Advanced Cryptography — Final Exam

Serge Vaudenay

20.6.2011

- duration: 3h00
- any document allowed
- a pocket calculator is allowed
- communication devices are not allowed
- readability and style of writing will be part of the grade
- it is unlikely we will answer any technical question during the exam
- do not forget to put your full name on your copy!

I Σ-Protocol for $\mathcal{P}$

We consider an alphabet Z, a polynomial P, and a predicate R. We assume that R can be computed in polynomial time. Given $x \in Z^*$, we let

$$R_x = \{w \in Z^*; R(x,w) \text{ and } |w| \leq P(|x|)\}$$

where $|x|$ denotes the length of x. We define the language L from R by

$$L = \{x \in Z^*; R_x \neq \emptyset\}$$

Q. In this question, we assume that there is an algorithm $\mathcal{A}$ such that for any $x \in L$, we obtain $\mathcal{A}(x) \in R_x$ and that for any $x \in Z^*$, the running time of $\mathcal{A}(x)$ is bounded by $P(|x|)$.
Construct a Σ-protocol for L. Carefully specify all protocol elements and prove all properties which must be satisfied.

II OR Proof

Let $Z = \{0,1\}$ be an alphabet. We consider two Σ-protocols Σ_1 and Σ_2 for two languages L_1 and L_2 over the alphabet Z defined by two predicates R_1 and R_2. We assume that Σ_1 and Σ_2 use the same challenge set E which is given a group structure with a law $+$. For Σ_i, $i \in \{1,2\}$, we denote $\mathcal{P}_i$ the prover algorithm, V_i the verification predicate, $\mathcal{E}_i$ the extractor, and $\mathcal{S}_i$ the simulator.

Q.1 (**AND proof**) Construct a Σ protocol $\Sigma = \Sigma_1 \text{ AND } \Sigma_2$ for the language defined by

$$R((x_1,x_2),(w_1,w_2)) \Longleftrightarrow R_1(x_1,w_1) \text{ AND } R_2(x_2,w_2)$$

(**OR proof**) In the remaining of the exercise, we now let

$$R((x_1,x_2),w) \Longleftrightarrow R_1(x_1,w) \text{ OR } R_2(x_2,w)$$

This predicate defines a new language L. We construct a new Σ-protocol $\Sigma = \Sigma_i \text{ OR } \Sigma_2$ for L by

- $\mathcal{P}((x_1,x_2),w;r_1,r_2)$ finds out i such that $R_i(x_i,w)$ holds, sets $j = 3-i$, then picks a random $e_j \in E$ and runs $\mathcal{S}_j(x_j,e_j;r_1) = (a_j,e_j,z_j)$. Then, it runs $\mathcal{P}(x_i,w;r_2) = a_i$ and yield (a_1,a_2).
- Upon receiving e, $\mathcal{P}((x_1,x_2),w,e;r_1,r_2)$ sets $e_i = e - e_j$, runs $\mathcal{P}(x_i,w,e_i;r_2) = z_i$ and yields (e_1,e_2,z_1,z_2).

The verification predicate is

$$V((x_1,x_2),(a_1,a_2),e,(e_1,e_2,z_1,z_2)) \Longleftrightarrow \begin{cases} e = e_1 + e_2 \text{ AND} \\ V_1(x_1,a_1,e_1,z_1) \text{ AND} \\ V_2(x_2,a_2,e_2,z_2) \end{cases}$$

Q.2 Show that Σ is complete and works in polynomial time.
Q.3 Construct an extractor $\mathcal{E}$ for Σ and show that is works, in polynomial time.
Q.4 Construct a simulator $\mathcal{S}$ for Σ and show that is works, in polynomial time.

III Smashing SQUASH-0

We consider an access control protocol called SQUASH-0 in which a client and a server hold a secret key K. In the protocol, the server sends a challenge C. The client must respond with

$$S = (\mathsf{stoi}(C \oplus K))^2 \bmod N$$

for a given modulus N, where stoi is a function transforming a bitstring into an integer by $\mathsf{stoi}(\varepsilon) = 0$ for the zero-length bitstring ε, and

$$\mathsf{stoi}(b\|s) = b + 2 \times \mathsf{stoi}(s)$$

for any bit $b \in \{0,1\}$ and any bitstring s. By convention, the least significant bit has position 0. We further assume that N is larger than K and C.

Q.1 Let c_i be -1 raised to the power of the bit position i in C. Let k_i be -1 raised to the power of the bit position i in K.
Show that

$$S = \left(\frac{1}{4}\sum_{i,j} 2^{i+j} c_i c_j k_i k_j - \frac{2^\ell - 1}{2}\sum_i 2^i c_i k_i + \frac{(2^\ell - 1)^2}{4}\right) \bmod N$$

where ℓ is the bitlength of N.
In what follows, we assume that $N = 2^\ell - 1$. Deduce

$$S = \left(\frac{1}{4}\sum_{i,j} 2^{i+j} c_i c_j k_i k_j\right) \bmod N$$

Q.2 Deduce that by using about ℓ^2 challenges and their responses, an adversary could recover K by solving a linear system of $O(\ell^2)$ equations with $\frac{\ell(\ell-1)}{2}$ unknowns.
As an example, consider $\ell = 1\,024$. What is the complexity of the attack?
Hint: define $\kappa_{i,j} = k_i k_j$.

Q.3 Given a function φ mapping a bitstring of length d to a real number, we define

$$\hat{\varphi}(V) = \sum_x (-1)^{x \cdot V} \varphi(x)$$

where $\cdot$ denotes the dot product between two bitstrings and the sum goes on all bitstrings x of length d. For the function $\varphi(x) = (-1)^{x \cdot U}$, show that $\hat{\varphi}(V) = 2^d$ if $V = U$ and $\hat{\varphi}(V) = 0$ otherwise. We write it $\hat{\varphi}(V) = 2^d 1_{V=U}$.

Q.4 In a chosen challenge attack, an adversary creates d challenges $C^1, \ldots, C^d$ and all linear combinations of these challenges. Namely, $C(x_1 \ldots x_d) = x_1 C^1 \oplus \cdots \oplus x_d C^d$. Given a d-bit vector x, we thus define $C(x)$. We write x as an argument of S and c_i as well so that $S(x)$ is the response to challenge $C(x)$ and $c_i(x)$ is -1 raised to the power of the bit position i in $C(x)$. Let U_i be the d-bit vector consisting of the bit at position i of $C^1, \ldots, C^d$.
Deduce that

$$\hat{S}(V) = \frac{1}{4}\sum_{i,j} 2^{d+i+j} k_i k_j 1_{V = U_i \oplus U_j}$$

Hint: observe $c_i(x) = (-1)^{x \cdot U_i}$ and use Q.1 then Q.3.

Q.5 With the same notations, we assume that the function mapping a non-ordered pair $\{i, j\}$ with $i \neq j$ to $U_i \oplus U_j$ behaves like a random function. We further assume that d is pretty small. For each V, estimate the number of non-ordered pairs $\{i, j\}$ with $i \neq j$ such that $V = U_i \oplus U_j$.
Deduce that we get 2^d equations modulo N with $\ell(\ell-1)2^{-d-1}$ unknowns $\kappa_{i,j}$ on average taking values in $\{-1, +1\}$.

Q.6 We take $d = 2\log_2 \ell$ and solve each equation by exhaustive search. Deduce a chosen-challenge attack to break the algorithm.
How many chosen challenges does it use, asymptotically?
What is its complexity?

IV PIF Implies PAF

We consider a function family F_k taking inputs of length λ, making outputs of length λ, and where the key k is also of length λ. We consider the two following games:

Game $\mathsf{PIF}(\mathcal{A}, 1^\lambda)$:

pick some random coins k of length λ
pick ρ
run $\mathcal{A}(\rho) \to x$
if $|x| \neq \lambda$, output 0 and stop
pick a random bit b
if $b = 0$ **then**
  compute $y = F_k(x)$
else
  pick a random y of λ bits
end if
run $\mathcal{A}(y; \rho) \to b'$
output $b \oplus b' \oplus 1$

Game $\mathsf{PAF}(\mathcal{A}, 1^\lambda)$:

pick some random coins k of length λ
pick ρ
pick a random x of length λ
compute $y = F_k(x)$
run $\mathcal{A}(y; \rho) \to x'$
output $1_{x=x'}$

We say that F_k is PIF-secure (resp. PAF-secure) if for all polynomially bounded $\mathcal{A}$, we have that $\Pr[\mathsf{PIF}(\mathcal{A}, 1^\lambda) = 1] - \frac{1}{2}$ (resp. $\Pr[\mathsf{PAF}(\mathcal{A}, 1^\lambda) = 1]$) is a negligible function in terms of λ.

Q. Show that if F_k is PIF-secure, then it is PAF-secure.
Hint: based on a PAF-adversary $\mathcal{A}$ and some coins $\rho' = r' \| \rho \| b''$, define $\mathcal{A}'(\rho') = x$ picked at random from r' then $\mathcal{A}'(y, \rho') = 1$ if $\mathcal{A}(y; \rho) = x$ and $\mathcal{A}'(y, \rho') = b''$ otherwise. By considering $\mathcal{A}'$ as a PIF-adversary, look at the link between $\Pr[\mathsf{PIF}(\mathcal{A}', 1^\lambda) = 1] - \frac{1}{2}$ and $\Pr[\mathsf{PAF}(\mathcal{A}, 1^\lambda) = 1]$.