Advanced Cryptography — Final Exam

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.

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:	oracle query $\mathcal{O}_m(h)$:
1: pick $m \in M$ with uniform distribution 2: run $c = \mathcal{A}^{\mathcal{O}_m}$ 3: return c	1: return $C_m(h)$

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

Game Γ_2:

- pick C^* a random permutation of H with uniform distribution
- run $c = \mathcal{A}^{\mathcal{O}_{C^*}}$
- return c

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

- 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 $\mathcal{O}_1(h)$:

- if h is not new, answer as previously (by keeping a table of previous queries)
- else pick a random $h^* \in G$ and return h^*

oracle query $\mathcal{O}_2(h)$:

- if h is not new, answer as previously (by keeping a table of previous queries)
- 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

- run $c = \mathcal{A}^{\mathcal{O}_i}$
- 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 $p'_2 = p_2$.

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.

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

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

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

2 Fiat-Shamir Revisited (Again)

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

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

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.

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.

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.

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 $H'(s) = yg^{H(s)}$.

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 $r = H'(s)$.

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:

Prover		**Verifier**
x_A s.t. $y_A = g^{x_A}$	y_A, y_B	
pick $k_A, s_B, e_B \in \mathbf{Z}_q$		
$r_A = g^{k_A}$, $r_B = g^{s_B} y_B^{-e_B}$	$\xrightarrow{r_A, r_B}$	
$e_A = c - e_B \bmod q$	$\xleftarrow{c}$	pick $c \in \mathbf{Z}_q$
$s_A = e_A x_A + k_A \bmod q$	$\xrightarrow{e_A, e_B, s_A, s_B}$	check $e_A + e_B \bmod q = c$
		$r_A y_A^{e_A} = g^{s_A}$, $r_B y_B^{e_B} = g^{s_B}$

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?

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

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

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.

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.

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.