

Exercise Sheet 3

Out: Thu, Nov 19, 2009

Due: Tue, Dec 8, 2009, 10am

Problem 1: Passive computational soundness for signatures

In the following, we will extend the passive computational soundness result from the lecture to signatures. There is a lot of text below, but do not be frightened, most of it is just a slight extension of concepts you already know from the passive computational soundness of encryption.

Consider the following symbolic model $\mathbf{M} = \mathbf{M}_{enc, sig} = (\mathbf{C}, \mathbf{N}, \mathbf{T}, \mathbf{D}, \vdash)$:

The constructors are $\mathbf{C} := \{enc/3, sig/3, sk/1, vk/1\}$. Here $sig(K, M, R)$ intuitively denotes a signature using signing key K , message M , and randomness R . And $sk(N)$ and $vk(N)$ intuitively denote a signing/verification key pair.

The destructors are $\mathbf{D} := \{dec/2, verify/2, vkofsig/1\}$ with

$$\begin{aligned} dec(x, enc(x, y, z)) &= y \\ verify(vk(x), sig(sk(x), y, z)) &= y \\ vkofsig(sig(sk(x), y, z)) &= vk(x) \end{aligned}$$

The message type \mathbf{T} is given by the following grammar:

$$\mathbf{T} ::= \mathbf{N} | enc(\mathbf{N}, \mathbf{T}, \mathbf{N}) | vk(\mathbf{N}) | sk(\mathbf{N}) | sig(sk(\mathbf{N}), \mathbf{T}, \mathbf{N}).$$

The nonces \mathbf{N} and the deduction relation \vdash are standard. (Note that the existence of the $vkofsig$ - and $verify$ -destructors implies that the adversary can extract the verification key and the message from a signature.)

Definition 1 (Existential unforgeability) *A signature scheme (K, S, V) consisting of probabilistic polynomial-time algorithms K (key-pair generation), S (signing) and V (verification) is existentially unforgeable if for any polynomial-time oracle machine Adv , the following probability is negligible in the security parameter k :*

$$\Pr[V(1^k, vk, \sigma^*) = m^* \text{ and } m^* \text{ is new} : (sk, vk) \leftarrow K(1^k), (m^*, \sigma^*) \leftarrow \text{Adv}^{\mathcal{S}}(1^k, vk)].$$

Here \mathcal{S} is an oracle which on input m returns $S(1^k, sk, m)$. We say that m^* is new if m^* has never been given to \mathcal{S} as input.

(Note: Usually, this definition is formulated somewhat differently. The algorithm V is given $(1^k, vk, \sigma^*, m^*)$ and outputs 1 if σ^* is a valid signature for m^* . We use the present

variant where V extracts m^* from the signature, because it fits nicer to the symbolic modeling. If verification fails, V outputs \perp .)

We say a computational implementation A is existentially unforgeable if (K, S, V) is existentially unforgeable where (i) all A_N for $N \in \mathbf{N}$ are identical, (ii) $K(1^k) := (\text{let } r = A_N(1^k) \text{ in } (A_{sk}(1^k, r), A_{vk}(1^k, r)))$ for some $N \in \mathbf{N}$ (i.e., K produces a signing and a verification key using the same randomness). (iii) $S(1^k, \text{key}, m) := A_{sig}(1^k, \text{key}, m, A_N(1^k))$ for some $N \in \mathbf{N}$. (iv) $V(1^k, \text{key}, \sigma) := A_{verify}(1^k, \text{key}, \sigma)$.

Definition 2 (P.s.p. for signatures) A passive secrecy property for signatures is a pair $\wp := (\text{sig}(sk(K), M, *), (m_1, \dots, m_n))$ with $K, M \in \mathbf{N}$ and $m_1, \dots, m_n \in \mathbf{T}$.

We say \wp holds symbolically iff for all $R \in \mathbf{N}$ we have $\{\text{sig}(sk(K), M, R)\} \not\subseteq \text{sig}(sk(K), M, R)$.

We say \wp holds computationally if for all probabilistic polynomial-time algorithms Adv , we have that $\text{Succ}(\wp, \text{Adv}, k)$ is negligible in k . Here Succ is defined as follows: First, compute r_N ($N \in \mathbf{N}$) and $b_i = \beta(m_i)$ as in Definition 21 in the lecture notes. Invoke $\sigma^* \leftarrow \text{Adv}(1^k, b_1, \dots, b_n)$. We say the adversary wins if $A_{verify}(1^k, A_{vk}(1^k, r_K), \sigma^*) = r_M$. $\text{Succ}(\wp, \text{Adv}, k)$ is the probability that the adversary wins.

Prove the following theorem. You may add additional assumptions as needed, as long as they are reasonable (such as fresh randomness, key-cycle-freeness, IND-CPA, existential unforgeability, etc.).

Theorem 1 (Passive computational soundness for signatures) Fix a passive secrecy property for signatures $\wp = (\text{sig}(K, M, *), (m_1, \dots, m_n))$. Let A be a computational implementation of $\mathbf{M}_{enc, sig}$. [Insert additional assumptions here.]

Then, if \wp holds symbolically, \wp holds computationally.

Hint: The main structure of the proof is very similar to that of the proof of Theorem 6 in the lecture notes. (Just be careful that you do not miss if some condition or definition needs to be changed due to the introduction of signatures in the symbolic model.) The proof of Claim 2, however, changes. To show Claim 2, you need to modify Succ_t such that it uses the oracle \mathcal{S} from the definition of existential unforgeability for terms $vk(K)$ and $\text{sig}(K, \dots, \dots)$. From the fact that \wp holds symbolically, you can derive that r_M is new (in the sense of the definition of existential unforgeability).

Problem 2: Correctness of computational implementations

In the tutorial (November 5), we have derived the following definition of the correctness of computational implementations:

Let a symbolic model $\mathbf{M} = (\mathbf{C}, \mathbf{N}, \mathbf{T}, \mathbf{D}, \vdash)$ be given.

Definition 3 (Condition) A condition is a term of the form

$$U ::= \mathbf{N} | C(U, \dots, U) | D(U, \dots, U)$$

where $C \in \mathbf{C}, D \in \mathbf{D}$.

We say a condition U holds symbolically if $eval(U) \neq \perp$ where $eval$ is defined by

$$\begin{aligned} eval(N) &:= N && \text{if } N \in \mathbf{N} \\ eval(C(U_1, \dots, U_n)) &:= C(eval(U_1), \dots, eval(U_n)) \\ &&& \text{if } C/n \in \mathbf{C}, C(eval(U_1), \dots, eval(U_n)) \in \mathbf{T} \\ eval(D(U_1, \dots, U_n)) &:= D(eval(U_1), \dots, eval(U_n)) \\ &&& \text{if } D/n \in \mathbf{D}, D(eval(U_1), \dots, eval(U_n)) \neq \perp \\ eval(x) &:= \perp && \text{otherwise.} \end{aligned}$$

(Remember that $D(eval(U_1), \dots, eval(U_n))$ means the result of applying the partial function D to the terms $eval(U_i)$.)

Definition 4 (Holds computationally) We say a condition U holds computationally if $\Pr[ceval(U) \neq \perp]$ is overwhelming in k where $ceval$ is defined by

$$\begin{aligned} ceval(N) &:= r_N && \text{if } N \in \mathbf{N} \\ ceval(F(U_1, \dots, U_n)) &:= A_F(1^k, ceval(U_1), \dots, ceval(U_n)) \\ &&& \text{if } F/n \in \mathbf{C} \cup \mathbf{D}, A_F(1^k, ceval(U_1), \dots, ceval(U_n)) \neq \perp \\ ceval(x) &:= \perp && \text{otherwise.} \end{aligned}$$

Here all r_N are chosen as $r_N \leftarrow A_N(1^k)$.

(There was also a definition “does not hold computationally”, but we omit that one here.)

Let \mathbf{M}' be defined like \mathbf{M}_{enc} from Definition 23 in the lecture notes, except that we add the destructor *equals* with $equals(x, x) = x$ and $equals(\dots) = \perp$ otherwise. (Note, \mathbf{M}_{enc} also has pairs, do not forget these.)

Prove the following theorem. You may add additional assumptions as needed, as long as they are reasonable (e.g., that $A_{equals}(x, y) = x$ iff $x = y$ and \perp otherwise).

Theorem 2 (Correctness of computational implementations) Fix a condition U with respect to \mathbf{M}' . Let A be a computational implementation of \mathbf{M}' . [**Insert additional assumptions here.**]

Then, if U holds symbolically, U holds computationally.

Hint: Show the following fact first (by induction over the structure of U): $ceval(U) = ceval(eval(U))$.