Advanced Cryptography — Final Exam
Solution

Serge Vaudenay

26.6.2015

– 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

WARNING: for each question, specially the ones of type "show that...", it is expected that the response contains understandable sentences.
The exam grade follows a linear scale in which each question has the same weight.

1 Davies-Meyer Construction

Given a security parameter λ, we construct two sets G^λ and M^λ and a function C^λ mapping an element $h \in G^\lambda$ and an element $k \in M^\lambda$ to an element $C_k^\lambda(h) \in G^\lambda$. (From now on, and for more readability, we do not write the λ superscript any longer.) We assume that G is given an additive group structure, with neutral element $0 \in G$. As an instance, we assume that $G = \{0,1\}^\lambda$. We assume a block cipher C on the block space G and the key space M: given $k \in M$ and $h \in G$, it encrypts h into $C_k(h)$. We define a keyed function F by

$$F_m(h) = C_m(h) + h$$

We define the following games, played by a polynomially bounded algorithm $\mathcal{A}^{\mathcal{O}}$ interacting with an oracle $\mathcal{O}$:

Game Γ_0:
1: pick $m \in M$ with uniform distribution
2: run $c = \mathcal{A}^{\mathcal{O}_m}$
3: return c

oracle query $\mathcal{O}_m(h)$:
1: return $C_m(h)$

Game Γ_1:
1: pick F^* a random function from G to G with uniform
2: run $c = \mathcal{A}^{\mathcal{O}_{F^*}}$
3: return c

oracle query $\mathcal{O}_{F^*}(h)$:
1: return $F^*(h)$

Game Γ_2:
1: pick C^* a random permutation of H with uniform distribution
2: run $c = \mathcal{A}^{\mathcal{O}_{C^*}}$
3: return c

oracle query $\mathcal{O}_{C^*}(h)$:
1: return $C^*(h)$

We let p_i be the probability that Γ_i returns 0. We say that C is a *pseudorandom function (PRF)* if for any polynomially bounded $\mathcal{A}$ we have that $p_1 - p_0$ is negligible. We say that C is a *pseudorandom permutation (PRP)* if for any polynomially bounded $\mathcal{A}$ we have that $p_2 - p_0$ is negligible.

We define two more oracles.

oracle query $\mathcal{O}_1(h)$:
1: if h is not new, answer as previously (by keeping a table of previous queries)
2: else pick a random $h^* \in G$ and return h^*

oracle query $\mathcal{O}_2(h)$:
1: if h is not new, answer as previously (by keeping a table of previous queries)
2: else pick a random $h^* \in G$ which is different from all previously drawn values and return h^*

We let Γ'_i be the game
1: run $c = \mathcal{A}^{\mathcal{O}_i}$
2: return c

and let p'_i be the probability that it returns 0.

Q.1 Show that for any $\mathcal{A}$, we have $p_1 = p'_1$ and $p'_2 = p_2$.

$\mathcal{O}_1$ defines the lazy sampling technique. It makes $\mathcal{O}_{F^*}$ and $\mathcal{O}_1$ generate the same distribution. So, $p_1 = p'_1$.
Similarly, $\mathcal{O}_{C^*}$ and $\mathcal{O}_2$ generate the same distribution. So, $p'_2 = p_2$.

Q.2 Let B be the event that the oracle $\mathcal{O}_1$ picks some h^* which was previously drawn. Show that $\Pr[B]$ is negligible.

Given the number q of fresh queries, we have less than q^2 pairs (h, h'). For each pair, the probability that $F^*(h) = F^*(h')$ is $2^{-\lambda}$. So, $\Pr[B] \leq q^2 2^{-\lambda}$. Since q is polynomially bounded, $\Pr[B]$ is negligible in terms of λ.

Q.3 Show that $p'_2 - p'_1$ is negligible.
HINT: show that $\Pr[\Gamma'_2 = 0] = \Pr[\Gamma'_1 = 0 | \neg B]$.

(This is actually the difference lemma.)
The distribution of h^* in Γ_2' is just the distribution in Γ_1' conditioned to the event B not occurring. So, $p_2' = \Pr[\Gamma_2' = 0] = \Pr[\Gamma_1' = 0 | \neg B]$. Hence,

$$p_2' = \frac{\Pr[\Gamma_1' = 0, \neg B]}{1 - \Pr[B]} \leq \frac{\Pr[\Gamma_1' = 0]}{1 - \Pr[B]} = \frac{p_1'}{1 - \Pr[B]}$$

So, $p_2' - p_1' \leq p_2' \Pr[B] \leq \Pr[B]$. Hence, $p_2' - p_1'$ is negligible due to the previous question.

Q.4 Deduce that if C is a PRP, then C is a PRF as well.

From all previous questions we obtain that $p_2 - p_1$ is negligible.
If C is a PRP, then $p_2 - p_0$ is negligible. Since $p_2 - p_1$ is negligible, then $p_1 - p_0$ is negligible as well. Hence, C is a PRF.

Q.5 If C is a PRF, show that F is a PRF.

Consider $\mathcal{A}$ playing the PRF game Γ with an oracle implementing $F_m(\cdot)$. This oracle can be defined with a nested oracle $\mathcal{O}_m$: $F_m(h) = \mathcal{O}_m(h) + h$. We compare Γ with the same game Γ' in which the nested oracle is replaced by $\mathcal{O}_{F^*}$. I.e., the oracle in Γ' is defined by $h \mapsto F^*(h) + h$. By considering $\mathcal{A}'$ as simulating $\mathcal{A}$ and the oracle except the nested one, we obtain a PRF game on C. Since C is a PRF, the two games have negligible difference $p' - p$ in the probability of success.
Now, compare Γ' with the same game Γ'' based on $\mathcal{O}_{F^*}$. Clearly, $h \mapsto F^*(h) + h$ and $h \mapsto F^*(h)$ have the same distribution when F^* is a uniformly distributed function from H to H. So, the probability of success is the same in both games: $p'' = p'$.
Finally, $p'' - p$ is negligible. So, F is a PRF.

Q.6 (Bonus question)
Do you see any reason why we do not use $(h, k) \mapsto C_k(h)$ as a compression function to construct a hash function

$$H(k_1, \ldots, k_n) = C_{\bar{n}}(C_{k_n}(\cdots C_{k_1}(0) \cdots))$$

where $\bar{n}$ is an element of M encoding the length n of $k_1, \ldots, k_n$), although it is a PRF? HINT: what would Ralph Merkle or Ivan Damgård say?

We use compression functions in the Merkle-Damgård construction in which we can prove that if the compression function is collision resistant, then the hash function is collision resistant. If we use $(h, k) \mapsto C_k(h)$ as a compression function, we cannot argue that it is collision resistant since for all k, $(C_k^{-1}(y), k)$ maps to the same value y.
So, we would like to use a collision resistant compression function. The function $(h, k) \mapsto C_k(h)$ is not but the Davies-Meyer one is likely to be so.

2 Fiat-Shamir Revisited (Again)

This exercise is inspired from Bernhard-Pereira-Warinschi, How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios*, Asiacrypt 2012, LNCS vol. 7658, Springer.*

Throughout this exercise, we consider some prime number q and some element g generating a multiplicative group G of order q. We assume that basic operations (multiplication, inversion, comparison) are easy but that the discrete logarithm problem is hard.

We consider the Schnorr Σ-protocol for the relation R defined by

$$R(y, x) \iff g^x = y$$

for $y \in G$ and $x \in \mathbf{Z}_q$. In the Σ-protocol, the prover picks $k \in \mathbf{Z}_q$ and sends $r = g^k$. The verifier picks $e \in \mathbf{Z}_q$ and sends it to the prover. The prover answers by $s = ex + k \bmod q$. The verifier checks that $ry^e = g^s$. The *regular* Fiat-Shamir transform constructs a non-interactive proof of knowledge from a Σ protocol by using a random oracle H. We consider here the *weak* Fiat-Shamir which is defined as follows:

$\mathsf{Proof}(y, x; k)$**:** compute $r = g^k$, $e = H(r)$, $s = ex + k \bmod q$. The output is (r, s).
$\mathsf{Verify}(y, r, s)$**:** check that $ry^{H(r)} = g^s$. If this passes, the output is accept. Otherwise, the output is reject.

Here, we assume that the random oracle H returns elements of $\mathbf{Z}_q$.

Q.1
- What is the difference between $\mathsf{Proof}/\mathsf{Verify}$ and the Schnorr signature scheme?
- Show that it is equivalent.
- What is the difference between the weak Fiat-Shamir transform and the regular Fiat-Shamir transform?
- Apply the regular Fiat-Shamir transform to the Schnorr proof.

The Schnorr signature uses $H(m, r)$ instead of $H(r)$, where m is the message to sign.
Another difference is that the Schnorr signature uses (e, s) instead of (r, s). It is equivalent since we can compute (r, s) from (e, s) by $r = y^{-e}g^s$ and (e, s) from (r, s) by $e = H(y, r)$.
In the regular Fiat-Shamir transform, we use $H(y, r)$ instead of $H(r)$.
If we apply it to the Schnorr protocol, we obtain
$\mathsf{Proof}(y, x; k)$**:** *compute $r = g^k$, $e = H(y, r)$, $s = ex + k \bmod q$. The output is (r, s).*
$\mathsf{Verify}(y, r, s)$**:** *check that $ry^{H(y,r)} = g^s$. If this passes, the output is* accept*. Otherwise, the output is* reject*.*

Q.2 We study the properties of the weak Fiat-Shamir transform on the Schnorr protocol.

Q.2a Show that the above Schnorr protocol satisfies the special soundness property. Deduce that it is a proof of knowledge of the discrete logarithm of y.

Given the parameters g and q, the instance y, and two transcripts (r, e, s) and (r, e', s') sharing the same r but with $e \neq e'$, we have $ry^e = g^s$ and $ry^{e'} = g^{s'}$ so $y = g^{\frac{s-s'}{e-e'}}$. Hence, we can extract $x = \frac{s-s'}{e-e'} \bmod q$ which is the discrete logarithm of y. So, the Schnorr protocol satisfies special soundness.
It was proven in the course that this implies that the protocol is a proof of knowledge: the extractor essentially rewinds the prover so that it uses the same r twice but use two different e and e' to obtain s and s' and deduce x.
Normally, the Fiat-Shamir transform of this protocol becomes a non-interactive proof of knowledge. One point here is that we use the weak *Fiat-Shamir transform and we will see that we do not obtain a non-interactive proof of knowledge.*

Q.2b In the weak Fiat-Shamir transform, y is not taken into account to compute e. Consequently, it is as if y could be established after e is received.
Show that we can forge a triplet (y, r, s) passing $\mathsf{Verify}(y, r, s)$ and for which we cannot compute the discrete logarithm of y, except in negligible cases.
HINT: first select r and s at random.

We can pick $r \in G$, $s \in \mathbf{Z}_q$ and $y = (r^{-1}g^s)^{\frac{1}{H(r)}}$. Then, $\mathsf{Verify}(y, r, s)$ is equivalent to $ry^{H(r)} = g^s$ which is true for $H(r) \neq 0$.
If $H(r) = 0$, we can just try again with another r until $H(r) \neq 0$.

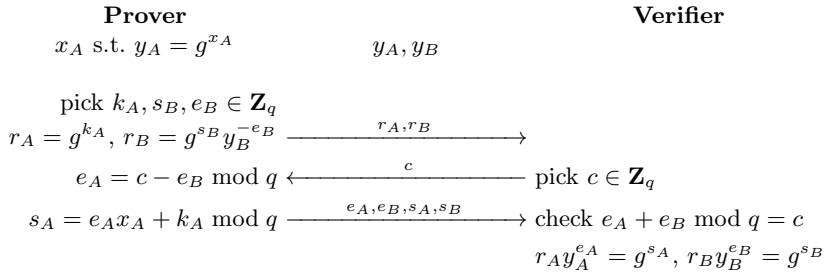
Q.2c Let H' be a random oracle producing elements of G. Prove that an algorithm $\mathcal{A}$ interacting with H' and producing a pair (s, k) such that $H'(s) = g^k$ can be transformed into an algorithm $\mathcal{B}$ which solves the discrete logarithm problem.
HINT: simulate H' by $H'(s) = yg^{H(s)}$.

$\mathcal{B}$ takes y as input and wants to compute x such that $y = g^x$. It simulates $\mathcal{A}$ and H'. The answer $H'(s)$ from H' is simulated by $H'(s) = yg^{H(s)}$. If H produces $\mathbf{Z}_q$ elements with uniform distribution, H' produces elements in G with uniform distribution. If $\mathcal{A}$ succeeds to produce (s, k) such that $H'(s) = g^k$, we deduce $yg^{H(s)} = g^k$ so $y = g^{k-H(s)}$. Hence, $\mathcal{B}$ can output $x = k - H(s) \bmod q$.

Q.2d Inspired by the Fiat-Shamir paradigm, further show that in the forgery of (y, r, s) from Q.2b, we can prove that we ignore the discrete logarithm of y.
HINT: take $r = H'(s)$.

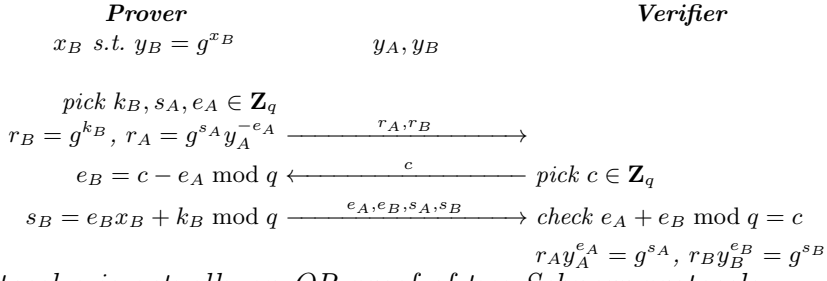
We take $s \in \mathbf{Z}_q$, $r = H'(s) \in G$ by using another random oracle H', and $y = (r^{-1}g^s)^{\frac{1}{H(r)}}$ (if $H(r) = 0$, we try again with another s).
Since $ry^{H(r)} = g^s$, the discrete logarithm of y is $\frac{s-\log_g r}{H(r)} \bmod q$. So, knowing $\log_g y$ is equivalent to knowing $k = \log_g r$. In the previous question, we have shown that knowing k implies solving the discrete logarithm problem. Since we assume that the discrete logarithm is hard, we cannot compute the discrete logarithm of y in the above forgery.

Q.3 We study here consequences on some deniable authentication scheme.
We define the relation $R'(y_A, y_B, x) \Leftrightarrow g^x \in \{y_A, y_B\}$ where x is the witness for the instance (y_A, y_B). We consider the following protocol ρ:



Q.3a We specified ρ when the prover has a witness x_A such that $y_A = g^{x_A}$. Show that there is an alternate prover algorithm for ρ making the protocol work by using a witness x_B such that $y_B = g^{x_B}$.
Have you seen a protocol like this before?

We just invert A and B on the prover side:



The protocol ρ is actually an OR proof of two Schnorr protocols.

Q.3b Prove that ρ satisfies the special soundness property of Σ protocols.

Given two valid transcripts $(r_A, r_B, c^i, e_A^i, e_B^i, s_A^i, s_B^i)$ using the same r_A, r_B with different c^1 and c^2, we obtain $e_A^1 + e_B^1 = c^1$ and $e_A^2 + e_B^2 = c^2$, so either $e_A^1 \neq e_A^2$ or $e_B^1 \neq e_B^2$. If $e_A^1 \neq e_A^2$, we have two transcripts (r_A, e_A^1, s_A^1) and (r_A, e_A^2, s_A^2) for the Schnorr proof from which we extract the discrete logarithm of y_A. If $e_B^1 \neq e_B^2$, we have two transcripts (r_B, e_B^1, s_B^1) and (r_B, e_B^2, s_B^2) for the Schnorr proof from which we extract the discrete logarithm of y_B. So, in any case we extract x such that $g^x \in \{y_A, y_B\}$.

Q.3c Prove that ρ satisfies the honest verifier zero-knowledge property of Σ protocols.

We simulate a transcript by first picking e_A, e_B, s_A, s_B then taking $c = e_A + e_B \bmod q$, $r_A = y_A^{-e_A} g^{s_A}$, and $r_B = y_B^{-e_B} g^{s_B}$.

Q.3d Prove that ρ is a Σ protocol for R (go through the checklist for Σ protocols) and construct a non-interactive proof system for R.

First of all, the protocol respects the structure of Σ protocols: it has three moves, starting from the prover, the verifier picks a random challenge c, and he verifies a condition based on the transcript only. It is complete: if both participants follow their algorithm, the verifier accepts.
To construct a non-interactive proof, we proceed as follows:

1: *pick* $k_A, s_B, e_B \in \mathbf{Z}_q$, $r_A = g^{k_A}$, $r_B = g^{s_B} y_B^{-e_B}$
2: *take* $c = H(y_A, y_B, r_A, r_B)$
3: *take* $e_A = c - e_B \bmod q$, $s_A = e_A x_A + k_A \bmod q$
4: *return* $(r_A, r_B, e_A, e_B, s_A, s_B)$

We used the strong Fiat-Shamir transform here by including y_A and y_B in the random oracle inputs. The verification checks check $e_A + e_B \bmod q = H(y_A, y_B, r_A, r_B)$, $r_A y_A^{e_A} = g^{s_A}$, and $r_B y_B^{e_B} = g^{s_B}$.

Q.3e Alice wants to send an email to Bob using deniable authentication. For this, both Alice and Bob exchange their public keys y_A and y_B and their "proofs" (r_A, s_A) and (r_B, s_B) such that $\mathsf{Verify}(y_A, r_A, s_A)$ and $\mathsf{Verify}(y_B, r_B, s_B)$ hold. Then, Alice modifies the non-interactive proof of Q.3d by adding her message m as input to the random oracle, like for signature schemes, and uses this modified non-interactive proof to authenticate her message.
If (y_A, r_A, s_A) and (y_B, r_B, s_B) were proofs of knowledge of the discrete logarithm of y_A and y_B, show that Bob is ensured that the message comes from Alice and that he cannot forward this evidence to anyone else.
NOTE: a semi-formal argument is OK for this question.

We have shown that the Fiat-Shamir transform produces signature schemes which are existentially unforgeable without a witness. From the view point of Bob, a witness is either his secret x' or Alice's secret x. Assuming that it is hard for Alice to obtain Bob's secret, then the signature must come from Alice. However, from the view point of someone else, the signature can come either from Bob or from Alice. Bob can indeed easily forge the signature with his x' (due to Q.3a). So, Bob cannot prove that the signature comes from Alice.

Q.3f In the above deniable authentication scheme, by using the fact that the weak Fiat-Shamir transform does not make (y_A, r_A, s_A) be a proof of knowledge of the discrete logarithm of y_A, show that Bob can maliciously register (y_B, r_B, s_B) and later show to someone else that the message originated from Alice.
NOTE: a semi-formal argument is OK for this question.

Bob can use Q.2b and Q.2d to create (y_B, r_B, s_B). He can later prove that he ignores the discrete logarithm of x. Then, by showing a signature from Alice and a proof of ignorance of x, he proves that the signature comes from Alice.