Advanced Cryptography — Final Exam

Serge Vaudenay

28.6.2010

- all documents are allowed
- a pocket calculator is allowed
- communication devices are not allowed
- answers to the exercises must be provided on a separate sheet
- readability and style of writing will be part of the grade
- do not forget to put your name on your copy!

1 Chameleon Hash Function from Σ-Protocol

In this exercise we define a **partial Σ-protocol** a tuple consisting of

- a relation $R(x, w)$ holding on two words x and w called an *instance* and a *witness* respectively, defining a language $L_R = \{x; \exists w \quad R(x, w)\}$;
- a function $a = P(x, w; r_P)$ computing a commitment a on an instance x, a witness w, and random coins r_P;
- a samplable domain E_x called *set of challenges*;
- a function for $z = P(x, w, e; r_P)$ computing a *response* z on an instance x, a witness w, a challenge e, and random coins r_P;
- a verification relation $V(x, a, e, z)$ on an instance x, a commitment a, a challenge e, and a response z.

We further define by A_x and Z_x the set of all possible values for a and z, respectively. These objects shall satisfy the following properties:

- $R(x, w)$, $P(x, w; r_P)$, $P(x, w, e; r_P)$, and $V(x, a, e, z)$ can be evaluated in a time which is polynomial with respect to the length of the instance x;
- we can generate a random element of E_x with uniform distribution in a time which is polynomial with respect to the length of x;
- for every x, w, r_p, e such that $e \in E_x$ and $R(x, w)$ holds, if we define $a = P(x, w; r_P)$ and $z = P(x, w, e; r_P)$, then $V(x, a, e, z)$ holds;

The partial Σ-protocol is executed as follows:

- the prover uses as input (x, w) such that $R(x, w)$ holds and receives some random coins r_p;
- the verifier uses as input x and receives some random coins to generate $e \in E_x$ uniformly distributed;
- the prover sends $a = P(x, w; r_P)$ to the verifier;
- the verifier sends e to the prover;
- the prover sends $z = P(x, w, e; r_P)$ to the verifier;
- the verifier checks that $V(x, a, e, z)$ holds and accept.

In the case of any deviation (e.g. the final check or any computation failed), the protocol fails.

Q.1 Which objects are missing to define a Σ-protocol?

We call a **strong Σ-protocol** a partial Σ-protocol together with a function $H_x(e, z) = a$ and a function $E(x, a, e, z, e', z')$ such that

- $H_x(e, z)$ can be evaluated in a time which is polynomial for any $e \in E_x$ and $z \in Z_x$;
- $E(x, a, e, z, e', z')$ can be evaluated in a time which is polynomial for any $a \in A_x$, $e, e' \in E_x$, and $z, z' \in Z_x$;
- we can generate a random elements of Z_x with uniform distribution in a time which is polynomial with respect to the length of x;
- for all x, $e \in E_x$, $z \in Z_x$, $V(x, H_x(e, z), e, z)$ holds;
- honestly executing the partial Σ-protocol makes z uniformly distributed in Z_x and independent from e;
- for all x, $a \in A_x$, $e \in E_x$, and $z \in Z_x$, $V(x, a, e, z)$ holds if and only if $a = H_x(e, z)$;
- for every x, a, e, e', z, z' such that $e, e' \in E_x$, $z, z' \in Z_x$, $(e, z) \neq (e', z')$, $V(x, a, e, z)$ holds, and $V(x, a, e', z')$ holds, if we define $w = E(x, a, e, z, e', z')$, then $R(x, w)$ holds.

Q.2 What is the difference between the hypothesis on E and the special soundness property of Σ-protocols?
Show that a strong Σ-protocol is a Σ-protocol.

Q.3 Show that given x and w such that $R(x, w)$ holds, we can create a collision on the function H_x.

Q.4 Show that given $x \in L_R$, finding a collision on H_x implies finding a witness for $x \in L_R$.
Deduce that if R is such that given $x \in L_R$ it is hard to find w such that $R(x, w)$ holds, we can define a trapdoor collision resistant hash function by using x as a common reference string.

Q.5 Recall the Goldwasser-Micali-Wigderson Σ-protocol based on graph isomorphism.
Show that the Golwasser-Micali-Wigderson Σ-protocol is not a strong Σ-protocol.

Q.6 Recall the Fiat-Shamir Σ-protocol.
Show that the Fiat Shamir Σ-protocol is not a strong Σ-protocol.

Q.7 Recall the Schnorr Σ-protocol.
Show that the Schnorr Σ-protocol is a strong Σ-protocol.
Deduce a trapdoor hash function based on this protocol. Does it remind you something?

2 Instances of the ElGamal

Let p be a large prime number and g be an element of $\mathbf{Z}_p^*$. We denote by q the order of g. We let $\mathcal{G}$ be a subgroup of $\mathbf{Z}_p^*$ which include g. We let $\mathcal{M} = \{0,1\}^\ell$ be the message space. We assume an injective function $e : \mathcal{M} \to \mathcal{G}$ which is called an *embedding function*. We further assume that given a random $m \in \mathcal{M}$, $e(m)$ "looks like" uniformly distributed in $\mathcal{G}$. In this exercise, we consider the ElGamal cryptosystem using domain parameters (p, g, q, e) with different choices on how to select them. Namely, a secret key is a value $x \in \mathbf{Z}_q$, its public key is $y = g^x \bmod p$. For any message $m \in \mathcal{M}$, the encryption of m with public key y is a pair (u, v) such that $u = g^r$ with $r \in \mathbf{Z}_q$ random and $v = e(m)y^r$. The decryption of (u, v) with secret key x is $m = e^{-1}(vu^{-x})$.

Q.1 We assume here that g is a generator of $\mathbf{Z}_p^*$. What is the value of q?
Is the cryptosystem IND-CPA secure? Why?

Q.2 We assume here that q is prime and that $\mathcal{G} = \mathbf{Z}_p^*$.
Is the cryptosystem IND-CPA secure? Why?

Q.3 We assume here that q is a large prime but much smaller than p, and that $\mathcal{G}$ is generated by g.
Is the cryptosystem IND-CPA secure? Why?
In practice, how to propose an efficient embedding function e?

Q.4 We assume here that $p = 1 + 2q$ with q prime and that $\mathcal{G}$ is generated by g.
Is the cryptosystem IND-CPA secure? Why?
Show that $\mathcal{G}$ is the subgroup of all quadratic residues in $\mathbf{Z}_p^*$.
Compute $\left(\frac{-1}{p}\right)$.
Deduce that for any $x \in \mathbf{Z}_p^*$ then either x or $-x$ is in $\mathcal{G}$.
Finally, if $\ell = \lfloor \log_2 q \rfloor$, propose a practical embedding function e.

3 Trusted Agent Setup

In this exercise we consider protocols between two participants A and B involving a trusted third participant C. This participant C is assumed to always be honest, which means that he always follows the protocol he is expected to follow. We further assume that communication channels between the participants are authenticated and protect message integrity.

As an example we can specify this way a commitment protocol in which A can commit on a value x to B:

- A's protocol: send the input x to C
- C's protocol: receive some message x' form A then send a message "commit" to B
- B's protocol: receive a message form C and check that it is "commit"

For the opening phase:

- A's protocol: send a message "open" to C
- C's protocol: receive a message from A, check that it is "open", and send x' to B
- B's protocol: receive a message x'' from C and take it as the output

The protocol is *perfectly hiding* since what B receives in the commit phase is independent from x. The protocol is *perfectly binding* since whatever x'' that B receives from C must be equal to x' (due to message integrity and authentication assumption) sent by C which must be the x' received by C in the first phase (due to the honesty assumption on C), which must be equal to x (due to message integrity and authentication assumption).

Q.1 Propose a zero-knowledge proof of knowledge for a relation $R(x, w)$ in which A can prove to B that he knows w such that the predicate $R(x, w)$ holds, being given a public x. The protocol shall be perfectly complete, perfectly sound, and perfectly zero-knowledge. Specify the protocol/algorithm for A, B, and C.

We now change the setup assumption a bit. We assume that there are several participants $C_1, \ldots, C_n$ such that when queried for the first time with a program p they boot on this program and follow it honestly. Additionally, participants can query them with a special query "Code" on which they return by p so that we can see what program they were booted with. These extra participants called *trusted agents* can have a pseudorandom generator embedded.

Q.2 Redo the previous question in this model.

Q.3 For any proof system in the standard model (that is, who does not involve any trusted agent), show that a malicious adversary using a trusted agent can break the deniability property. Namely, the malicious verifier can later prove that w such that $R(x, w)$ holds is known by someone.