Cryptographic protocols: formal and computational proofs
Mid Term exam

December 1, 2020
Duration 3h. All documents are allowed

Solutions should be sent back as pdf files + possibly ProVerif files. All file names must be prefixed by your name.

Exercise 1

In this exercise, we use three primitives: a symmetric encryption scheme $\mathsf{senc}(\_,\_,\_)$, a public key encryption scheme $\mathsf{aenc}(\_,\_,\_)$ and a signature scheme $\mathsf{sign}(\_,\_)$.

This is modelled in the symbolic model using additional primitives $\mathsf{check}(\_,\_)$, $\mathsf{getm}(\_)$, $\mathsf{adec}(\_,\_)$, $\mathsf{sdec}(\_,\_)$, $\mathsf{pk}(\_)$, $\mathsf{vk}(\_)$, $\mathsf{sk}(\_)$, $\mathsf{dk}(\_)$, $\mathsf{Id}(\_)$, $\langle \_,\_ \rangle$ and the following rules.

$$\begin{array}{rcl}
\mathsf{sdec}(\mathsf{senc}(x,y,z),y) & = & x \\
\mathsf{adec}(\mathsf{aenc}(x,\mathsf{pk}(y),z),\mathsf{dk}(y)) & = & x \\
\mathsf{getm}(\mathsf{sign}(x,y)) & = & x \\
\mathsf{check}(\mathsf{sign}(x,\mathsf{sk}(y)),\mathsf{vk}(y)) & = & \mathsf{true} \\
\mathsf{getvk}(\mathsf{Id}(x)) & = & \mathsf{vk}(x) \\
\mathsf{getpk}(\mathsf{Id}(x)) & = & \mathsf{pk}(x) \\
\pi_1(\langle x,y \rangle) & = & x \\
\pi_2(\langle x,y \rangle) & = & y
\end{array}$$

Identities $\mathsf{Id}(x)$ are public: they are supposed to be known from the attacker. Note that public keys $\mathsf{pk}(x)$ and verification keys $\mathsf{vk}(x)$ are also known from the attacker since they can be obtained from $\mathsf{getpk}(\mathsf{Id}(x))$ and $\mathsf{getvk}(\mathsf{Id}(x))$ respectively.

Consider the two protocols described informally below,

Protocol 1:

$$\begin{array}{rl}
A \to B: & \langle \mathsf{Id}(r_A), \mathsf{sign}(\mathsf{aenc}(k,\mathsf{pk}(r_B),r),\mathsf{sk}(r_A)) \rangle \\
B \to A: & \mathsf{senc}(s,k,r')
\end{array}$$

Protocol 2:

$$\begin{array}{rl}
A \to B: & \langle \mathsf{Id}(r_A), \mathsf{aenc}(\mathsf{sign}(k,\mathsf{sk}(r_A)),\mathsf{pk}(r_B),r) \rangle \\
B \to A: & \mathsf{senc}(s,k,r')
\end{array}$$

1. Write formally for each of the two protocols, processes $P_A(r_A, id_B)$, $P_B(r_B, id_A)$ in which k, s are fresh random numbers. For the process P_B, we assume that B only accepts communication if the first projection of the input message is id_A. Either write the processes on paper or attach a ProVerif code.

2. Use ProVerif to check the secrecy of s in both protocols in the two following cases:

   (a) only two honest agents participate, i.e. $P_A(r_A, \mathsf{Id}(r_B))$ and $P_B(r_B, \mathsf{Id}(r_A))$

   (b) two honest agents and a corrupt agent participate, i.e. $P_A(r_A, \mathsf{Id}(r_C))$ and $P_B(r_B, \mathsf{Id}(r_A))$ where r_C is the seed of the corrupt agent.

   When ProVerif claims that the protocol is insecure, extract an attack trace: write (on paper) a formal trace using the operational semantics (Successive transition rules of conditional branching, evaluation, random generation and parallel composition can be written with a single instance of the transitive closure $\rightarrow^*$).

3. For protocols 1 and 2, define a process $P'_B(r_B)$ replacing $P_B(r_B, id_A)$ such that B now accepts communication from anyone. In which way does it modify the secrecy of s in the same scenarios of the previous question ? Explain informally why requesting only the secrecy of s is not adequate.

4. Modify the processes, including events that allow to check the agreement property on the key k. Use the process $P'_B(r_B)$ of the previous question (Either the processes on papers or attach a ProVerif code).

   Use ProVerif to check the agreement in both protocols.

5. For protocol 1, give a Horn clause translation $\mathcal{H}$ of $\mathtt{out}(\mathsf{Id}(r_B)); \mathtt{out}(r_C); !P'_B(r_B)$.

6. Show how the attacker clauses, together with $\mathcal{H}$, allow to deduce a copy of the secret s.

Exercise 2

We assume here that the encryption scheme is IND-CPA. $k_1, k_2, k_3, r_1, r_2, r_3, r_4$ are arbitrary distinct names. u, v are arbitrary terms, in which k_1, k_2, r_1, r_2, r_3 do not occur.
In the following cases, are the two frames statically equivalent ? Computationally indistinguishable (for any IND-CPA encryption scheme) ? Justify your answer.

1. $\phi_1 = \nu k_1, k_2, k_3, r_1, r_2.\ \{k_1\}_{k_2}^{r_1}, k_3, \{\langle k_1, k_2\rangle\}_{k_3}^{r_2},$
   $\phi_2 = \nu k_1, k_2, k_3, r_1, r_2.\ \{k_2\}_{k_1}^{r_1}, k_3, \{\langle k_1, k_2\rangle\}_{k_3}^{r_2}$

2. $\phi_1 = \nu k_1, k_2, k_3, r_1, r_2.\ \{\{k_1\}_{k_2}^{r_1}\}_{k_3}^{r_2}, k_3, \{\{k_2\}_{k_1}^{r_3}\}_{k_3}^{r_4},$
   $\phi_2 = \nu k_1, k_2, k_3, r_1, r_2.\ \{\{k_1\}_{k_2}^{r_1}\}_{k_3}^{r_2}, k_3, \{\{k_1\}_{k_2}^{r_3}\}_{k_3}^{r_4}$

3. $\phi_1 = \nu k_1, k_2, k_3, r_1, r_2, r_3.\ \{\{u\}_{k_1}^{r_1}\}_{k_1}^{r_2}, \{\{u\}_{k_2}^{r_3}\}_{k_1}^{r_1},$
   $\phi_2 = \nu k_1, k_2, k_3, r_1, r_2, r_3. \{\{u\}_{k_1}^{r_1}\}_{k_2}^{r_2}, \{u\}_{k_2}^{r_3}$

4. $\phi_1 = \nu k_1, k_2, k_3, r_1, r_2.\ \{\{u\}_{k_1}^{r_1}\}_{k_1}^{r_2}$ and $\phi_2 = \nu k_1, k_2, k_3, r_1, r_2.\ \{\{u\}_{k_2}^{r_1}\}_{k_1}^{r_2}$

Exercise 3

Assume the encryption scheme is IND-CPA. Are the two following processes computationally indistinguishable ? Justify (relying on the definition of computational indistinguishability).

$$P_1 = \nu k \nu r.\ \mathtt{in}(c, x).\ \mathsf{if}\ x = 0\ \mathsf{then}\ \mathtt{out}(c, \{1\}_k^r)\ \mathsf{else}\ \mathtt{out}(c, \{x\}_k^r)$$

$$P_2 = \nu k \nu r.\ \mathtt{in}(c, x).\mathtt{out}(c, \{x\}_k^r)$$

Solution

Exercise 1

1. - Protocol 1:

$$\begin{aligned}
P_A(r_a, id_B) = \quad & \nu k.\nu r. \\
& \mathtt{out}((\mathsf{Id}(r_A), \mathsf{sign}(\mathsf{aenc}(k, \mathsf{getpk}(id_B), r), \mathsf{sk}(r_A)))). \\
& \mathtt{in}(x). \\
& \mathtt{let}\ s = \mathsf{sdec}(x, k)\ \mathtt{in} \\
& 0
\end{aligned}$$

$$\begin{aligned}
P_B(r_B, id_A) = \quad & \mathtt{in}(z). \\
& \mathtt{let}\ x_{id} = \pi_1(z)\ \mathtt{in} \\
& \mathtt{let}\ x = \pi_2(z)\ \mathtt{in} \\
& \mathtt{if}\ x_{id} = id_A\ \mathtt{then} \\
& \mathtt{if}\ \mathsf{check}(x, \mathsf{getvk}(id_A))\ \mathtt{then} \\
& \mathtt{let}\ x_k = \mathsf{adec}(\mathsf{getm}(x), \mathsf{dk}(r_B))\ \mathtt{in} \\
& \nu r'.\nu s. \\
& \mathtt{out}(\mathsf{senc}(s, k, r')).0
\end{aligned}$$

   - Protocol 2:

$$\begin{aligned}
P_A(r_a, id_B) = \quad & \nu k.\nu r. \\
& \mathtt{out}((\mathsf{Id}(r_A), \mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(r_A)), \mathsf{getpk}(id_B), r))). \\
& \mathtt{in}(x). \\
& \mathtt{let}\ s = \mathsf{sdec}(x, k)\ \mathtt{in} \\
& 0
\end{aligned}$$

$$\begin{aligned}
P_B(r_B, id_A) = \quad & \mathtt{in}(z). \\
& \mathtt{let}\ x_{id} = \pi_1(z)\ \mathtt{in} \\
& \mathtt{let}\ x = \pi_2(z)\mathtt{in} \\
& \mathtt{if}\ x_{id} = id_A\ \mathtt{then} \\
& \mathtt{let}\ sig = \mathsf{adec}(x, \mathsf{dk}(r_B))\ \mathtt{in} \\
& \mathtt{if}\ \mathsf{check}(sig, \mathsf{getvk}(id_A))\ \mathtt{then} \\
& \mathtt{let}\ x_k = \mathsf{getm}(sig)\ \mathtt{in} \\
& \nu r'.\nu s. \\
& \mathtt{out}(\mathsf{senc}(s, k, r')).0
\end{aligned}$$

2. (a) Secrecy of s holds

   (b) Secrecy of s does not hold. We consider system

$$Sys = \mathtt{out}(\mathsf{Id}(r_A)).\mathtt{out}(\mathsf{Id}(r_B)).(P_A(r_A, \mathsf{Id}(r_C)) \mid P_B(r_B, \mathsf{Id}(r_A)))$$

The attack trace is as follows:

$$
\begin{array}{cl}
 & (\nu\{r_A, r_B\}.\emptyset, \emptyset, Sys) \\
\to & (\nu\{r_A, r_B\}.\mathsf{Id}(r_A), \emptyset, \mathsf{out}(\mathsf{Id}(r_B)).(P_A(r_A, \mathsf{Id}(r_C)) \mid P_B(r_B, \mathsf{Id}(r_A)))) \\
\to & (\nu\{r_A, r_B\}.\mathsf{Id}(r_A) \cdot \mathsf{Id}(r_B), \emptyset, P_A(r_A, \mathsf{Id}(r_C)) \mid P_B(r_B, \mathsf{Id}(r_A))) \\
\to^* & (\nu\{r_A, r_B, k, r\}.\mathsf{Id}(r_A) \cdot \mathsf{Id}(r_B), \emptyset, P_A^1 \mid P_B(r_B, \mathsf{Id}(r_A))) \\
\to & (\nu\{r_A, r_B, k, r\}.\mathsf{Id}(r_A) \cdot \mathsf{Id}(r_B) \cdot (\mathsf{Id}(r_A), M_1), \emptyset, P_A^2 \mid P_B(r_B, \mathsf{Id}(r_A))) \\
\xrightarrow{u} & (\nu\{r_A, r_B, k, r\}.\mathsf{Id}(r_A) \cdot \mathsf{Id}(r_B) \cdot (\mathsf{Id}(r_A), M_1), \sigma, P_A^2 \mid P_B^1 \\
\to^* & (\nu\{r_A, r_B, k, r, s, r'\}.\mathsf{Id}(r_A) \cdot \mathsf{Id}(r_B) \cdot (\mathsf{Id}(r_A), M_1), \sigma', P_A^2 \mid P_B^2 \\
\to & (\nu\{r_A, r_B, k, r, s, r'\}.\mathsf{Id}(r_A) \cdot \mathsf{Id}(r_B) \cdot (\mathsf{Id}(r_A), M_1) \cdot \mathsf{senc}(s, k, r'), \sigma', P_A^2 \mid 0)
\end{array}
$$

with :

- $M_1 = \mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(r_A)), \mathsf{pk}(r_C), r)$
- $u = (\mathsf{Id}(r_A), \mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(r_A)), \mathsf{pk}(r_B), r''))$
- $\sigma = \{z \to u\}$
- $\sigma' = \sigma \cup \{x_{id} \to \mathsf{Id}(r_A), x \to \mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(r_A)), \mathsf{pk}(r_B), r''), sig \to \mathsf{sign}(k, \mathsf{sk}(r_A)), x_k \to k\}$
-

$$
\begin{array}{rl}
P_A^1 = & \mathsf{out}((\mathsf{Id}(r_A), \mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(r_A)), \mathsf{getpk}(id_B), r))). \\
 & \mathtt{in}(x). \\
 & \mathtt{let}\ s = \mathsf{sdec}(x, k)\ \mathtt{in} \\
 & 0
\end{array}
$$

$$
\begin{array}{rl}
P_A^2 = & \mathtt{in}(x). \\
 & \mathtt{let}\ s = \mathsf{sdec}(x, k)\ \mathtt{in} \\
 & 0
\end{array}
$$

$$
\begin{array}{rl}
P_B^1 = & \mathtt{let}\ x_{id} = \pi_1(z)\ \mathtt{in} \\
 & \mathtt{let}\ x = \pi_2(z)\mathtt{in} \\
 & \mathtt{if}\ x_{id} = id_A\ \mathtt{then} \\
 & \mathtt{let}\ sig = \mathsf{adec}(x, \mathsf{dk}(r_B))\ \mathtt{in} \\
 & \mathtt{if}\ \mathsf{check}(sig, \mathsf{getvk}(id_A))\ \mathtt{then} \\
 & \mathtt{let}\ k = \mathsf{getm}(sig)\ \mathtt{in} \\
 & \nu r'.\nu s. \\
 & \mathsf{out}(\mathsf{senc}(s, k, r')).0
\end{array}
$$

$$
P_B^2 = \mathsf{out}(\mathsf{senc}(s, x_k, r')).0
$$

3. For Protocol 1, $P_B(r_B)$ is defined as follows:

$$
\begin{array}{rl}
P_B'(r_B) = & \mathtt{in}(z). \\
 & \mathtt{let}\ x_{id} = \pi_1(z)\ \mathtt{in} \\
 & \mathtt{let}\ x = \pi_2(z)\ \mathtt{in} \\
 & \mathtt{if}\ \mathsf{check}(x, \mathsf{getvk}(x_{id}))\ \mathtt{then} \\
 & \mathtt{let}\ x_k = \mathsf{adec}(\mathsf{getm}(x), \mathsf{dk}(r_B))\ \mathtt{in} \\
 & \nu r'.\nu s. \\
 & \mathsf{out}(\mathsf{senc}(s, k, r')).0
\end{array}
$$

For Protocol 2, $P_B(r_B)$ is defined as follows:

$$\begin{array}{rl} P'_B(r_B) = & \mathtt{in}(z). \\ & \mathtt{let}\ x_{id} = \pi_1(z)\ \mathtt{in} \\ & \mathtt{let}\ x = \pi_2(z)\mathtt{in} \\ & \mathtt{let}\ sig = \mathsf{adec}(x, \mathsf{dk}(r_B))\ \mathtt{in} \\ & \mathtt{if}\ \mathsf{check}(sig, \mathsf{getvk}(x_{id}))\ \mathtt{then} \\ & \mathtt{let}\ x_k = \mathsf{getm}(sig)\ \mathtt{in} \\ & \nu r'.\nu s. \\ & \mathtt{out}(\mathsf{senc}(s, k, r')).0 \end{array}$$

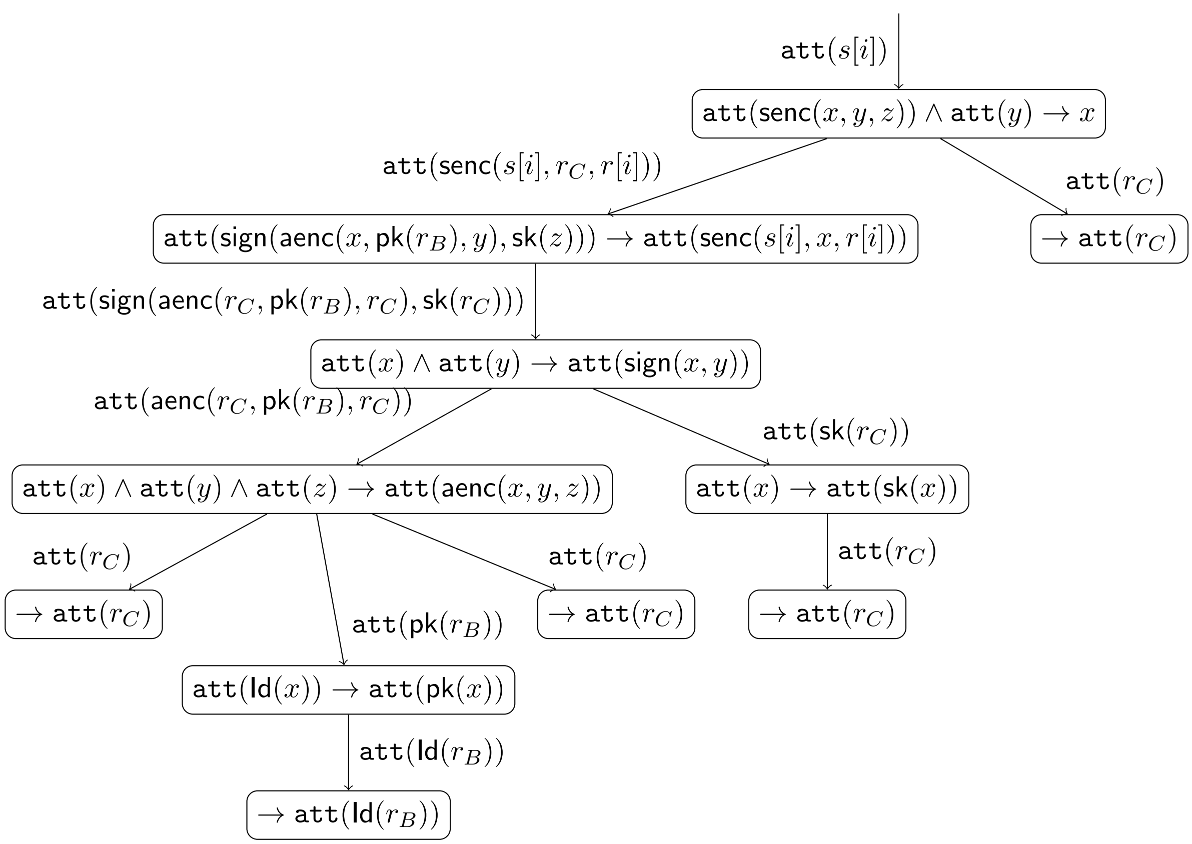
Since B now accepts communication from anyone, it also accept a communication directly from the attacker, i.e. an identity $\mathsf{Id}(r_C)$ where r_C is known to the attacker. In such a case, s will not be secret anymore. We would need to prove the secrecy of s only when B communicates with honest participants.

4. See files *ex1_q4_protocol_1.pv* and *ex1_q4_protocol_2.pv*

5.

$$\begin{array}{l} \rightarrow \mathtt{att}(\mathsf{Id}(r_B)) \\ \rightarrow \mathtt{att}(r_C) \\ \mathtt{att}(\mathsf{sign}(\mathsf{aenc}(x, \mathsf{pk}(r_B), y), \mathsf{sk}(z))) \rightarrow \mathtt{att}(\mathsf{senc}(s[i], x, r[i])) \end{array}$$

6.



Exercise 2

1. The two sequences are not statically equivalent: if we choose $R_1 = \mathsf{dec}(x_1, \pi_2(\mathsf{dec}(x_3, x_2)))$ and $R_2 = \pi_1(\mathsf{dec}(x_3, x_2))$, $R_1\phi_1 \downarrow = R_2\phi_1 \downarrow$, while $R_1\phi_2 \downarrow \neq R_2\phi_2 \downarrow$.

   They are not computationally indistinguishable as the PPT implementing respectively R_1 and R_2 will distinguish them.

2. The two frames are not statically equivalent since $\mathsf{EK}(\mathsf{dec}(x_1, x_2), \mathsf{dec}(x_3, x_2))\phi_1 \downarrow \neq \mathtt{true}$ while $\mathsf{EK}(\mathsf{dec}(x_1, x_2), \mathsf{dec}(x_3, x_2))\phi_2 \downarrow = \mathtt{true}$. Therefore, they are neither computationally indistinguishable, at least for IND-CPA encryption schemes that reveal part of the encryption keys (e.g., the first bit)

3. The two frames are not computationally equivalent for some IND-CPA encryption schemes: if the encryption of m has a length strictly larger than the length of m and the encryption reveals the length of the plaintexts (which never impairs IND-CPA), the adversary can observe bitstrings of the same length in ϕ_1 and bitstrings of different lengths in ϕ_2.

   Though EL has not been fully defined in the class, it should also distinguish the two frames (otherwise the computational soundness theorem would fail).

4. $\phi_1 \sim \phi_2$. Indeed, for any recipe R, $R\phi_i \downarrow = R\phi_i$ or $R\phi_i \downarrow$ is independent of ϕ_i. It follows that $Eq(\phi_1) = Eq(\phi_2)$.

   Now they are computationally indistinguishable, but we cannot use directly the computational soundness theorem as k_1 occurs twice in the first term.

   Assume we have an attacker $\mathcal{A}$ on the indistinguishability of the two frames:

   $$\epsilon = \mathbf{Prob}\{k_1, k_2, r_1, r_2, \rho : \mathcal{A}([\![\{\{u\}_{k_1}^{r_1}\}_{k_1}^{r_2}]\!]) = 1\} - \mathbf{Prob}\{k_1, k_2, r_3, r_2, \rho : \mathcal{A}([\![\{\{u\}_{k_2}^{r_3}\}_{k_1}^{r_2}]\!]) = 1\}$$

   is non negligible (for convenience, we renamed r_1 in r_3 in the second frame).

   Let us construct an attacker $\mathcal{B}$ on IND-CPA.

   (a) $\mathcal{B}$ first computes $[\![u]\!]$ (which is possible since k_1, k_2, r_1, r_2 do not occur in u.

   (b) $\mathcal{B}$ submits $([\![u]\!], [\![u]\!])$ to the k_1-encryption oracle (and gets back $[\![\{u\}_{k_1}^{r_1}]\!]$)

   (c) $\mathcal{B}$ draws a key k_2 a random seed r_3 and computes $[\![\{u\}_{k_2}^{r_3}]\!]$

   (d) $\mathcal{B}$ submits $([\![\{u\}_{k_1}^{r_1}]\!], [\![\{u\}_{k_2}^{r_3}]\!])$ to the encryption oracle and gets either $m_1 = [\![\{\{u\}_{k_1}^{r_1}\}_{k_1}^{r_2}]\!]$ or $m_2 = [\![\{\{u\}_{k_2}^{r_3}\}_{k_1}^{r_2}]\!]$.

   (e) $\mathcal{B}$ calls $\mathcal{A}$ on m_i and returns the same result as $\mathcal{A}$

   $\mathcal{B}$ wins the IND-CPA game with an advantage ϵ.

Exercise 3

They are indeed computationally indistinguishable: consider three PPT $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$.

$\mathcal{A}_1$ computes an input for either of the two processes.

If this input is not 0, then the traces will consist in identical messages for P_1 and P_2 (which, of course cannot be distonguished by $\mathcal{A}_2, \mathcal{A}_3$).

So, the only relevant trace is when $\mathcal{A}_1$ sends 0, in which case we get the two single element traces $\{1\}_k^r$ and $\{0\}_k^r$. Hence the computational indistinguishability of the two processes is equivalent to the computational indistinguishability of the two frames $\nu k \nu r.\ \{1\}_k^r$ and $\nu k \nu r.\ \{0\}_k^r$, which holds true when the encryption scheme is IND-CPA.