CryptoVerif Exercises

Bruno Blanchet
INRIA Paris
bruno.blanchet@inria.fr

December 2020

1 Cryptographic schemes

1.1 Exercise 1: encrypt-then-MAC is IND-CCA2

Definition 1 (IND-CCA2 symmetric encryption). *The advantage of the adversary against indistinguishability under adaptive chosen-ciphertext attacks (IND-CCA2) of a symmetric encryption scheme* SE *is:*

$$\mathsf{Succ}^{\mathsf{ind-cca2}}_{\mathsf{SE}}(t, q_e, q_d, l_e, l_d) = \max_{\mathcal{A}} 2\Pr\left[\begin{array}{l} b \overset{R}{\leftarrow} \{0,1\}; k \overset{R}{\leftarrow} kgen; \\ b' \leftarrow \mathcal{A}^{enc(LR(.,.,b),k), dec(.,k)} : b' = b \wedge \\ \mathcal{A} \text{ has not called } dec(.,k) \text{ on the result of} \\ enc(LR(.,.,b),k) \end{array}\right] - 1$$

where $\mathcal{A}$ *runs in time at most* t*, calls* $enc(LR(.,.,b),k)$ *at most* q_e *times on messages of length at most* l_e*, calls* $dec(.,k)$ *at most* q_d *times on messages of length at most* l_d*.*

Show using CryptoVerif that, if the MAC scheme is SUF-CMA and the encryption scheme is IND-CPA, then the encrypt-then-MAC scheme is IND-CCA2.

1.2 Exercise 2: A public-key encryption scheme

Definition 2 (IND-CPA public-key encryption). *A public-key encryption scheme* AE *consists of*

- *a key generation algorithm* $(pk, sk) \overset{R}{\leftarrow} kgen$
- *a probabilistic encryption algorithm* $enc(m, pk)$
- *a decryption algorithm* $dec(m, sk)$

such that $dec(enc(m, pk), sk) = m$.

The advantage of the adversary against indistinguishability under chosen-plaintext attacks (IND-CPA) is

$$\mathsf{Succ}^{\mathsf{ind-cca2}}_{\mathsf{AE}}(t) = \max_{\mathcal{A}} 2\Pr\left[\begin{array}{l} b \overset{R}{\leftarrow} \{0,1\}; (pk, sk) \overset{R}{\leftarrow} kgen; \\ (m_0, m_1, s) \leftarrow \mathcal{A}_1(pk); y \leftarrow enc(m_b, pk); \\ b' \leftarrow \mathcal{A}_2(m_0, m_1, s, y) : b' = b \end{array}\right] - 1$$

where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *runs in time at most* t*.*

Suppose that H is a hash function in the Random Oracle Model and that f is a one-way trapdoor permutation; we write f_{pk} for the permutation associated with the public key pk; its inverse is f_{sk}^{-1} where $(pk, sk) \overset{R}{\leftarrow} kgen$.

Consider the encryption function $E_{pk}(x) = f_{pk}(r) || H(r) \oplus x$, where $||$ denotes concatenation and $\oplus$ denotes exclusive or (Bellare & Rogaway, CCS'93).

- What is the decryption function?
- Show using CryptoVerif that this public-key encryption scheme is IND-CPA.

  Hint: the assumptions on cryptographic primitives can be defined using macros of the standard library of CryptoVerif: `ROM_hash` for the random oracle, `OW_trapdoor_perm` for the one-way trapdoor permutation, `Xor` for exclusive or. See Section 6 of the CryptoVerif manual for details on these macros.

1.3 Exercise 3: Full-Domain Hash signature scheme

Definition 3. *A signature scheme* S *consists of*

- *a key generation algorithm* $(pk, sk) \overset{R}{\leftarrow} kgen$
- *a signature algorithm* $sign(m, sk)$
- *a verification algorithm* $verify(m, pk, s)$

such that $verify(m, pk, sign(m, sk)) = 1$.

The advantage of the adversary against unforgeability under chosen message attacks (UF-CMA) of signatures is:

$$\mathsf{Succ}_{\mathsf{S}}^{\mathsf{uf-cma}}(t, q_s, l) =$$
$$\max_{\mathcal{A}} \Pr \left[\begin{array}{l} (pk, sk) \overset{R}{\leftarrow} kgen; (m, s) \leftarrow \mathcal{A}^{sign(.,sk)}(pk) : verify(m, pk, s) \wedge \\ m \text{ was never queried to the oracle } sign(., sk) \end{array} \right]$$

where $\mathcal{A}$ *runs in time at most* t,
calls $sign(., sk)$ *at most* q_s *times with messages of length at most* l.

Suppose that H is a hash function in the Random Oracle Model and that f is a one-way trapdoor permutation (as in the previous exercise).

We define a signature scheme as follows: $sign(m, sk) = f_{sk}^{-1}(H(m))$.

- What is the signature verification function?
- Show that this signature scheme is UF-CMA.

  Hint: the assumptions on cryptographic primitives can be defined using macros of the standard library of CryptoVerif: `ROM_hash` for the random oracle, `OW_trapdoor_perm` for the one-way trapdoor permutation. See Section 6 of the CryptoVerif manual for details on these macros.

2 Protocols

2.1 Exercise 4: Woo-Lam shared-key protocol

$\{M\}_k$ denotes the symmetric encryption of message M under the key k, using an authenticated encryption scheme (IND-CPA and INT-CTXT, macro `IND_CPA_INT_CTXT_sym_enc`; see Section 6 of the CryptoVerif manual for details on this macro).

Consider the fixed version of the Woo-Lam shared-key protocol, by Gordon and Jeffrey (CSFW'01):

$$\begin{array}{rl}
A \to B: & A \\
B \to A: & N \text{ (fresh random nonce)} \\
A \to B: & \{m3, B, N\}_{kAS} \\
B \to S: & A, B, \{m3, B, N\}_{kAS} \\
S \to B: & \{m5, A, N\}_{kBS}
\end{array}$$

At the end, B verifies that $\{m5, A, N\}_{kBS}$ is the message from S.

$m3$ and $m5$ are distinct constants. A and B are the names of the participants. kAS is a key shared between A and the server S, kBS is a key shared between B and the server S.

Show that, at the end of the protocol, A is authenticated to B.

Suggestion: one may consider

1. First, a simple version in which A talks only to B, B talks only to A, and S talks only to A and B.
2. Then, generalize to the case in which A, B, and S may also talk to dishonest participants.

2.2 Exercise 5: Needham-Schroeder public-key protocol

$\{M\}_{pk}$ denotes the encryption of message M under the public pk, using an IND-CCA2 public-key encryption scheme (macro `IND_CCA2_public_key_enc`; see Section 6 of the CryptoVerif manual for details on this macro).

- Consider the Needham-Schroeder public-key protocol, as fixed by Lowe. We first consider a simplified version without certificates:

  $$\begin{array}{rl}
  A \to B: & \{N_A, pk_A\}_{pk_B} \\
  B \to A: & \{N_A, N_B, pk_B\}_{pk_A} \\
  A \to B: & \{N_B\}_{pk_B}
  \end{array}$$

  Show that, at the end of the protocol, A and B are mutually authenticated.

  N_A and N_B are two random nonces, chosen respectively by A and B. pk_A and pk_B are the public keys of A and B, respectively.

  Note: the proof requires manual guidance (distinguish whether the key of interlocutor is pk_A, pk_B or some other key). The commands for manual guidance are presented in Section 7 of the CryptoVerif manual. The command to use for distinguishing cases is `insert` ⟨*program point*⟩ `"if` ⟨*condition*⟩ `then"`. Feel free to ask questions.

- Now consider the full version with certificates:

$$\begin{array}{ll}
A \to S\text{:} & (A, B) \\
S \to A\text{:} & (pk_B, B, \{pk_B, B\}_{sk_S}) \\
A \to B\text{:} & \{N_A, A\}_{pk_B} \\
B \to S\text{:} & (B, A) \\
S \to B\text{:} & (pk_A, A, \{pk_A, A\}_{sk_S}) \\
B \to A\text{:} & \{N_A, N_B, B\}_{pk_A} \\
A \to B\text{:} & \{N_B\}_{pk_B}
\end{array}$$

Show that, at the end of the protocol, A and B are mutually authenticated.

Note: the proof may require manual guidance (apply the security of signature under sk_S first). The commands for manual guidance are presented in Section 7 of the CryptoVerif manual. The command to use for applying a cryptographic assumption is `crypto`. Feel free to ask questions.