Advanced Cryptography — Final Exam

Serge Vaudenay

26.6.2014

- duration: 3h00
- documents are 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 Security Interference

We consider a zero-knowledge proof of knowledge π in which a prover $P(x, w)$ holding a witness w for an instance x can convince a verifier $V(x)$ that he knows w such that the relation $R(x, w)$ holds.

We construct a mutual-authentication protocol π' in which two participants $A(x, w)$ and $B(x, w)$ share the secret w for the instance x. The protocol π' runs as follows:

1: A and B execute π: A runs $P(x, w)$ and B runs $V(x)$
2: if $V(x)$ accepted for B, B sends w to A
3: A accepts if and only if w is correct

Q.1 Show that there is an algorithm $\mathcal{E}^{C^*}$ calling C^* as a subroutine such that, for every input z and every malicious algorithm $C^*(x, z)$, if $C^*(x, z)$ interacts with $B(x, w)$ and $B(x, w)$ accepts, then $\mathcal{E}^{C^*}(x, z) = w'$ such that $R(x, w')$ holds.

Q.2 Show that there is an algorithm $\mathcal{S}^{C^*}$ calling C^* as a subroutine such that, for every input z and every malicious algorithm $C^*(x, z)$, if $C^*(x, z)$ interacts with $A(x, w)$ and $A(x, w)$ accepts, then $\mathcal{S}^{C^*}(x, z) = w'$ such that $R(x, w')$ holds.
WARNING: $\mathcal{S}$ does not know w, a priori.

Q.3 Show that π and π' do not compose: even though a malicious verifier learns nothing from $P(x, w)$ and a malicious Alice learns nothing from $B(x, w)$, in a network where $P(x, w)$ and $B(x, w)$ are two honest participants, show that a malicious participant can extract w.

2 Distance Bounding

We consider a *distance-bounding protocol*, in which there is a prover P and a verifier V sharing a secret x. The protocol starts with an initialization phase which consists of setting up a matrix $a \in \{0, 1\}^{n \times 2}$ to be shared between P and V. (We will see later how this initialization phase works.) Then, we have n rounds of time-critical challenge-response exchanges: in the ith round, V sends a random $c_i \in \{1, 2\}$ to which P answers by $r_i = a_{i,c_i}$. V accepts the response if it is correct and if the elapsed time between sending c_i and receiving r_i is at most $\frac{2B}{C}$, where B is a distance bound and C is the speed of light. We say that the protocol *succeeds* if V accepts the response in all rounds. We assume that the time used to compute is negligible against the time of flight of messages. So, a honest prover within a distance up to B can pass all rounds. We want the protocol to resist to two types of threats:

- In a concurrent setting with several honest provers using key x and several honest verifiers using key x, including a target verifier $\mathcal{V}$, if there is no prover within a distance up to B to $\mathcal{V}$, no malicious participant $\mathcal{A}$ can make a protocol with $\mathcal{V}$ succeed. If this holds, we say the protocol is *secure*.
- A malicious prover within a distance larger than B to the verifier cannot make the protocol succeeds. In what follows we call this threat a *distance fraud*.

We stress that the above malicious participant starts by ignoring x while the malicious prover in distance fraud knows x.

Q.1 (General security upper bound.)
We assume that the initialization phase is such that a computed by a honest verifier is a uniformly distributed matrix no matter any malicious environment.

Q.1a We consider a honest verifier $\mathcal{V}$ and a malicious participant $\mathcal{A}$ with no other participant. Show that $\mathcal{A}$ can make the protocol succeed with probability 2^{-n}.

Q.1b We consider a man-in-the-middle $\mathcal{A}$ between a honest prover P and a honest verifier $\mathcal{V}$ who are within a distance larger than B.
Show that $\mathcal{A}$ can make the protocol succeed with probability $\left(\frac{3}{4}\right)^n$.
HINT: assume that $\mathcal{A}$ can make a challenge-response exchange with P before he receives the first challenge from $\mathcal{V}$.

Q.2 (General distance fraud.)
We make the same assumption on a.

Q.2a Show that a far-away malicious prover who sends random r_i's can make a distance fraud with probability 2^{-n}.
HINT: assume that the malicious prover can predict when c_i will be sent by the verifier.

Q.2b Find another strategy so that the distance fraud works with probability $\left(\frac{3}{4}\right)^n$.

Q.3 (Distance fraud for a dedicated protocol.)
We consider a protocol with the following initialization phase: The verifier selects a nonce $N_V \in \{0,1\}^n$ and sends it to the prover. The prover selects a nonce $N_P \in \{0,1\}^n$ and sends it to the verifier. Both compute $a_{.,1} = \mathsf{PRF}_x(N_V)$ by using a pseudorandom function PRF and $a_{.,2} = a_{.,1} \oplus N_P$.
Make a distance fraud which succeeds with probability 1.

Q.4 (Security of a dedicated protocol.)
We now modify the initialization phase by having $a_{.,1} = \mathsf{PRF}_x(N_P, N_V)$ and $a_{.,2} = a_{.,1} \oplus x$.

Q.4a Show that a malicious man-in-the-middle between P and V (who are within a distance up to B) can extract x_i.
HINT: assume that the adversary can see if the protocol succeeded on the side of V.

Q.4b In a setting with n provers and $n+1$ verifiers, show that the protocol is insecure: we can have an attack succeeding with probability 1.
HINT: use the previous question!

3 On a Weak Fiat-Shamir Transform

Throughout this exercise, we consider some (G, q, g) depending on a security parameter t, where G is a group, q is a prime number, and g is an element of G of order q. We assume that $q > 2^t$, that the size of q is polynomially bounded, and that we can make basic operations (multiplication, inversion, comparison) in G in polynomial time.

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

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

In the Σ-protocol, the prover picks $k \in \mathbf{Z}_q$ and sends $r = g^k$. The verifier picks $e \in \{1, \ldots, 2^t\}$ and sends it to the prover. The prover answers by $s = ex + k \bmod q$. The verifier checks that $ry^e = g^s$. In the *weak* Fiat-Shamir transform constructs a non-interactive proof system by using a random oracle H 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.

We assume that the random oracle H returns elements of $\mathbf{Z}_q$ which are uniformly distributed. A proof (r, s) for y is aimed at producing evidence that the algorithm which forged (r, s) knows x such that $g^x = y$.

Q.1 Construct an efficient algorithm $\mathcal{A}^H$ invoking H and producing a triplet (y, r, s) such that $y \neq 1$, y is spanned by g, and $\mathsf{Verify}(y, (r, s)) = \mathsf{accept}$, with probability larger than $1 - 2^{-t}$.

Q.2 In the (strong) Fiat-Shamir construction, the query to H is $y \| r$ instead of r alone. In this case, say why the previous attack does not work.

Q.3 We let $y \neq 1$ spanned by g be *fixed*.
Let $\mathcal{A}^H$ be an algorithm invoking H. We consider the following experiment:

1: pick ρ and H
2: set $(r, s) = \mathcal{A}^H(\rho)$
3: set $\mathsf{Out} = \mathsf{Verify}(y, (r, s))$

The goal of this question is to show that there is a generic transform $\mathcal{T}$ such that for any polynomially bounded algorithm $\mathcal{A}^H$ such that $\Pr[\mathsf{Out} = \mathsf{accept}] \geq 1 - 2^{-t}$ (over the distribution of ρ and H) $\mathcal{B} = \mathcal{T}(\mathcal{A})$ is a polynomially bounded algorithm producing the discrete logarithm of y.

Q.3a Let E be the event that during the computation of $\mathcal{A}$, a query to H was made with the final value r of the proof. Show that $\Pr[E] \geq 1 - 2 \times 2^{-t}$.
HINT: first show that $\Pr[\mathsf{Out} = \mathsf{accept} | \neg E] \leq 2^{-t}$.

Q.3b We consider a simulator for $\mathcal{A}$ and H. The simulation of H is done following the lazy sampling technique (i.e., fresh random coins are flipped only when needed). The simulation defines a tree of the partial views of the simulator, where each node corresponds to the view when a fresh call to H is made, and the q sons of the node correspond to the possible coin flips to respond to the query. A leaf λ corresponds to the end of the execution of $\mathcal{A}$. The event $\mathsf{Succ}(\lambda)$ holds if $\mathcal{A}$ outputs some (r, s) making the verification accept *and* r was queried to H. If $\mathsf{Succ}(\lambda)$ holds, we let $\mathsf{dist}(\lambda)$ be the ancestor of λ corresponding to the $H(r)$ oracle call. Otherwise, we let $\mathsf{dist}(\lambda) = \lambda$.
We let p be the probability that a random descent in the tree ends to a leaf λ such that $\mathsf{Succ}(\lambda)$ holds. We let d be the expected length of a random descent. Given a node ν in the tree, we let Y be a random leaf obtained by a random descent starting from ν. We let $f(\nu) = \Pr[\mathsf{Succ}(Y), \mathsf{dist}(Y) = \nu]$. We let X be a random leaf obtained by a random descent from the root. We let Y be a random leaf obtained by a random descent from $\mathsf{dist}(X)$. The Forking Lemma says that $E(f(\mathsf{dist}(X))) \geq \frac{p^2}{2d}$.
Show that if d is polynomially bounded, we can make a polynomial-time algorithm walking in this tree and producing with probability at least $\frac{p^2}{2d} - (1 - p) - 2^{-t}$ two

leaves X and Y such that $\mathsf{Succ}(X)$ and $\mathsf{Succ}(Y)$ hold, $\mathsf{dist}(X) = \mathsf{dist}(Y)$, and with X and Y in different subtrees connected to $\mathsf{dist}(X) = \mathsf{dist}(Y)$.

Q.3c Show that by using $\mathcal{A}^H$ as a subroutine we can make a polynomial-time algorithm $\mathcal{B}$ which outputs x such that $g^x = y$ with a probability which is not negligible.

Q.4 The previous reduction works for attacks $\mathcal{A}$ in which y is determined at the beginning. Assuming that now y is not determined and we consider an attack producing valid (y, r, s) triplets. Assume that for each such attack $\mathcal{A}$, there exists an algorithm $\mathcal{B}$ such that for each View, if $\mathcal{A}(\mathsf{View}) = (y, r, s)$ such that $\mathsf{Verify}(y, (r, s))$ accepts, $\mathcal{B}(\mathsf{View}) = x$ such that $y = g^x$.

Show that we can solve the discrete logarithm problem: we can construct a polynomial-time algorithm $\mathcal{C}$ such that given z as input, it outputs $\mathcal{C}(z)$ such that $g^{\mathcal{C}(z)} = z$.

HINT: Construct some $\mathcal{A}$ like in Q.1 but with $r = z$.