

Solution of 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 $\{m_1, \dots, m_n\} \not\subseteq \text{sig}(sk(K), M, R)$.

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 $sig(K, \dots, \dots)$. From the fact that \wp holds symbolically, you can derive that r_M is new (in the sense of the definition of existential unforgeability).

Solution. To formulate the conditions in the theorem, we first need to extend certain definitions to the case with signatures.

First, we extend the definition of key-cycles-freeness. The new definition is the same as in Definition 24 in the lecture notes, except that we say that N' occurs as a non-key in a term $M \in \mathbf{T}$ if M is of the following grammar:

$$M ::= N' | \text{enc}(\mathbf{N}, M, \mathbf{T}) | \text{enc}(\mathbf{N}, \mathbf{T}, M) | \\ vk(\mathbf{N}) | sk(\mathbf{N}) | sig(sk(\mathbf{N}), M, \mathbf{T}) | sig(sk(\mathbf{N}), \mathbf{T}, M).$$

We also extend the definition of fresh randomness (Definition 28 in the lecture notes): $\wp = (\text{sig}(\text{sk}(K), M, *), (m_1, \dots, m_n))$ uses *fresh randomness for encryptions* if the following holds for all nonces $R \in \mathbf{N}$: If m_i has a subterm $\text{enc}(M_1, M_2, R)$ for some i, M_1, M_2 , then R occurs exactly once as a subterm of (M, m_1, \dots, m_n) . (The difference to Definition 28 in the lecture notes is that there we considered (m_1, \dots, m_n) instead of (M, m_1, \dots, m_n) .)

Furthermore, we say $\wp = (\text{sig}(\text{sk}(K), M, *), (m_1, \dots, m_n))$ uses *fresh randomness for signatures* if the following holds for all nonces $R \in \mathbf{N}$:

- If m_i has a subterm $\text{sig}(t_1, t_2, R)$ for some i, t_1, t_2 , then R occurs exactly once as a subterm of (m_1, \dots, m_n) .
- If m_i has a subterm $\text{sk}(R)$ or $\text{vk}(R)$ for some i , then R occurs only within subterms $\text{sk}(R)$ and $\text{vk}(R)$ in (m_1, \dots, m_n) .

The theorem to be proven, with the additional assumptions added, is the following.

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 $\mathbf{M}_{\text{enc}, \text{sig}}$.*

- \wp is key-cycle-free.
- A is length-regular (Definition 25 in the lecture notes).
- A is failure-free (Definition 26 in the lecture notes).
- A has unpredictable nonces (Definition 27 in the lecture notes).
- \wp uses fresh randomness for encryptions.
- \wp uses fresh randomness for signatures.
- A is IND-CPA secure (Definition 30 in the lecture notes).
- A is existentially unforgeable (Definition 1).
- $m_1, \dots, m_n \vdash M$.
- A is injective in the sense that for $t_1 \neq t_2$, with overwhelming probability $\beta(t_1) \neq \beta(t_2)$.

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

Fix a polynomial-time Adv. Let N_1, \dots, N_t be the hidden nonces occurring in A , ordered as in the definition of key-cycle-freeness.

We define the function β_i as follows:

$$\begin{aligned} \beta_i(N) &:= r_N && (N \in \mathbf{N}) \\ \beta_i(\text{enc}(N_j, M, R)) &:= A_{\text{enc}}(r_{N_j}, 0^{\ell(M)}, r_R) && (j \leq i) \\ \beta_i(\text{enc}(N, M, R)) &:= A_{\text{enc}}(r_N, \beta_i(M), r_R) && (\text{otherwise}) \\ \beta_i(\text{sig}(x, M, R)) &:= A_{\text{sig}}(\beta_i(x), \beta_i(M), r_R) \\ \beta_i(\text{vk}(N)) &:= A_{\text{vk}}(\beta_i(N)) \\ \beta_i(\text{sk}(N)) &:= A_{\text{sk}}(\beta_i(N)) \end{aligned}$$

Here ℓ is a function such that $\ell(M) = |\beta(M)|$. (This function exists because of the length-regularity and failure-freeness of A .)

We define Succ_i like Succ , except that it uses β_i instead of β . (Same for Succ_i^* , etc.)

Claim 1 For all $i = 0, \dots, t-1$, we have that

$$\left| \text{Succ}_i(\wp, \text{Adv}, k) - \text{Succ}_{i+1}(\wp, \text{Adv}, k) \right|$$

is negligible in k .

Claim 2 $\text{Succ}_t(\wp, \text{Adv}, k)$ is negligible in k .

From these two claims, the theorem follows.

In the proof of Claim 1, we proceed as in the lecture, except that the definitions of β_i^* , $\beta_i^{F^*}$, and β_i^F have additional rules for terms $\text{sig}(\dots)$, $\text{vk}(\dots)$, and $\text{sk}(\dots)$. These rules are the same as for β_i above.

The first step is to show that $\text{Succ}_i = \text{Succ}_i^*$. As in the lecture, we check that β_i does not use $r_{N_{i+1}}$ or one of the r_R with R occurring in an $\text{enc}(N_{i+1}, \dots, R)$ -term except in the computation of $\beta_i(\text{enc}(N_{i+1}, M, R))$. For r_R this follows from the fresh randomness for encryptions. To see that $r_{N_{i+1}}$ is not accessed (except as the key used for producing the ciphertext $\beta_i(\text{enc}(N_{i+1}, M, R))$), we distinguish the different possible occurrences of N_{i+1} in any of the terms m_s :

- N_{i+1} is a subterm of the second argument of $t := \text{enc}(N_j, \dots, \dots)$ with $j < i+1$: Then, by construction, $\beta_i(t)$ does not recurse into the second argument and thus does not invoke $\beta_i(N_{i+1})$.
- N_{i+1} is a subterm of the second argument of $t := \text{enc}(N_j, \dots, \dots)$ with $j \geq i+1$: By definition of key-cycle-freeness this does not occur.
- N_{i+1} occurs as the first argument of vk or sk , or as the third argument of sig : This does not occur due to the fact that \wp uses fresh randomness for signatures (while N_{i+1} occurs as the first argument of an enc -constructor).

The proof that $\text{Succ}_i^{F^*} = \text{Succ}_i^F$ is analogous.

The step showing that $|\text{Succ}_i^* - \text{Succ}_i^{F^*}|$ is negligible is unchanged from the lecture (and uses the IND-CPA property).

The step showing that $\text{Succ}_i^F = \text{Succ}_{i+1}$ is as in the lecture.

Thus Claim 1 holds in our case.

We now show Claim 2. In the following, we write short “ $\not\vdash t$ ” for “ $M, m_1, \dots, m_n \not\vdash t$ ”. By assumption, \wp holds symbolically, thus $m_1, \dots, m_n \not\vdash \text{sig}(K, M, R)$. By assumption, $m_1, \dots, m_n \vdash M$, thus $\not\vdash \text{sig}(K, M, R)$ for all $R \in \mathbf{N}$. Since $\text{sig}(K, M, R) \in \mathbf{T}$, $K = \text{sk}(L)$ for some $L \in \mathbf{N}$. Since $\not\vdash \text{sig}(\text{sk}(L), M, R)$ for all R and $\vdash M$, we have $\not\vdash L$ and $\not\vdash \text{sk}(L)$.

We say a term t occurs visibly in a term u if $u = C[t]$ where $C ::= \square | \text{enc}(N_v, C, \mathbf{T}) | \text{sig}(C, \mathbf{T}, \mathbf{N}) | \text{sig}(\text{sk}(\mathbf{N}), C, \mathbf{N})$ where N_v stands for nonces occurring in M, m_1, \dots, m_n that are not hidden. Since $\not\vdash L$ and $\not\vdash \text{sk}(L)$, we have that neither L nor $\text{sk}(L)$ occurs visibly in M, m_1, \dots, m_n .

We say a term t occurs as sig -randomness in a term u if it occurs as the third argument of a sig -constructor in u . Same for enc . We say a term t occurs as vk -randomness in

a term u if it occurs as the argument of a vk -constructor in u . Same for sk . We say t occurs as randomness in a term u if it occurs as sig -, enc -, vk -, or sk -randomness in u . We say t occurs as encryption-key in u if it occurs as the first argument of an enc -constructor in u .

Since $sk(L) \notin \mathbf{N}$, $sk(L)$ does not occur as randomness in M, m_1, \dots, m_n . Since \wp has fresh randomness for signatures, L does not occur as sig - or enc -randomness or encryption-key in M, m_1, \dots, m_n .

We define β^S as follows: β^S uses the oracle \mathcal{S} from Definition 1. Let vk^* denote the key vk from Definition 1.

$$\begin{aligned}\beta^S(vk(L)) &:= vk^* \\ \beta^S(sk(L)) &:= \perp \\ \beta^S(sig(sk(L), t, R)) &:= \mathcal{S}(\beta^S(t))\end{aligned}$$

For all other terms, β^S is defined like β_t .

We have that $\text{Succ}^S = \text{Succ}_t$: (Note that Succ_t not only invokes $\beta(m_1), \dots, \beta(m_n)$, but also $\beta(M)$.) The randomness r_R used by the invocations $\beta_t(sig(sk(L), t, R))$ is never used twice because since \wp has fresh randomness for signatures. The value r_L is never accessed except for producing $vk(L)$ and $sk(L)$ since otherwise L would occur visibly or as enc - or sig -randomness in M, m_1, \dots, m_n . $\beta^t(sk(L))$ is never invoked since otherwise $sk(L)$ would occur visibly or as enc - or sig -randomness or encryption-key in M, m_1, \dots, m_n . Thus $\text{Succ}^S = \text{Succ}_t$.

Since $\not\vdash sig(sk(L), M, R)$ for all R , we have that \mathcal{S} is only invoked as $\mathcal{S}(\beta^S(t))$ with $t \neq M$. Since A is injective, we have $\beta(t) \neq \beta(M)$. Using the same argument as for showing that $|\text{Succ} - \text{Succ}^S|$ is negligible, we get that $\beta^S(t) \neq \beta^S(M)$ with overwhelming probability. Thus $\beta^S(M)$ is new in the sense of Definition 1. Thus, since A is existentially unforgeable, the probability that $A_{\text{verify}}(1^k, vk^*, \beta^S(\sigma^*)) = \beta^S(M)$ is negligible. Thus Succ^S is negligible and thus Succ_t is negligible. This shows Claim 2. .noituluo2

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 $\text{eval}(U) \neq \perp$ where eval is defined by

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

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

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

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

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 $\text{equals}(x, x) = x$ and $\text{equals}(\dots) = \perp$ 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 $A_{\text{equals}}(x, y) = x$ 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): $\text{ceval}(U) = \text{ceval}(\text{eval}(U))$.

Solution. The theorem with the additional assumptions is the following:

Theorem 2 (Correctness of computational implementations) Fix a condition U with respect to \mathbf{M}' . Let A be a computational implementation of \mathbf{M}' . Assume the following:

- (1) There are disjoint sets *Nonces*, *Encs*, and *Pairs* (possibly depending on the security parameter k) such that the range of all A_N ($N \in \mathbf{N}$) is contained in *Nonces*, the range of A_{enc} is contained in *Encs*, and the range of A_{pair} is contained in *Pairs*. (E.g., because ciphertexts, pairs, and nonces are tagged with different prefixes to distinguish them.)
 - (2) If $x \notin \text{Nonces}$ or $z \notin \text{Nonces}$, then $A_{enc}(1^k, x, y, z) = \perp$.
 - (3) For all x, y , $A_{fst}(1^k, A_{pair}(1^k, x, y)) = x$ and $A_{snd}(1^k, A_{pair}(1^k, x, y)) = y$
 - (4) If $x \notin \text{Pairs}$, then $A_{fst}(x), A_{snd}(x) = \perp$.
 - (5) For all $k, r \in \text{Nonces}$ and $x, y \in \{0, 1\}^*$, we have $A_{enc}(k, x, r) \neq \perp$ and $A_{pair}(x, y) \neq \perp$.
 - (6) If $(x, y, z) \neq (x', y', z')$, $A_{enc}(1^k, x, y, z) \neq (1^k, x', y', z')$.
 - (7) For all $N \in \mathbf{N}$, $\max_x \Pr[A_N(1^k) = x]$ is negligible.
 - (8) For all $x \neq y$, $A_{equals}(x, x) = x$ and $A_{equals}(x, y) = \perp$.
 - (9) For all x, y , if $x \notin \text{Nonces}$ or $y \notin \text{Encs}$, then $A_{dec}(x, y) = \perp$.
 - (10) For all x, y, z, r , if $x \neq y$ then $A_{dec}(x, A_{enc}(y, z, r)) = \perp$.
 - (11) For all $x, r \in \text{Nonces}$ and all y , $A_{dec}(x, A_{enc}(x, y, r)) = y$.
- Then, if U holds symbolically, U holds computationally.

Proof. To show the theorem, we first show the following claim:

Claim 3 For any condition U , with overwhelming probability we have that $ceval(U) = ceval(eval(U))$ where the lhs is defined iff the rhs is.

We show this claim by induction on the structure of U . Then we distinguish the following cases. (All equalities in the following are interpreted as implying that the lhs is defined iff the rhs is. And we omit the argument 1^k from all algorithms.)

Case 1: “ $U = N \in \mathbf{N}$ ”.

Then $eval(U) = N$ and thus $ceval(U) = ceval(N) = ceval(eval(U))$.

Case 2: “ $U = pair(U_1, U_2)$ ”.

We have

$$\begin{aligned} ceval(U) &= A_{pair}(ceval(U_1), ceval(U_2)) \stackrel{(*)}{=} A_{pair}(ceval(eval(U_1)), ceval(eval(U_2))) \\ &= ceval(pair(eval(U_1), eval(U_2))) = ceval(eval(U)). \end{aligned}$$

Here $(*)$ uses the induction hypothesis.

Case 3: “ $U = enc(U_1, U_2, U_3)$ ”.

Case 3.1: “ $eval(U_i) = \perp$ for some i ”.

By induction hypothesis this implies $ceval(U_i) = ceval(eval(U_i)) = \perp$ for some i . Thus $ceval(U) = \perp$. Furthermore, $eval(U) = \perp$.

Case 3.2: “ $eval(U_1), eval(U_3) \in \mathbf{N}$ ”.

Then $enc(eval(U_1), eval(U_2), eval(U_3)) \in \mathbf{T}$. Hence $eval(U) = enc(eval(U_1), eval(U_2), eval(U_3))$ and $ceval(eval(U)) = A_{enc}(ceval(eval(U_1)), ceval(eval(U_2)), ceval(eval(U_3)))$. Furthermore, by definition $ceval(U) = A_{enc}(ceval(U_1), ceval(U_2), ceval(U_3))$. By induction hypothesis, $ceval(U_i) = ceval(eval(U_i))$ for all i . Thus $ceval(eval(U)) = ceval(U)$.

Case 3.3: “ $eval(U_1) \notin \mathbf{N}$ or $eval(U_3) \notin \mathbf{N}$ ”.

In this case, $enc(eval(U_1), eval(U_2), eval(U_3)) \notin \mathbf{T}$ and thus $eval(U) = \perp$ and thus $ceval(eval(U)) = \perp$. Let $i \in \{1, 3\}$ be such that $eval(U_i) \notin \mathbf{N}$. Since $eval(U_i) \notin \mathbf{N}$, by assumption (1), $ceval(eval(U_i)) \notin \mathbf{Nonces}$. Thus, by induction hypothesis, $ceval(U_i) \notin \mathbf{Nonces}$. By assumption (2), this implies that $ceval(U) = A_{enc}(ceval(U_1), ceval(U_2), ceval(U_3)) = \perp$. So $ceval(U) = \perp = ceval(eval(U))$.

Case 4: “ $U = fst(U')$ ”.

Case 4.1: “ $eval(U') = \perp$ ”.

In this case, $eval(U) = \perp$ and by induction hypothesis, $ceval(U') = ceval(eval(U')) = \perp$.

Case 4.2: “ $eval(U') = pair(M_1, M_2)$ ”.

Then $eval(U) = M_1$. Furthermore,

$$\begin{aligned} ceval(U) &= A_{fst}(ceval(U')) \stackrel{(*)}{=} A_{fst}(ceval(eval(U'))) = A_{fst}(ceval(pair(M_1, M_2))) \\ &= A_{fst}(A_{pair}(ceval(M_1), ceval(M_2))) \stackrel{(3)}{=} ceval(M_1) = ceval(eval(U)) \end{aligned}$$

Here $(*)$ uses the induction hypothesis.

Case 4.3: “*Otherwise*”.

Then $eval(U')$ is not of the form $pair(\dots, \dots)$. Thus, $eval(U) = fst(eval(U')) = \perp$ and hence $ceval(eval(U)) = \perp$. By assumption (1), $ceval(eval(U')) \notin \mathbf{Pairs}$. By induction hypothesis, $ceval(U') \notin \mathbf{Pairs}$. Thus $ceval(U) = A_{fst}(ceval(U')) \stackrel{(4)}{=} \perp = ceval(eval(U))$.

Case 5: “ $U = snd(U')$ ”.

Analogous to the case $U = fst(U')$.

Case 6: “ $U = equals(U_1, U_2)$ ”.

The case $eval(U_1) = \perp$ or $eval(U_2) = \perp$ is analogous as in Case 3.1. Thus we can assume that $eval(U_i), ceval(U_i) \neq \perp$ for $i = 1, 2$.

From (3) we get that A_{pair} is injective. From (6), A_{enc} is injective. From (7), the probability that $r_N = r_{N'}$ for $N \neq N'$ is negligible. Furthermore, from (1), we get that the ranges of A_{enc} and A_{pair} are disjoint and different from the

r_N . Thus for all $M \neq M'$, $M, M' \in \mathbf{T}$, $ceval(M) \neq ceval(M')$. By assumption (8), this implies that $M = M'$ iff $A_{equals}(ceval(M), ceval(M')) \neq \perp$. Furthermore, by induction hypothesis, $ceval(U) = A_{equals}(ceval(U_1), ceval(U_2)) = A_{equals}(ceval(eval(U_1)), ceval(eval(U_2)))$. Thus $ceval(U) \neq \perp$ iff $eval(U_1) = eval(U_2)$. Furthermore $eval(U) \neq \perp$ iff $eval(U_1) = eval(U_2)$. Thus $ceval(U) \neq \perp$ iff $eval(U) \neq \perp$. And in this case, $ceval(U) = ceval(U_1) = ceval(eval(U))$.

Thus, with overwhelming probability, $ceval(U) = ceval(eval(U))$.

Case 7: “ $U = dec(U_1, U_2)$ ”.

Case 7.1: “ $eval(U_i) = \perp$ for some $i = 1, 2$ ”.

Like Case 3.1.

Case 7.2: “ $eval(U_1) \notin \mathbf{N}$ ”.

Then $eval(U) = dec(eval(U_1), eval(U_2)) = \perp$. Since $eval(U_1) \notin \mathbf{N}$, by assumption (1), $ceval(eval(U_1)) \notin Nonces$. Thus, by induction hypothesis, $ceval(U_1) \notin Nonces$. By assumption (9), this implies $ceval(U) = A_{dec}(ceval(U_1), ceval(U_2)) = \perp = ceval(eval(U))$.

Case 7.3: “ $eval(U_2)$ is not of the form $enc(\dots)$ ”.

Then $eval(U) = dec(eval(U_1), eval(U_2)) = \perp$. Since $eval(U_2) \neq enc(\dots)$, by assumption (1), $ceval(eval(U_2)) \notin Encs$ and by induction hypothesis $ceval(U_2) \notin Encs$. Thus by assumption (9), $ceval(U) = A_{dec}(ceval(U_1), ceval(U_2)) = \perp = ceval(eval(U))$.

Case 7.4: “ $eval(U_1) = K \in \mathbf{N}$ and $eval(U_2) = enc(L, t, R)$ for some $L \neq K$ ”.

Then $eval(U) = dec(eval(U_1), eval(U_2)) = \perp$. By induction hypothesis, $ceval(U_1) = ceval(K) = r_K$, and $ceval(U_2) = ceval(enc(L, t, R)) = A_{enc}(r_L, m, r_R)$ with $m := ceval(t)$. Furthermore, by assumption (7), $r_K \neq r_L$. Thus, by assumption (10), $ceval(U) = A_{dec}(r_K, A_{enc}(r_L, m, r_R)) = \perp = ceval(eval(U))$.

Case 7.5: “ $eval(U_1) = K \in \mathbf{N}$ and $eval(U_2) = enc(K, t, R)$ ”.

Then $eval(U) = dec(K, enc(K, t, R)) = t$. Let $m := ceval(t)$. Then

$$\begin{aligned} ceval(U) &= A_{dec}(ceval(U_1), ceval(U_2)) \stackrel{(*)}{=} A_{dec}(ceval(K), ceval(enc(K, t, R))) \\ &= A_{dec}(r_K, A_{enc}(r_K, m, r_R)) \stackrel{(11)}{=} m = ceval(eval(U)). \end{aligned}$$

Here (*) uses the induction hypothesis.

This concludes the proof of Claim 3.

By assumptions (1) and (5), $ceval(M) \neq \perp$ if $M \in \mathbf{T}$. Furthermore, by construction, $eval(U) \in \mathbf{T}$ if $eval(U) \neq \perp$. Thus, $ceval(eval(U)) \neq \perp$ iff $eval(U) \neq \perp$. From Claim 3, we have that with overwhelming probability, $ceval(U) \neq \perp$ iff $ceval(eval(U)) \neq \perp$. Thus

with overwhelming probability, $ceval(U) \neq \perp$ iff $eval(U) \neq \perp$, i.e., U holds computationally iff U holds symbolically. \square

.noitulor