{seq}_x$ **then**
  abort {it is not allowed to repeat a query}
end if
set $\mathsf{seq}_x \leftarrow (\mathsf{seq}_x, x)$ {insert x in the list of queries}

if $q = \mathsf{challenge}$ and $b = 1$ **then**
  set $y \in \mathcal{Y}$ at random
  set $\mathsf{seq}_y \leftarrow (\mathsf{seq}_y, y)$
  run $(q, x) = \mathcal{A}(\mathsf{seq}_y; \rho)$
  if $x \in \mathsf{seq}_x$ **then**
    abort {it is not allowed to repeat a query}
  end if
  set $\mathsf{seq}_x \leftarrow (\mathsf{seq}_x, x)$ {insert x in the list of queries}
end if
output (q, x)

So, $\mathcal{B}$ simulates $\mathcal{A}$ and simulates the answer to random for the $q = \mathsf{challenge}$ and $b = 1$ case.
Show that

$$\begin{aligned}
\Pr[\Gamma_0^{\mathsf{prePRF}}(\mathcal{A}) = 1] &= \Pr[\Gamma_0^{\mathsf{PRF}}(\mathcal{B}_0) = 1] \\
\Pr[\Gamma_1^{\mathsf{prePRF}}(\mathcal{A}) = 1] &= \Pr[\Gamma_0^{\mathsf{PRF}}(\mathcal{B}_1) = 1] \\
\Pr[\Gamma_1^{\mathsf{PRF}}(\mathcal{B}_0) = 1] &= \Pr[\Gamma_1^{\mathsf{PRF}}(\mathcal{B}_1) = 1]
\end{aligned}$$

Essentially, $\mathcal{B}_b$ simulates $\mathcal{A}$ who plays the prePRF game but skips a response to a challenge set to random. Clearly, the game $\Gamma_0^{\mathsf{prePRF}}(\mathcal{A})$ is fully simulated by $\Gamma_0^{\mathsf{PRF}}(\mathcal{B}_0)$. Similarly, $\Gamma_1^{\mathsf{prePRF}}(\mathcal{A})$ is fully simulated by $\Gamma_0^{\mathsf{PRF}}(\mathcal{B}_1)$. If we consider Γ_1^{PRF}, all answers to queries are random so there is no difference between $\mathcal{B}_0$ and $\mathcal{B}_1$. Hence, $\Gamma_1^{\mathsf{PRF}}(\mathcal{B}_0)$ and $\Gamma_1^{\mathsf{PRF}}(\mathcal{B}_1)$ are identical.

Q.2 Show that if f is a PRF, then f is a prePRF.

Let $\mathcal{A}$ be a prePRF adversary. We construct and adversary $\mathcal{B}_b$ as in the previous question. We have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{\mathsf{prePRF}} &= \Pr[\Gamma_0^{\mathsf{prePRF}}(\mathcal{A}) = 1] - \Pr[\Gamma_1^{\mathsf{prePRF}}(\mathcal{A}) = 1] \\
&= \Pr[\Gamma_0^{\mathsf{PRF}}(\mathcal{B}_0) = 1] - \Pr[\Gamma_0^{\mathsf{PRF}}(\mathcal{B}_1) = 1] \\
&= \Pr[\Gamma_0^{\mathsf{PRF}}(\mathcal{B}_0) = 1] - \Pr[\Gamma_1^{\mathsf{PRF}}(\mathcal{B}_0) = 1] + \\
&\quad \Pr[\Gamma_1^{\mathsf{PRF}}(\mathcal{B}_1) = 1] - \Pr[\Gamma_0^{\mathsf{PRF}}(\mathcal{B}_1) = 1] \\
&= \mathsf{Adv}_{\mathcal{B}_0}^{\mathsf{PRF}} - \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{PRF}}
\end{aligned}$$

Since f is a PRF, both $\mathsf{Adv}_{\mathcal{B}_0}^{\mathsf{PRF}}$ and $\mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{PRF}}$ are negligible. So, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{prePRF}}$ is negligible. As this holds for any PPT adversary $\mathcal{A}$, f is a prePRF.

Q.3 We define the following game:

Game $\Gamma^j(\mathcal{A})$:
pick $s \in \mathcal{K}$ at random

set ρ to a long enough sequence of random coins
set $i = 1$
$(q, x_i) \leftarrow \mathcal{A}(;\rho)$
while $q \neq \mathsf{final}$ **do**
  if $x_i \in \{x_1, \ldots, x_{i-1}\}$ **then**
    abort {it is not allowed to repeat a query}
  end if
  if $i \leq j$ **then**
    set $y_i = f_s(x_i)$ {answer using f_s to the j first queries}
  else
    set $y_i \in \mathcal{Y}$ at random
  end if
  $i \leftarrow i + 1$
  $(q, x_i) \leftarrow \mathcal{A}(y_1, \ldots, y_{i-1}; \rho)$
end while{we must have $q = \mathsf{final}$}
output x_i

Show that for a PPT adversary $\mathcal{A}$, there exists some polynomially bounded Q such that we have

$$\Pr[\Gamma^Q(\mathcal{A}) = 1] = \Pr[\Gamma_0^{\mathsf{PRF}}(\mathcal{A}) = 1]$$
$$\Pr[\Gamma^0(\mathcal{A}) = 1] = \Pr[\Gamma_1^{\mathsf{PRF}}(\mathcal{A}) = 1]$$

For $j = 0$, the Γ^0 game always answers to queries at random, so just like the Γ_1^{PRF} game. If we set Q to at least the total number of queries by $\mathcal{A}$, which must be polynomially bounded, the Γ^Q game always answers to queries by setting $y_i \in \mathcal{Y}$ at random, so just like the Γ_0^{PRF} game.

Q.4 Given a PPT adversary $\mathcal{A}$ and an integer j, we construct an adversary $\mathcal{B}_j$ as follows:

$\mathcal{B}_j(y_1, \ldots, y_{i-1}; \rho)$:
run $(q, x_i) = \mathcal{A}(y_1, \ldots, y_{i-1}; \rho)$
if $i = j$ **then**
  set q to $\mathsf{challenge}$
end if
if $i > j$ **then**
  while $q \neq \mathsf{final}$ and $x_i \notin \{x_1, \ldots, x_{i-1}\}$ **do**
    set $y_i \in \mathcal{Y}$ at random
    $i \leftarrow i + 1$
    run $(q, x_i) = \mathcal{A}(y_1, \ldots, y_{i-1}; \rho)$
  end while
end if

output (q, x_i)

Show that

$$\Pr[\Gamma^j(\mathcal{A}) = 1] = \Pr[\Gamma_0^{\mathsf{prePRF}}(\mathcal{B}_j) = 1]$$
$$\Pr[\Gamma^{j-1}(\mathcal{A}) = 1] = \Pr[\Gamma_1^{\mathsf{prePRF}}(\mathcal{B}_j) = 1]$$

This means that $\mathcal{B}_j$ will make exactly $j-1$ queries which fully simulate the $\mathcal{A}$ queries. The jth one will be given as a challenge in the prePRF *game. Then, $\mathcal{B}$ will simulate $\mathcal{A}$ who is always answered at random (unless a query repeats in which case it will repeat as well and let the game abort).*
In $\Gamma_0^{\mathsf{prePRF}}$, exactly j queries use f_s and others use random output, just like in Γ^j. In $\Gamma_1^{\mathsf{prePRF}}$, exactly $j-1$ queries use f_s and others use random output, just like in Γ^{j-1}.

Q.5 Show that if f is a prePRF, then f is a PRF.

Let $\mathcal{A}$ be a PPT adversary playing the PRF game. We construct the $\mathcal{B}_j$ adversaries. Due to the two previous questions, for Q large enough, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRF}} &= \Pr[\Gamma_0^{\mathsf{PRF}}(\mathcal{A}) = 1] - \Pr[\Gamma_1^{\mathsf{PRF}}(\mathcal{A}) = 1] \\
&= \Pr[\Gamma^Q(\mathcal{A}) = 1] - \Pr[\Gamma^0(\mathcal{A}) = 1] \\
&= \sum_{j=1}^{Q} \Pr[\Gamma^j(\mathcal{A}) = 1] - \Pr[\Gamma^{j-1}(\mathcal{A}) = 1] \\
&= \sum_{j=1}^{Q} \Pr[\Gamma_0^{\mathsf{prePRF}}(\mathcal{B}_j) = 1] - \Pr[\Gamma_1^{\mathsf{prePRF}}(\mathcal{B}_j) = 1] \\
&= \sum_{j=1}^{Q} \mathsf{Adv}_{\mathcal{B}_j}^{\mathsf{prePRF}}
\end{aligned}$$

If f is a prePRF, then all terms in this sum are negligible. Since the number Q of terms is polynomially bounded, the sum is also negligible. So, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PRF}}$ is negligible. As this holds for any PPT adversary $\mathcal{A}$, f is a PRF.