Examination of the module MPRI 2-30
Cryptographic protocols: formal and computational proofs

(A single two-sided document is allowed; electronic devices are forbidden; duration: 3h)

February 27, 2024

Please use different sheets for the two parts of the exam.

Part A (1 h 30, 1/2 the mark)

Each question comes with the number of lines used to answer it in the solutions (which is concise). This number is here to give a rough estimate of the level of details expected: your answers may be longer or shorter. This does not indicate a question difficulty.

1 Key Encapsulation Mechanism

A *Key Encapsulation Mechanism* (KEM) is a tuple of functions $(\mathsf{pk}(\cdot), \mathsf{sk}(\cdot), \mathsf{encap}(\cdot,\cdot,\cdot), \mathsf{decap}(\cdot,\cdot))$ such that:

- $\mathsf{pk}(\mathsf{n})$ and $\mathsf{sk}(\mathsf{n})$ are, resp., the *public* and *private* keys;
- $\mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}), \mathsf{r})$ returns an *encapsulation* c of an *output key* k using *randomness* r;
- if c is an encapsulation, then $\mathsf{decap}(c, \mathsf{sk}(\mathsf{n}))$ *decapsulate* c into the output key k.

A KEM must satisfy the following relation:

$$\forall n, k, r.\, \mathsf{decap}(\mathsf{encap}(k, \mathsf{pk}(n)), \mathsf{sk}(n), r) = k$$

A KEM is said to be IND-CPA$_{\text{KEM}}$ if no adversary can distinguish between the output key k and a fresh randomly sampled key k^*, even if it knows the encapsulation of k. I.e., for any PTIME adversary $\mathcal{A}$, it must be the case that:

$$\left| \begin{array}{l} \Pr_{\mathsf{n},\mathsf{k},\mathsf{r}} (\mathcal{A}(\mathsf{pk}(\mathsf{n}), c, \mathsf{k}) = 1 \text{ where } c = \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}), \mathsf{r})) \\ - \Pr_{\mathsf{n},\mathsf{k},\mathsf{k}^*,\mathsf{r}} (\mathcal{A}(\mathsf{pk}(\mathsf{n}), c, \mathsf{k}^*) = 1 \text{ where } c = \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}), \mathsf{r})) \end{array} \right|$$

is a negligible function of η, where $\mathsf{n}, \mathsf{k}, \mathsf{k}^*$ are sampled uniformly in $\{0,1\}^\eta$.

Question 1 (3 line). *What is the difference between a KEM and an Public Key Encryption (PKE) scheme?*

Solution. A PKE can safely encrypt arbitrary values and provides data confidentiality.

A KEM can only encrypt randomly generated keys safely, by guaranteeing that this key is indistinguishable from a fresh random value. ■

Question 2 (4 lines)**.** *Give sufficient syntactic conditions (as general as possible) under which the following indistinguishability formula:*

$$\vec{u}, \mathit{encap}(k, \mathit{pk}(n), r), k \sim \vec{u}, \mathit{encap}(k, \mathit{pk}(n), r), k^*$$

is valid in any model in which the KEM is IND-CPA$_{KEM}$.

Solution. We must have that:

- $\vec{u}$ is a ground term and $\mathsf{n}, \mathsf{k}, \mathsf{k}^*, \mathsf{r}$ are names;
- r, k and k^* do not occur anywhere in $\vec{u}$, and n only occurs in $\vec{u}$ is sub-terms of the form $\mathsf{pk}(\mathsf{n})$.

■

2 A KEM-Based Messaging Protocol

We consider a *symmetric* key encryption scheme $(\mathsf{senc}(\cdot,\cdot,\cdot), \mathsf{sdec}(\cdot,\cdot))$ that verifies:

$$\forall m, k, r.\ \mathsf{sdec}(\mathsf{senc}(m, k, r), k) = m$$

We assume that the symmetric encryption is IND-CPA. We provide in Figure 1 a rule schema which is sound under this assumption.

In this section, we also assume that the KEM is IND-CPA$_{\text{KEM}}$.

The Protocol We consider a simple one-way messaging protocol between a *sender* $\mathcal{S}$ and a *receiver* $\mathcal{R}$. The receiver $\mathcal{R}$ possesses a public/private KEM key pair $(\mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{sk}(\mathsf{n}_\mathcal{R}))$, and we assume that the sender $\mathcal{S}$ knows the KEM public key $\mathsf{pk}(\mathsf{n}_\mathcal{R})$. To send a message m (which we model has a constant value), the sender $\mathcal{S}$ samples an output key k, encapsulate it under $\mathsf{pk}(\mathsf{n}_\mathcal{R})$ by computing $e \stackrel{\text{def}}{=} \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0)$, and sends $\langle e, \mathsf{senc}(m, \mathsf{k}, \mathsf{r})\rangle$ to $\mathcal{R}$. We model this process as follows:

$$\mathcal{S} := \nu\,\mathsf{k};\ \nu\,\mathsf{r}_0;\ \nu\,\mathsf{r};\ \mathbf{out}\big(\mathsf{c}_\mathcal{R}, \langle \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(m, \mathsf{k}, \mathsf{r})\rangle\big)$$

where $\langle\cdot,\cdot\rangle$ is the pair function, and we will use π_1 and π_2 as, resp., first and second projections:

$$\pi_1(\langle x, y\rangle) = x \quad \text{and} \quad \pi_2(\langle x, y\rangle) = y \qquad \text{(for all } x, y)$$

Question 3 (2 lines)**.** *Describe how the receiver decrypts the message it received from $\mathcal{S}$ to retrieve m.*

Solution. On input x, it decapsulate the key by computing $k' = \mathsf{decap}(\pi_1(x), \mathsf{sk}(\mathsf{n}_\mathcal{R}))$, and decrypts the message by computing $m' = \mathsf{sdec}(\pi_2(x), k')$. ■

We consider the following idealized process $\mathcal{S}_I$:

$$\mathcal{S}_I := \nu\,\mathsf{k};\ \nu\,\mathsf{r};\ \nu\,\mathsf{r}_0;\ \mathbf{out}\big(\mathsf{c}_\mathcal{R}, \langle \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(0^{|m|}, \mathsf{k}, \mathsf{r})\rangle\big)$$

Question 4 (18 lines)**.** *Prove that $\nu\,\mathsf{n}_\mathcal{R};\ \mathcal{S} \approx \nu\,\mathsf{n}_\mathcal{R};\ \mathcal{S}_I$ using the logic from the lecture.*

Solution. The processes $\mathcal{S}$ and $\mathcal{S}_I$ only has one folding for action trace $\mathbf{out}(\mathsf{c}_\mathcal{R})$ which yield the equivalence:

$$\langle \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(m, \mathsf{k}, \mathsf{r})\rangle \sim \langle \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(0^{|m|}, \mathsf{k}, \mathsf{r})\rangle$$

First, we break it using FA:

$$\mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(m, \mathsf{k}, \mathsf{r}) \sim \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(0^{|m|}, \mathsf{k}, \mathsf{r}) \qquad (1)$$

Then, we prove two intermediate results:

- We replace the left output key by a fresh key $\mathsf{k}^* \in \mathcal{N}$, i.e. we prove:

$$\mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(m, \mathsf{k}, \mathsf{r}) \sim \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(m, \mathsf{k}^*, \mathsf{r})$$

  First, we apply FA to remove senc and m, and Fresh to remove r:

$$\mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{k} \sim \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{k}^*$$

  Then, we use IND-CPA$_{\text{KEM}}$ for the KEM, which immediately concludes (syntactic conditions trivially holds here).

- The same proof steps allow to replace the right output key by a fresh key, i.e. to show:

$$\mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(0^{|m|}, \mathsf{k}, \mathsf{r}) \sim \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(0^{|m|}, \mathsf{k}^*, \mathsf{r})$$

Coming back to (1) and using transitivity, we obtain:

$$\mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(m, \mathsf{k}^*, \mathsf{r}) \sim \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(0^{|m|}, \mathsf{k}^*, \mathsf{r})$$

Then, we use IND-CPA for symmetric encryption (since syntactic conditions holds for k^*, as it is fresh), which immediately concludes. ■

We now consider an extended process $\mathcal{R}'$ in which the receiver sends a response m' to $\mathcal{S}$. Roughly, after retrieving the output key k and the message m from its input, $\mathcal{R}'$ sends the encryption of m' under key k:

$$\mathcal{R}' := \mathbf{in}(\mathsf{c}_\mathcal{R}, x);\ [\ldots]\textit{(retrieve k and m)}\ ;\ \nu\, \mathsf{r}';\ \mathbf{out}\big(\mathsf{c}_\mathcal{S}, \mathsf{senc}(m', \mathsf{k}, \mathsf{r}')\rangle\big)$$

Question 5 (1 line). *Write an idealized version $\mathcal{R}'_I$ of $\mathcal{R}'$, in the spirit of what we did with $\mathcal{S}_I$.*

Solution.

$$\mathcal{R}'_I := \mathbf{in}(\mathsf{c}_\mathcal{R}, x);\ k' := \mathsf{decap}(\pi_1(x), \mathsf{sk}(\mathsf{n}_\mathcal{R}));\ \nu\, \mathsf{r}';\ \mathbf{out}\big(\mathsf{c}_\mathcal{S}, \mathsf{senc}(0^{|m'|}, k', \mathsf{r}')\rangle\big) \quad ■$$

Question 6 (7 lines). *Does the equivalence $\nu\, \mathsf{n}_\mathcal{R};\ (\mathcal{S} \mid \mathcal{R}') \approx \nu\, \mathsf{n}_\mathcal{R};\ (\mathcal{S}_I \mid \mathcal{R}'_I)$ holds? If yes, quickly explain how the proof of question 4 should be adapted. If not, quickly describe an attack.*

Solution. The equivalence does not hold. Indeed, the adversary can simply generate its own output key $\mathsf{k}_\mathcal{A}$, and use it to send the encryption of some arbitrary message (say 1) to the receiver:

$$\langle \mathsf{encap}(\mathsf{k}_\mathcal{A}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}_0), \mathsf{senc}(1, \mathsf{k}_\mathcal{A}, \mathsf{r})\rangle$$

Then, the receiver will answer using $\mathsf{k}_\mathcal{A}$, which the adversary knows. Thus, it can decrypt the final message and check whether it obtains m' or 0, which allow it to distinguish between the left and right scenario with probability 1. Adding some form of signature would solve this. ■

Question 7 (5 lines). *Propose a modification $\mathcal{S}'$ of the process $\mathcal{S}$ that efficiently sends many messages $m_1, \ldots, m_N$ instead of just one. Each output can only send one message m_i.*

Solution. A trivial solution is to add a replication in front of $\mathcal{S}$, and to do:

$$\mathcal{S}' := !_{i \le N}\ \big(\nu\, \mathsf{k}_i;\ \nu\, \mathsf{r}'_i;\ \nu\, \mathsf{r}_i;\ \mathbf{out}(\mathsf{c}^i_\mathcal{R}, \langle \mathsf{encap}(\mathsf{k}_i, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r}'_i), \mathsf{senc}(m_i, \mathsf{k}_i, \mathsf{r}_i)\rangle)\big)$$

But this is very in-efficient, as asymmetric cryptography is much slower than symmetric cryptography. It is better to re-use the output k for each sub-sequent encryption, e.g. as follows:

$$\mathcal{S}' := \nu\, \mathsf{k};\ \nu\, \mathsf{r};\ \mathbf{out}(\mathsf{c}_\mathcal{R}, \mathsf{encap}(\mathsf{k}, \mathsf{pk}(\mathsf{n}_\mathcal{R}), \mathsf{r})\rangle);\ !_{i \le N}\ \big(\nu\, \mathsf{r}_i;\ \mathbf{out}(\mathsf{c}^i_\mathcal{R}, \mathsf{senc}(m_i, \mathsf{k}, \mathsf{r}_i)\big) \quad ■$$

3 Robustness of a PKE

(Do not confuse the notations for PKE in this section with the notation of the previous section.)

We consider a public key encryption scheme $(\mathsf{pk}(\cdot), \mathsf{sk}(\cdot), \{\cdot\}^{\cdot}_{\cdot}, \mathsf{dec}(\cdot,\cdot))$ that verifies:

$$\forall m, n, r.\ \mathsf{dec}(\{m\}^r_{\mathsf{pk}(n)}, \mathsf{sk}(n)) = m$$

The PKE is said to be robust if no efficient adversary can produce a message who successfully decrypts for two public/private key pairs $(\mathsf{pk}(\mathsf{n}_1), \mathsf{sk}(\mathsf{n}_1))$ and $(\mathsf{pk}(\mathsf{n}_2), \mathsf{sk}(\mathsf{n}_2))$ — where $\mathsf{n}_1 \neq \mathsf{n}_2$. To that end, we assume the existence of a special symbol $\perp$ used to denote a failed decryption (we require that $\perp$ is different from any message m that may be encrypted), and we define two robustness notions:

- The PKE verifies the *robustness-V1* property iff. for any PTIME adversary $\mathcal{A}$, the quantity:

$$\Pr_{\mathsf{n}_1,\mathsf{n}_2}\left(\mathsf{dec}(c, \mathsf{sk}(\mathsf{n}_1)) \neq \perp \text{ and } \mathsf{dec}(c, \mathsf{sk}(\mathsf{n}_2)) \neq \perp \text{ where } c := \mathcal{A}(1^\eta, \mathsf{pk}(\mathsf{n}_1), \mathsf{pk}(\mathsf{n}_2))\right)$$

  is negligible in η.

- The PKE verifies the *robustness-V2* property iff. for any PTIME adversary $\mathcal{A}$, the quantity:

$$\Pr_{\mathsf{n}_1,\mathsf{n}_2,\mathsf{r}}\left(\mathsf{dec}(\{m\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n}_1)}, \mathsf{sk}(\mathsf{n}_2)) \neq \perp \text{ where } m := \mathcal{A}(1^\eta, \mathsf{pk}(\mathsf{n}_1), \mathsf{pk}(\mathsf{n}_2))\right)$$

  is negligible in η. *(Note that m is encrypted under the first key pair but decrypted using the second key pair).*

Question 8 (10 lines). *What is the relation between robustness-V1 and robustness-V2?*

Solution. Robustness-V1 implies Robustness-V2. Indeed, consider an adversary $\mathcal{A}$ against Robustness-V2. Let $\mathcal{B}$ the adversary against Robustness-V1:

$$\begin{aligned}\mathcal{B}(1^\eta, \mathsf{pk}(\mathsf{n}_1), \mathsf{pk}(\mathsf{n}_2)) \stackrel{\text{def}}{=} \quad & m := \mathcal{A}(1^\eta, \mathsf{pk}(\mathsf{n}_1), \mathsf{pk}(\mathsf{n}_2));\\ & r \to \$;\ c := \{m\}^r_{\mathsf{pk}(\mathsf{n}_1)}\\ & \textbf{return } c\end{aligned}$$

By soundness of the PKE, c decrypts under $\mathsf{pk}(\mathsf{n}_1)$ to m. Thus, $\mathsf{dec}(m, \mathsf{sk}(\mathsf{n}_1)) \neq \perp$ (we required that $m \neq \perp$). Hence the probability that $\mathcal{B}$ wins Robustness-V1 is exactly the probability that $\mathcal{A}$ wins Robustness-V2. Assuming that the PKE verifies Robustness-V1, the former quantity is negligible. Thus so is the latter. ■

Question 9 (5 lines). *Design a rule schema of the logic that is valid in any model where the public key encryption scheme satisfies the robustness-V1 property. Do the same for robustness-V2.*

Solution. For any names $\mathsf{n}_1, \mathsf{n}_2$ and ground term t such that that n_1 (resp. n_2) only occurs in t in subterms of the form $\mathsf{pk}(\mathsf{n}_1)$ (resp. $\mathsf{pk}(\mathsf{n}_2)$):

$$\left[\mathsf{dec}(t, \mathsf{sk}(\mathsf{n}_1)) = \perp \,\dot{\vee}\, \mathsf{dec}(t, \mathsf{sk}(\mathsf{n}_2)) = \perp\right] \qquad (\textbf{Robustness-V1})$$

$$\left[\mathsf{dec}(\{t\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n}_1)}, \mathsf{sk}(\mathsf{n}_2)) = \perp\right] \qquad (\textbf{Robustness-V2})$$

where r is a name that does not occur in t. ■

If $(\mathsf{senc}(\cdot,\cdot,\cdot), \mathsf{sdec}(\cdot,\cdot))$ is an IND-CPA scheme, then the *ground* rule:

$$\frac{\mathsf{len}(m_0) \doteq \mathsf{len}(m_1) \sim \mathsf{true}}{\vec{u}, \mathsf{senc}(m_0, \mathsf{r}, \mathsf{k}) \sim \vec{u}, \mathsf{senc}(m_1, \mathsf{r}, \mathsf{k})} \text{ IND-CPA}$$

is sound, when $\mathsf{k}, \mathsf{r} \in \mathcal{N}$ are names that **do not** appears in $\vec{u}, m_0, m_1$.

Figure 1: Rule for symmetric encryption.