Advanced Cryptography — Final Exam

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

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:

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

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*.

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

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

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 $\mathsf{Gen}(1^s) \to (q, \mathsf{parms}, g)$ 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.

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

Q.3 Show that if the $\mathsf{GenFixed}$-DDH is hard, then the $\mathsf{GenRand}$-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 $S(\mathsf{Gen})$:

pick a large enough sequence of independent fair coin flips ρ
run $\mathsf{Gen}(1^s; \rho) \to (q, \mathsf{params}, g)$
pick $x, y \in \mathbf{Z}_q^*$ with uniform distribution (x, y, and ρ are independent)
set $X = g^x$, $Y = g^y$ {the solution to the problem is g^{xy}}
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 $\mathsf{GenRand}$-CDH solver $\mathcal{A}$, construct a $\mathsf{GenFixed}$-CDH solver $\mathcal{B}$ with similar complexity and

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

Q.5 Given a $\mathsf{GenFixed}$-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, $\mathsf{GenFixed}$ 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 $\mathsf{GenFixed}$-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 $\mathcal{A}$.

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 $w = \mathcal{O}(\log q)$.

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

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

Q.8 Show that $\mathsf{GenFixed}$-CDH is hard if and only if $\mathsf{GenRand}$-CDH is hard.

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

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

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

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

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

$$\Pr[\mathit{\Gamma}^{j}(\mathcal{A}) = 1] = \Pr[\mathit{\Gamma}_0^{\mathsf{prePRF}}(\mathcal{B}_j) = 1]$$
$$\Pr[\mathit{\Gamma}^{j-1}(\mathcal{A}) = 1] = \Pr[\mathit{\Gamma}_1^{\mathsf{prePRF}}(\mathcal{B}_j) = 1]$$

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