

Solution of Exercise Sheet 4

Out: Tue, Dec 15, 2009

Due: Tue, Jan 5, 2010, 10am

Problem 1: Computational implementations (10 points) for public key encryption

Let (K, E, D) be an IND-CCA secure encryption scheme with probabilistic polynomial-time key-generation K and encryption E , and deterministic polynomial-time decryption D . Assume that for all $m \in \{0, 1\}^*$, $k \in \mathbb{N}$,

$$\Pr[m = m' : (ek, dk) \leftarrow K(1^k), c \leftarrow E(ek, m), m' \leftarrow D(dk, c)] = 1. \quad (1)$$

(That is, decryption never fails.)

Construct a computational implementation A for the symbolic model \mathbf{M}_{pke} (as in Section 8.1 in the lecture notes) that satisfies the implementation conditions given in Definition 44 in the lecture notes.

Note: Do not assume anything about the encryption scheme except what is explicitly given in the problem statement. In particular, do not assume that all encryption/verification keys have the same length, that only k -bits of randomness are used, that the encryption key can be efficiently extracted from the ciphertext, ...

Solution. We use the following notation:

Let $\langle x, y \rangle$ denote an encoding of the pair (x, y) for bitstrings x, y . We assume that the encoding is efficiently computable, efficiently invertible, and length regular.¹ Let $[m]_\ell := 0^{\ell-|m|}1m$ and $[m]_\ell := \perp$ if $|m| > \ell$. Let $\mathbf{N}, \mathbf{ek}, \mathbf{dk}, \mathbf{enc}, \mathbf{genc}$ denote arbitrary distinct bitstrings satisfying $|\mathbf{enc}| = |\mathbf{genc}|$.

To simplify notation, we write patterns $[\cdot]$, $[\cdot]_\ell$ and $\langle \cdot, \cdot \rangle$. A bitstring c matches $[x]_\ell$ if there is some x with $[x]_\ell = c$. A bitstring c matches $[x]$ if there is some x and some ℓ such that $[x]_\ell = c$. A bitstring c matches $\langle x, y \rangle$ if there are x, y such that $\langle x, y \rangle = c$.

When defining a function, we assume that the function fails (returns \perp) if some pattern match or some recursive function call fails.

Let G be a pseudo-random generator taking k bits and returning a stream of bits. Let $G_k(r)$ denote the first k bits of $G(r)$. Let $G'_k(r)$ denote $G(r)$ without the first k bits. I.e., $G(r) = G_k(r) \| G'_k(r)$.

Let $(K_{ek}(r), K_{dk}(r)) := K$ when K runs with randomness $G(r)$. Let $E'(e, m, r) := E(e, m)$ when E runs with randomness $G'_k(r)$.

¹A simple though inefficient example of such an encoding would be $\langle x, y \rangle := 0^{|x|}1xy$.

Let ℓ_{ek} be a polynomial upper bound on the length of the output of K_{ek} . Let $\ell_{enc}(l)$ be a polynomial upper bound on the length of $E(e, \langle m, r \rangle)$ with $|e| \leq \ell_{ek}$, $|m| \leq l$, $|r| = k$.

We can now define the computational implementation:

$$\begin{aligned}
A_N &:= \langle \mathbb{N}, r \rangle \text{ for } r \xleftarrow{R} \{0, 1\}^k \\
A_{dk}(\langle \mathbb{N}, r \rangle) &:= \langle \mathbf{dk}, r \rangle && (|r| = k) \\
A_{ek}(\langle \mathbb{N}, r \rangle) &:= \langle \mathbf{ek}, [K_{ek}(r)]_{\ell_{ek}} \rangle && (|r| = k) \\
A_{enc}(\langle \mathbf{ek}, [e]_{\ell_{ek}} \rangle, m, \langle \mathbb{N}, r \rangle) &:= \langle \mathbf{enc}, [E'(e, \langle m, G_k(r) \rangle, r)]_{\ell_{enc}(|m|)}, G_k(r), [e]_{\ell_{ek}} \rangle && (|r| = k) \\
&\text{(if } E' \text{ fails in prec. case)} := \langle \mathbf{genc}, [0]_{\ell_{enc}(|m|)}, G_k(r), [e]_{\ell_{ek}} \rangle && (|r| = k \text{ and } E' \text{ fails}) \\
A_{dec}(\langle \mathbf{dk}, r' \rangle, \langle \mathbf{enc}, c', \hat{r}, [e]_{\ell_{ek}} \rangle) &:= ([c] := c', \langle m, r \rangle := D(K_{dk}(r'), c), \text{return } m) \\
&&& (|r'| = k, K_{ek}(r') = e, c' = [c]_{\ell_{enc}(|m|)}, |\hat{r}| = k, r = \hat{r}) \\
A_{ekof}(\langle \mathbf{enc}, [c], r, [e]_{\ell_{ek}} \rangle) &:= \langle \mathbf{ek}, [e]_{\ell_{ek}} \rangle && (|r| = k) \\
A_{ekof}(\langle \mathbf{genc}, [c], r, [e]_{\ell_{ek}} \rangle) &:= \langle \mathbf{ek}, [e]_{\ell_{ek}} \rangle && (|r| = k)
\end{aligned}$$

In all other cases, the functions A_x return \perp . In particular, $A_{garbage}$ and $A_{garbageEnc}$ always return \perp .

Comments: We have included the rule that A_{enc} may produce bitstrings $\langle \mathbf{genc}, \dots \rangle$ because E might fail for invalid (but valid looking) encryption keys, but A_{enc} may not fail for any encryption key, as long as that key satisfies the purely syntactic criterion for encryption keys given below. We have included $G_k(r)$ in the output of A_{enc} to ensure that two ciphertexts are different with overwhelming probability. It is not sufficient to include $G_k(r)$ within the actual ciphertext because if the encryption key is not generated honestly, there is no guarantee that the ciphertext will contain the plaintext or any randomness (the ciphertext might even be a constant). In the case that E' succeeds, we have also included $G_k(r)$ in the plaintext. This is necessary in order not to lose the IND-CCA security: If $G_k(r)$ was only attached to the ciphertext (i.e., $\langle \mathbf{enc}, [E'(e, m, r)]_{\ell_{enc}(|m|)}, G_k(r), [e]_{\ell_{ek}} \rangle$), then an adversary in the IND-CCA game could change one valid ciphertext $\langle \mathbf{enc}, [E'(e, m, r)]_{\ell_{enc}(|m|)}, G_k(r), [e]_{\ell_{ek}} \rangle$ into another ciphertext $\langle \mathbf{enc}, [E'(e, m, r)]_{\ell_{enc}(|m|)}, r^*, [e]_{\ell_{ek}} \rangle$ that would still be accepted by A_{dec} ; this would violate IND-CCA security.

We define the set of nonces as $Nonces_k := \{\langle \mathbb{N}, r \rangle : r \in \{0, 1\}^k\}$. The set of ciphertexts as $\{\langle \mathbf{enc}, [c]_l, r, [e]_{\ell_{ek}} \rangle, \langle \mathbf{genc}, [c]_l, r, [e]_{\ell_{ek}} \rangle : c, e \in \{0, 1\}^*, r \in \{0, 1\}^k, l \geq 0\}$. The set of encryption keys as $\{\langle \mathbf{ek}, [e]_{\ell_{ek}} \rangle : c, e \in \{0, 1\}^*\}$. The set of decryption keys as $\{\langle \mathbf{dk}, r \rangle : r \in \{0, 1\}^k\}$.

Implementation conditions 1–7 follow directly from the definition of the computational implementation A . Implementation condition 8 additionally uses (1). To see that implementation condition 10 holds, note that $G_k(r) \neq G_k(r')$ with overwhelming probability for $r, r' \xleftarrow{R} \{0, 1\}^k$.

We now sketch that implementation condition 9 holds, i.e., that the encryption scheme $(K^\circ, E^\circ, D^\circ)$ constructed from $A_{ek}, A_{dk}, A_{enc}, A_{dec}$ (as described in implemen-

tation condition 9) is IND-CCA secure. Let $(K^\dagger, E^\dagger, D^\dagger)$ be defined like $(K^\circ, E^\circ, D^\circ)$, except that whenever $G(r)$ is computed, a random string is used instead (and in particular, since $G_k(r)$ and $G'_k(r)$ are computed from $G(r)$, these now return random strings, too). Since G is a pseudo-random generator, $(K^\dagger, E^\dagger, D^\dagger)$ and $(K^\circ, E^\circ, D^\circ)$ cannot be distinguished by a polynomial-time machine, thus it is sufficient to show that $(K^\dagger, E^\dagger, D^\dagger)$ is IND-CCA secure.

Consider the following oracle \mathcal{E}^\dagger : It initially picks $(ek^\dagger, dk^\dagger) \leftarrow K^\dagger$ and $b \xleftarrow{R} \{0, 1\}$. Upon query **getkey** it answers ek^\dagger . Upon query (\mathbf{enc}, m_0, m_1) with $|m_0| = |m_1|$, it computes and returns $c_*^\dagger := E^\dagger(ek^\dagger, m_b)$. We allow only one **enc**-query. Upon query $(\mathbf{dec}, c^\dagger)$ with $c^\dagger \neq c_*^\dagger$, it computes and returns $D^\dagger(dk^\dagger, c^\dagger)$.

The oracle \mathcal{E} is defined analogously, but uses the IND-CCA secure encryption scheme (K, E, D) instead.

Assume that $(K^\dagger, E^\dagger, D^\dagger)$ is not IND-CCA secure. Then there is an adversary Adv such that $|\Pr[\text{Adv}^{\mathcal{E}^\dagger} = b] - \frac{1}{2}|$ is not negligible. Now, we define an alternative implementation of \mathcal{E}^\dagger . The alternative implementation invokes the oracle \mathcal{E} . Upon query **getkey**, it retrieves ek from \mathcal{E} and returns $\langle \mathbf{ek}, [ek]_{\ell_{ek}} \rangle$. Upon query (\mathbf{enc}, m_0, m_1) with $|m_0| = |m_1|$, it picks $r_* \xleftarrow{R} \{0, 1\}^k$ and sends $(\mathbf{enc}, \langle m_0, r_* \rangle, \langle m_1, r_* \rangle)$ to \mathcal{E} . When getting the ciphertext c_* from \mathcal{E} , it returns $c_*^\dagger := \langle \mathbf{enc}, [c_*]_{\ell_{\text{enc}}(|m_0|)}, r_*, [ek]_{\ell_{ek}} \rangle$. (ek is retrieved from \mathcal{E} using a **getkey** query if necessary.) When \mathcal{E}^\dagger gets a query $(\mathbf{dec}, c^\dagger)$ with $c^\dagger \neq c_*^\dagger$, it does the following:

- Parse $\langle \mathbf{enc}, [c], \hat{r}, [\hat{ek}]_{\ell_{ek}} \rangle := c^\dagger$ with $|\hat{r}| = k$. If this fails, return \perp .
- If $c \neq c_*$, send (\mathbf{dec}, c) to \mathcal{E} , and then compute and return the plaintext as A_{dec} would have done.
- If $c = c_*$ and $\hat{r} \neq r_*$, return \perp . (The original construction of \mathcal{E}^\dagger would also have returned \perp because $c = c_*$ decrypts to $\langle m_b, r_* \rangle$.)
- If $c = c_*$ and $\hat{ek} \neq ek$, return \perp . (The original \mathcal{E}^\dagger would also have returned \perp because A_{dec} checks whether the encryption key attached to the ciphertext is the right one.)
- If $c = c_*$, but for $\langle \cdot, c', \cdot, \cdot \rangle := c^\dagger$ it holds that $c' \neq [c]_{\ell_{\text{enc}}(|m_0|)}$, return \perp . (The original \mathcal{E}^\dagger would also have returned \perp : The plaintext of $c = c_*$ is m_b . Thus A_{dec} would check $c' = [c]_{\ell_{\text{enc}}(|m_b|)}$. But this check fails because thus $c' \neq [c]_{\ell_{\text{enc}}(|m_0|)} = [c]_{\ell_{\text{enc}}(|m_b|)}$.)
- Other cases do not occur since then we would have $c^\dagger = c_*^\dagger$, and thus the $(\mathbf{dec}, c^\dagger)$ -query would be ignored by \mathcal{E}^\dagger .

The modified construction of \mathcal{E}^\dagger is perfectly indistinguishable from the original construction, thus for the modified construction we have that $|\Pr[\text{Adv}^{\mathcal{E}^\dagger} = b] - \frac{1}{2}|$ is not negligible. Since \mathcal{E}^\dagger in turn invokes \mathcal{E} , we can construct an adversary B from Adv and \mathcal{E}^\dagger such that $|\Pr[B^\mathcal{E} = b] - \frac{1}{2}| = |\Pr[\text{Adv}^{\mathcal{E}^\dagger} = b] - \frac{1}{2}|$ is not negligible. This contradicts the IND-CCA

security of (K, E, D) . Thus $(K^\dagger, E^\dagger, D^\dagger)$ is IND-CCA secure and thus $(K^\circ, E^\circ, D^\circ)$ is IND-CCA secure. Hence implementation condition 9 is satisfied. .noitulon

Problem 2: A complete analysis (10 points)

Consider the following protocol:

- Protocol participants are Alice and Bob. They communicate over an insecure channel (i.e., all messages are effectively sent to and received from the adversary).
- Let (ek_B, dk_B) be a preshared encryption key pair. Assume that Alice knows ek_B and Bob knows dk_B . (The actual distribution of these keys is not part of the protocol.)
- Alice chooses a nonce N_A .
- Alice encrypts N_A using ek_B and sends the resulting ciphertext c to Bob.
- Bob raises an event *begin* and then decrypts c , extracts the nonce N_A and sends N_A to Alice.
- When Alice receives N_A , she raises an event *end*.

We say the protocol is secure if the event *end* never occurs unless the event *begin* has occurred before.

- (a) Model the above protocol as a CoSP protocol Π . Model the security of the protocol (i.e., *end* only occurs after *begin*) as a trace property \wp .

Note: Make sure that you do not implicitly assume some kind of synchronization (i.e., that the first step of Alice occurs before the first step of Bob and the first step of Bob occurs before the second step of Alice).

Solution. The CoSP protocol Π is depicted in Figure 2. Let *begin* be the set of all copies of the input-node in B . Let *end* be the set of all copies of the nondeterministic node in A_2 . Let

$$\wp := \{(\nu_1, \dots, \nu_t) : \forall i. (\nu_i \in \textit{end} \Rightarrow \exists j < i. \nu_j \in \textit{begin})\}.$$

.noitulon

- (b) Show that Π symbolically satisfies \wp . You may do this either by hand or using a theorem prover like SPASS (depending on your preferences and mood).

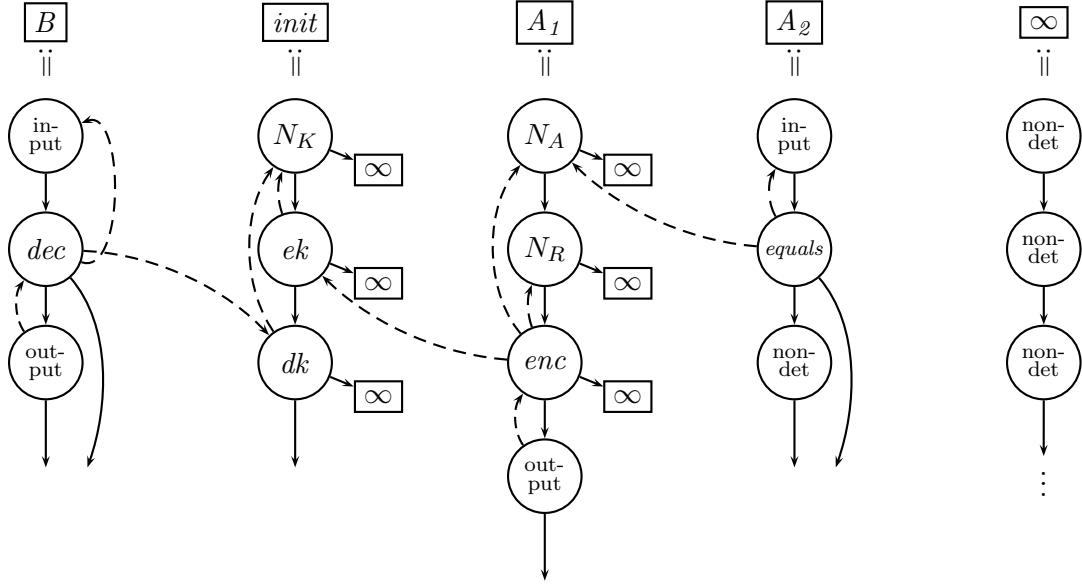


Figure 1: Building blocks for the CoSP tree.

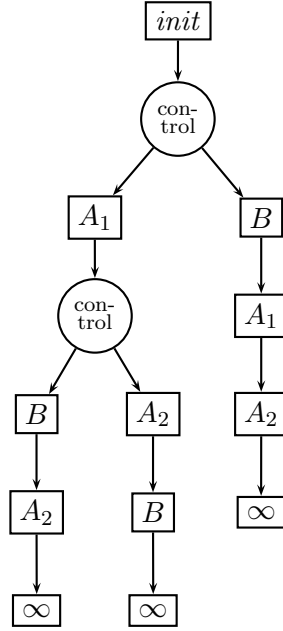


Figure 2: The CoSP protocol II. The boxes refer to the subtrees specified in Figure 1. If a subtree x has two outgoing edges, and a subtree y is attached to the outgoing edge of x in this diagram, we assume that a separate copy of y is attached to each outgoing edge of x . For example, in the middle path, both the outgoing edge of the nondeterministic node of A_2 and the no-edge of the *equals*-node are connected to a copy of B . When an argument edge (dashed) from some node a in one subtree goes to a node b in another subtree (e.g., the *equals*-node in A_2 refers to the N_A -node in A_1), then we assume that this connection is drawn from a to the copy of b that is on the same path as a .

Solution. Assume that Π does not symbolically satisfy \wp . Then there is a symbolic node trace $t' \notin \wp$, i.e., $t' = (\nu_1, \dots, \nu_s, \nu, \dots)$ where $\nu_1, \dots, \nu_s \notin \text{begin}$, and $\nu \in \text{end}$. Since a symbolic node trace necessarily is a path in Π , the node ν must be the nondeterministic node in A_2 in the middle path in Figure 2. (All other copies of that node are preceded by a copy of the input node of B .)

For this node to be reached, the *equals*-node in A_2 must succeed. By definition of *equals*, this implies that the term at the input node in A_2 (call it x) equals the term at the N_A -node, i.e., N_A . Thus it holds that $S \vdash N_A$ where S is the set of terms at the output nodes on the path to x . The only such output node is the output node of A_1 , this output node has the term $\text{enc}(\text{ek}(N_K), N_A, N_R)$. Thus $\text{enc}(\text{ek}(N_K), N_A, N_R) \vdash N_A$. Since $N_K, N_A, N_R \in \mathbf{N}_P$, it is easy to see from the definition of \vdash that $\text{enc}(\text{ek}(N_K), N_A, N_R) \not\vdash N_A$ (by structural induction with the induction hypothesis that all terms derivable from $\text{enc}(\text{ek}(N_K), N_A, N_R)$ contain N_K and N_A only within subterms $\text{enc}(\text{ek}(N_K), N_A, N_R)$). **.noitulos**

- (c) Show that Π computationally satisfies \wp , i.e., that the protocol described above is secure even in a cryptographic sense.

Hint: Use Lemma 9 from the lecture notes.

Solution. It is easy to see that the protocol Π is key-safe (Definition 43 in the lecture notes) and that \wp is a trace property (i.e., prefix closed and efficiently decidable). Assume a computational implementation A satisfying the implementation conditions from Definition 44 in the lecture notes. (For the existence of such a computational implementation, see Problem 1.) Then, by Lemma 9 in the lecture notes (and Definition 38 in the lecture notes), (Π, A) computationally satisfies \wp . **.noitulos**