Cryptographic protocols: formal and computational proofs
First part: symbolic verification and computational soundness

November 30. Duration 3h.
All documents are allowed. Electronic devices are forbidden.

The length of a solution is indicated for each question. There might be valid solutions that are longer (or shorter). The bonus questions are not evaluated in this exam.

In what follows we consider asymmetric encryption and pairing, together with the rewrite rules (resp. the inference rules) that have been seen during the lectures.

We consider the following variant of the Needham-Schroeder-Lowe protocol, which is informally described by:

\[
\begin{align*}
A &\to B : \text{aenc}(\langle A, N_A \rangle, B) \\
B &\to A : \text{aenc}(\langle N_A, \langle N_B, B \rangle \rangle, A) \\
A &\to B : \text{aenc}(N_B, B)
\end{align*}
\]

Formally, we consider the following process for the role \(B\), which is parametrized by \(a\) and \(b\):

\[
P_B(b, a) = \nu n_B. \text{in}(x). \quad \text{let } x_1 = \text{adec}(x, \text{sk}(b)) \text{ in} \\
\text{let } x_2 = \pi_1(x_1) \text{ in} \quad \text{let } x_3 = \pi_2(x_1) \text{ in} \\
\text{if } x_2 = a \text{ then } \text{out}(\text{aenc}(\langle x_3, \langle n_B, b \rangle \rangle, a)) \cdot 0
\]

Equivalently in a small constructor-based calculus, the process would be written

\[
P_B(b, a) = \nu n_B. \text{in}(\text{aenc}(\langle a, x_3 \rangle, b)). \text{out}(\text{aenc}(\langle x_3, \langle n_B, b \rangle \rangle, a)) \cdot 0
\]

1. [5 lines] Propose similar formalizations for the role \(A\), parametrized by \(a\) and \(b\)

2. Consider the scenario \(P = \text{out}(a).\text{out}(b).\text{out}(c).\text{out}(\text{sk}(c)).0 \parallel P_B(b, a) \parallel P_B(a, c)\). We consider executions of \(P\), in which the output actions are executed first.

   (a) [6 lines] Give the two deducibility constraint systems \(C_1, C_2\), corresponding respectively to the case where \(P_B(b, a)\) moves first (until the end) and then \(P_B(a, c)\) moves until the end (this is \(C_1\)) and to the case where \(P_B(a, c)\) moves first and then \(P_B(b, a)\) (This is \(C_2\)).

   (b) [26 lines] Solve the deducibility constraint system \(C_1\), using the simplification rules of the lectures. (You may notice that the rules can always be applied to the first unsolved deducibility constraint, according to the completeness proof. This avoids unnecessary branching).
(c) [14 lines] Find all attacks on the weak secrecy of the nonce $n_B$, generated in the process $P_B(b,a)$ in the scenario $P$, when the input action of $P_B(b,a)$ is executed before the input action of $P_B(a,c)$.

(d) [2 lines] What can we conclude on the protocol?

3. We consider again the process $P$.

(a) [6 lines] Compute the Horn clauses associated with $P$

(b) [18 lines] Show how the saturation process finds the attack of the question 2c

4. In order to fix the previous problems, we propose to introduce a length test. Formally, we introduce a new function symbol $L$ whose interpretation $\ell$ is given by:

$$\ell(a) = 1 \text{ if } a \text{ is a name } \quad \ell(\langle u, v \rangle) = \ell(u) + \ell(v) + 1 \quad \ell(\text{aenc}(u,v)) = \ell(u) + \ell(v)$$

We also assume that the attacker has at least one name (for instance, any scenario first outputs a name), so that he can construct messages of arbitrary positive length.

Now, each time a process receives a message, it checks that it has the expected length. More precisely, we consider a small process algebra defined as follows:

- **Simple processes** are given by the grammar:
  
  $$\text{SP} :: \text{0} \mid \text{in}(\text{CTerm}).\text{SP} \mid \text{out}(\text{CTerm}).\text{SP} \mid \text{if Cond then SP else SP}$$

And the first occurrence of a variable in a simple process is always in an input message.

- **CTerm** is defined (as usual) as the set of terms constructed using pairing, encryption, names and variables (no symbol $L$).

- **Cond** is a Boolean combination of atomic formulas of the form $L(u_i) = n_i$ where $u_i$ is a CTerm and $n_i$ is a positive integer. If $u_i$ is a message (a ground CTerm), the atomic condition $L(u_i) = n_i$ is valid if $\ell(u_i) = n_i$. This is extended to Boolean combinations of ground atomic conditions.

Processes are defined as $(\nu n_1, \ldots, \nu n_k).P_1 \| \cdots \| P_m$ where $P_1, \ldots, P_m$ are simple processes.

The operational semantics is defined as expected (|| is associative and commutative):

$$((\nu \pi)\text{in}(u) \cdot P)\|Q, M) \xrightarrow{\text{in}(u\sigma)} ((\nu \pi)P\sigma\|Q, M) \quad \text{If } (\nu \pi)M \vdash u\sigma \text{ and } u\sigma \text{ is a message}$$

$$((\nu \pi)\text{out}(u) \cdot P)\|Q, M) \xrightarrow{\text{out}(u)} ((\nu \pi)P\|Q, M \cup \{u\})$$

$$(\nu \pi)\text{if } C \text{ then } P \text{ else } Q\|R, M) \rightarrow ((\nu \pi)P\|R, M) \quad \text{If } C \text{ is valid}$$

$$(\nu \pi)\text{if } C \text{ then } P \text{ else } Q\|R, M) \rightarrow ((\nu \pi)Q\|R, M) \quad \text{If } \neg C \text{ is valid}$$

(a) [3 lines] Propose a modification $P_B'(b,a)$ of the process $P_B(b,a)$ in this new process calculus, in which the expected length of the input messages are checked.
(b) [34 lines] Propose an extension of the deducibility constraints and a symbolic operational semantics \([\cdot]\) of the above process calculus, that maps every process \(P\) to a finite set of pairs \((t_S, D)\) where \(t_S\) is a symbolic trace and \(D\) is a deducibility constraint in such a way that 

\[
t \quad \text{is a trace of } P \quad \text{iff} \quad \exists (t_S, D) \in [P] \quad \text{and a substitution } \sigma \text{ such that } \sigma \text{ is a solution of } D \quad \text{and } t_S\sigma = t
\]

(c) [30 lines] Assuming that a (black box) linear arithmetic constraint solving procedure \(A\) is available (given a Boolean combination of linear equations, \(A\) returns 1 if it is satisfiable and 0 otherwise), design an extension of the deducibility constraint solving procedure to the constraints of the previous question. Show that it allows to decide the existence of an attack on weak secrecy.

(d) [27 lines] Using this new formalism, prove that there is no attack on the weak secrecy of \(n_B\) in the scenario \(P'\), obtained by replacing \(P_B\) with \(P'_B\) in \(P\).

(e) **Bonus question:** How would you extend the Horn clauses formalism in order to take the length tests into account?

5. A name \(n\) is **strongly secret** in a frame \(\phi = \nu n.\nu m.s_1,\ldots,s_k\) if, for a name \(n'\), the two frames \(\nu n,\nu n'.\nu m.s_1,\ldots,s_k, n'\) and \(\nu n,\nu n',\nu m.s'_1,\ldots,s'_k, n'\) are statically equivalent, where \(s'_i\) is the term \(s_i\), in which \(n\) is replaced with \(n'\).

(a) [4 lines] Give an example of a frame \(\phi\) such that \(n\) is weakly secret \((\phi \not\vdash n)\) and \(n\) is not strongly secret.

(b) [6 lines] Conversely, show that, if \(n\) is strongly secret in \(\phi\), then it is weakly secret in \(\phi\)

(c) **Bonus question:** A name \(n\) is strongly secret in a process \(P\), if, for any trace of \(P\), \(n\) is strongly secret in the final frame of the trace.

Is \(n_B\) strongly secret in the process \(P'\) of the question 4d?
Solution

1. 

\[ P_A(a, b) = \nu n_A. \text{out}(\text{aenc}(\langle a, n_A \rangle, b)). \text{in}(y). \]

\[ \text{let } y_1 = \text{adec}(y, \text{sk}(c)) \text{ in } \text{let } y_2 = \pi_1(y_1) \text{ in } \]

\[ \text{let } y_3 = \pi_2(y_1) \text{ in } \text{let } y_4 = \pi_1(y_3) \text{ in } \text{let } y_5 = \pi_2(y_3) \text{ in } \]

\[ \text{if } y_2 = n_A \land y_5 = b \text{ then out}(\text{aenc}(y_4, b)) \cdot 0 \]

\[ P_A(a, b) = \nu n_A. \text{out}(\text{aenc}(\langle a, n_A \rangle, b)). \text{in}(\text{aenc}(\langle n_A, \langle x, b \rangle \rangle, a)). \text{out}(\text{aenc}(x, b)). 0 \]

2. (a) We rename the name \( n_B \) generated in \( P_B(a, c) \) into \( n'_B \) (names are pushed in front of the process).

When \( P_B(a, b) \) moves first:

\[ a, b, c, \text{sk}(c) \ni \text{aenc}(\langle a, x_3 \rangle, b) \]

\[ a, b, c, \text{sk}(c), \text{aenc}(\langle x_3, \langle n_B, b \rangle \rangle, a) \ni \text{aenc}(\langle c, y_3 \rangle, a) \]

When \( P_B(a, c) \) moves first:

\[ a, b, c, \text{sk}(c) \ni \text{aenc}(\langle c, y_3 \rangle, a) \]

\[ a, b, c, \text{sk}(c), \text{aenc}(\langle y_3, \langle n'_B, a \rangle \rangle, c) \ni \text{aenc}(\langle c, x_3 \rangle, b) \]

(b) In order to avoid branching too much, we may first observe that it is possible to apply the rules to the constraints in increasing order of the left members of deducibility constraints, without breaking the completeness proof.

There is then only one possible rule that applies to the first constraint in the system: \( R_5 \). This yields the system

\[ C_{11} = \left\{ \begin{array}{l}
  a, b, c, \text{sk}(c) \ni b \\
  a, b, c, \text{sk}(c) \ni \langle a, x_3 \rangle \\
  a, b, c, \text{sk}(c), \text{aenc}(\langle x_3, \langle n_B, b \rangle \rangle, a) \ni \text{aenc}(\langle c, y_3 \rangle, a)
\end{array} \right\} \]

We may remove the first constraint, according to \( R_1 \). Then only \( R_5 \) can be applied to the second constraint, that yields:

\[ C_{12} = \left\{ \begin{array}{l}
  a, b, c, \text{sk}(c) \ni a \\
  a, b, c, \text{sk}(c) \ni x_3 \\
  a, b, c, \text{sk}(c), \text{aenc}(\langle x_3, \langle n_B, b \rangle \rangle, a) \ni \text{aenc}(\langle c, y_3 \rangle, a)
\end{array} \right\} \]

Again, the first constraint can be removed in both cases and (in both situations) the second constraint is solved. We may then apply either \( R_5 \) or \( R_2 \) on the last constraint in both situations. This yields respectively:
\[ C_{13} = \begin{cases} 
  a, b, c, sk(c) \vdash x_3 \\
  a, b, c, sk(c), aenc((x_3, \langle n_B, b \rangle), a) \vdash a \\
  a, b, c, sk(c), aenc((x_3, \langle n_B, b \rangle), a) \vdash \langle c, y_3 \rangle 
 \end{cases} \]

\[ C_{14} = \begin{cases} 
  a, b, c, sk(c) \vdash c \\
  a, b, c, sk(c), aenc((c, \langle n_B, b \rangle), a) \vdash aenc((c, \langle n_B, b \rangle), a) \\
  x_3 = c \land y_3 = \langle n_B, b \rangle 
 \end{cases} \]

After two more steps, we get 2 solved forms in each case:

\[ C_{15} = \begin{cases} 
  a, b, c, sk(c) \vdash x_3 \\
  a, b, c, sk(c), aenc((x_3, \langle n_B, b \rangle), a) \vdash y_3 
 \end{cases} \]

\[ C_{16} = \{ x_3 = c \land y_3 = \langle n_B, b \rangle \} \]

(c) In order to find all attacks on the weak secrecy of \( n_B \), we add in both cases the constraint

\[ a, b, c, sk(c), aenc((x_3, \langle n_B, b \rangle), a), aenc((y_3, \langle n'_B, a \rangle), a) \vdash n_B \]

- Together with \( C_{15} \), this yields a failure (no solution) since we cannot reach a solved form.
- Together with \( C_{16} \), this yields \( x_3 = c \land y_3 = \langle n_B, b \rangle \) and

\[ a, b, c, sk(c), aenc((c, \langle n_B, b \rangle), a), aenc((\langle n_B, b \rangle, \langle n'_B, a \rangle), a) \vdash n_B \]

which simplifies to \( \top \) using the rule \( R_1 \), since there is a proof

\[
\frac{aenc(\langle n_B, b \rangle, \langle n'_B, a \rangle), c \quad sk(c)}{\langle n_B, b \rangle, \langle n'_B, a \rangle} {\langle n_B, b \rangle, \langle n'_B, a \rangle} {n_B}
\]

In the end, there is one (and only one) attack in this scenario, that corresponds to playing \( P_B(b, a) \) first and using the bindings \( x_3 = c \land y_3 = \langle n_B, b \rangle \).

(d) Since \( n_B \) is generated by a honest agent, and its target is a honest agent, we can conclude that the protocol is flawed.
3. (a)

\[
\begin{align*}
(1) & \quad \rightarrow \text{att}(a) \\
(2) & \quad \rightarrow \text{att}(b) \\
(3) & \quad \rightarrow \text{att}(c) \\
(4) & \quad \rightarrow \text{att}(sk(c)) \\
(5) & \quad \text{att(aenc}(\langle a, x_3 \rangle, b) \rightarrow \text{att(aenc}(\langle x_3, \langle n_b, b \rangle \rangle, a)) \\
(6) & \quad \text{att(aenc}((c, y_3), a) \rightarrow \text{att(aenc}(\langle y_3, \langle n_b', a \rangle \rangle, c)
\end{align*}
\]

(b) The attacker’s rules:

\[
\begin{align*}
(7) & \quad \text{att}(x), \text{att}(y) \rightarrow \text{att}(\langle x, y \rangle) \\
(8) & \quad \text{att}(x), \text{att}(y) \rightarrow \text{att(aenc}(\langle x, y \rangle)) \\
(9) & \quad \text{att}(\langle x, y \rangle) \rightarrow \text{att}(x) \\
(10) & \quad \text{att}(\langle x, y \rangle) \rightarrow \text{att}(y) \\
(11) & \quad \text{att(aenc}(\langle x, y \rangle), \text{att}(sk(y)) \rightarrow \text{att}(x)
\end{align*}
\]

Selected literals are underlined. We obtain successively:

\[
\begin{align*}
(4) + (11) = (16) & \quad \text{att(aenc}(x, c) \rightarrow \text{att}(x) \\
(5) + (8) = (12) & \quad \text{att(aenc}(\langle a, x_3 \rangle), \text{att}(b) \rightarrow \text{att(aenc}(\langle x_3, \langle n_b, b \rangle \rangle, a)) \\
(12) + (7) = (13) & \quad \text{att}(a), \text{att}(x_3), \text{att}(b) \rightarrow \text{att(aenc}(\langle x_3, \langle n_b, b \rangle \rangle, a)) \\
(13) + (1) = (14) & \quad \text{att}(x_3), \text{att}(b) \rightarrow \text{att(aenc}(\langle x_3, \langle n_b, b \rangle \rangle, a)) \\
(14) + (2) = (15) & \quad \text{att}(x_3) \rightarrow \text{att(aenc}(\langle x_3, \langle n_b, b \rangle \rangle, a))
\end{align*}
\]

(15) subsumes (13) and (14), that are deleted. (6) + (15) yields

\[
\begin{align*}
(17) & \quad \text{att}(c) \rightarrow \text{att(aenc}(\langle \langle n_b, b \rangle, \langle n_b', a \rangle \rangle, c))
\end{align*}
\]

And then successively:

\[
\begin{align*}
(17) + (3) = (18) & \quad \rightarrow \text{att(aenc}(\langle \langle n_b, b \rangle, \langle n_b', a \rangle \rangle, c) \\
(16) + (18) = (19) & \quad \rightarrow \text{att}(\langle \langle n_b, b \rangle, \langle n_b', a \rangle \rangle) \\
(10) + (19) = (20) & \quad \rightarrow \text{att}(n_b) \\
(10) + (20) = (21) & \quad \rightarrow \text{att}(n_b)
\end{align*}
\]

4. (a)

\[
P_B(b, a) = \nu n_B. \text{in(aenc}(\langle a, x_3 \rangle, b)) . \text{if } L(x_3) = 1 \text{ then out(aenc}(\langle x_3, \langle n_B, b \rangle \rangle, a)) \cdot 0
\]

(b) Constraint systems consist in conjunctions of deducibility constraints and Boolean combinations of constraints of the form $L(u_i) = n_i$. A solution of such a constraint system is a solution $\sigma$ of the deducibility constraint part such that, when applied to the length constraint part, we get a valid condition.

Assume a process $P = (\nu n_1, \ldots, \nu n_k). (P_1 || \cdots || P_m)$. We omit $\nu n_1, \ldots, n_k$ that are assumed as parameters in what follows. We define $[P] = [P_1 || \cdots || P_n][\emptyset, T, \emptyset]$ and $[0](t, H, S) = \{ (t, S) \}$.

$[P_1 || \cdots || P_m](t, H, S) = \bigcup_{i, P_i \neq 0} [P_1 || \cdots || P_{i-1} || P_{i+1} || \cdots || P_m : P_i](t, H, S)$
\[
\begin{align*}
\Gamma : \text{in}(u).P\ &\equiv \ [P]\{(t, H, S)\} \\
\Gamma : \text{out}(u).P\ &\equiv \ [P]\{(t, \text{out}(u), H \cup \{u\}, S)\} \\
\text{if } C \text{ then } P_1 \text{ else } P_2\ &\equiv \ [P_1](t, H, S) \cup [P_2](t, H, S \cup \{\neg C\})
\end{align*}
\]

By induction on \( n \), If

\[ P \overset{\alpha_1}{\longrightarrow} \ldots \overset{\alpha_n}{\longrightarrow} P' \]

Then \( [P'](t_S, H, S) \subseteq [P] \) for some \( t_S, H, S \) such that there is a substitution \( \sigma \) satisfying \( t_S \sigma = \alpha_1 \ldots \alpha_n \), \( H \sigma = H_n \) where \( H_n \) is the sequence of output messages in \( \alpha_1, \ldots, \alpha_n \), and \( \sigma \) is a solution of \( S \). In the base case, both \( t_S \) and \( t = \alpha_1 \ldots \alpha_n \) are empty. Otherwise, we investigate all possible moves of \( P' \). For instance if \( P' \overset{\text{in}(u)}{\longrightarrow} P'' \), then \( P' = \text{in}(v).Q\|R \), \( H_n \vdash u \), \( u = v \sigma_1 \), \( P'' = Q\sigma_1\|R \). (This can be detailed)

Conversely, if \( [P'](t, H, S) \subseteq [P] \) and \( \sigma \) is a solution of \( S \), then there is a \( P'' \) (not necessarily \( P' \)) such that \( P \overset{\text{out}(u)}{\longrightarrow} P'' \). (This can be detailed).

(c) Let \((D, C)\) be a constraint system that consists of a deducibility constraint part \( D \) and a length constraint part \( C \). First, we solve \( D \) using the procedure of the lectures: we compute a finite set of deducibility constraints \( D_1, \ldots, D_n \) such that

- \( \sigma \) is a solution of \( D \) if \( \sigma \) is a solution of some \( D_i \)
- Every \( D_i \) is a solved form, i.e. consists of a substitution \( \sigma_i \) and a conjunction of deducibility constraints \( t_i \vdash x_i \) where \( x_i \) is not in the domain of \( \sigma \).

If \( n = 0 \), then \((D, C)\) is unsolvable. Otherwise, for each \( D_i \), we consider the constraint \( C_i = C\sigma_i \). Equations are simplified according to the semantics of \( L \):

\[
\begin{align*}
L((u, v)) &= n \quad \rightarrow \quad L(u) + L(v) = n - 1 \quad \text{If } n \geq 3 \\
L((u, v)) &= n \quad \rightarrow \quad \perp \quad \text{If } n < 3 \\
L(a) &= 1 \quad \rightarrow \quad \top \quad \text{If } a \text{ is a name} \\
L(a) &= n \quad \rightarrow \quad \perp \quad \text{If } a \text{ is a name and } n > 1
\end{align*}
\]

These rules are obviously correct and terminating and the simplified formulas are Boolean combinations of formulas of the form \( L(x_1) + \cdots + L(x_n) = m \) where \( x_1, \ldots, x_n \) are (not necessarily distinct) variables.

Now, assign, for each variable \( x_i \) an integer variable \( z_i \) (its length). The simplified constraint \( C_i \) can be seen as a linear constraint \( C_i' \) over the variables \( z_i \). Using the black box arithmetic constraint solving algorithm, we decide whether \( C_i' \) has a solution. If it does not, \((D_i, C_i)\) is removed. Otherwise, we keep \((D_i, C_i)\) as a solved form of our constraint systems. We claim that such a solved form always has a solution: given an assignment \( \theta \) of the variables \( z_i \) that satisfies \( C_i \), we can find an assignment \( \sigma \) of the variables \( x_i \) such that \( x_i\sigma \) can be constructed by the attacker and \( L(x_i\sigma) = z_i \).

This yields therefore a decision procedure for the weak secrecy of \( s \)

i. Compute the symbolic semantics \([P]\) of the process \( P \)
ii. For each \((t, D) \in \llbracket P \rrbracket\), let \(M\) be the set of output messages in \(t\). Add the constraint \(M \vdash s\).

iii. If one of the resulting constraint has a solution, then there is an attack. Otherwise \(s\) is weakly secure.

(d) First, we may always assume that the output actions are executed first, when they are enabled: if there is an attack on a weak secrecy, then there is an attack for such an interleaving since output actions only give more power to the attacker.

In the question 2d, we considered the case where the input action of \(P_B(b, a)\) was performed before the input action of \(P_B(a, c)\). We now compute the constraint corresponding to the other ordering of the input actions. This yields successively the following constraint systems (using the same strategy as in the question 2d):

\[ C_{21} \rightsquigarrow R_5 C_{22} \rightsquigarrow R_1 C_{22}'' \rightsquigarrow R_5 C_{23}, C_{22}'' \rightsquigarrow R_2 C_{24} \text{ and } C_{23} \rightsquigarrow R_1 R_5 C_{25}, C_{24} \rightsquigarrow R_1 C_{26}. \]

\[
C_{21} = \begin{cases} 
  a, b, c, \text{sk}(c) \not\vdash a \\
  a, b, c, \text{sk}(c) \not\vdash \langle c, y_3 \rangle \\
  a, b, c, \text{sk}(c), \text{aenc}(\langle y_3, \langle n_B', a \rangle \rangle, c) \not\vdash \text{aenc}(\langle a, x_3 \rangle, b) 
\end{cases}
\]

\[
C_{22} = \begin{cases} 
  a, b, c, \text{sk}(c) \not\vdash c \\
  a, b, c, \text{sk}(c) \not\vdash y_3 \\
  a, b, c, \text{sk}(c), \text{aenc}(\langle y_3, \langle n_B', a \rangle \rangle, c) \not\vdash \text{aenc}(\langle a, x_3 \rangle, b) 
\end{cases}
\]

\[
C_{23} = \begin{cases} 
  a, b, c, \text{sk}(c) \not\vdash y_3 \\
  a, b, c, \text{sk}(c), \text{aenc}(\langle y_3, \langle n_B', a \rangle \rangle, c) \not\vdash b \\
  a, b, c, \text{sk}(c), \text{aenc}(\langle y_3, \langle n_B', a \rangle \rangle, c) \not\vdash \langle a, x_3 \rangle 
\end{cases}
\]

\[
C_{24} = \begin{cases} 
  a, b, c, \text{sk}(c) \not\vdash a \\
  a, b, c, \text{sk}(c), \text{aenc}(\langle a, \langle n_B', a \rangle \rangle, c) \not\vdash \text{aenc}(\langle a, \langle n_B', a \rangle \rangle, b) \\
  y_3 = a \land x_3 = \langle n_B', a \rangle 
\end{cases}
\]

\[
C_{25} = \begin{cases} 
  a, b, c, \text{sk}(c) \not\vdash y_3 \\
  a, b, c, \text{sk}(c), \text{aenc}(\langle y_3, \langle n_B', a \rangle \rangle, c) \not\vdash x_3 
\end{cases}
\]

\[ C_{26} = \{ \ y_3 = a \land x_3 = \langle n_B', a \rangle \}
\]

In the end \(C_{25}\) and \(C_{26}\) are the two solved forms corresponding to this interleaving of actions.

We only have to consider the solved forms computed in the question 2d and the above solved forms, together with the new constraints.
Together with the length constraints, this yields

\[
C'_{15} = \begin{cases} 
\{ a, b, c, \text{sk}(c) \} \vdash x_3 \\
\{ a, b, c, \text{sk}(c), \text{aenc}(\langle x_3, \langle n_B, b \rangle \rangle, a) \} \vdash y_3 \\
L(x_3) = 1 \land L(y_3) = 1 
\end{cases}
\]

\[
C'_{16} = \begin{cases} 
x_3 = c \land y_3 = \langle n_B, b \rangle \land L(c) = 1 \land L(\langle n_B, b \rangle) = 1 
\end{cases}
\]

\[
C'_{25} = \begin{cases} 
\{ a, b, c, \text{sk}(c) \} \vdash y_3 \\
\{ a, b, c, \text{sk}(c), \text{aenc}(\langle y_3, \langle n'_B, a \rangle \rangle, c) \} \vdash x_3 \\
L(x_3) = 1 \land L(y_3) = 1 
\end{cases}
\]

\[
C'_{26} = \begin{cases} 
y_3 = a \land x_3 = \langle n'_B, a \rangle \land L(\langle n'_B, a \rangle) = 1 \land L(a) = 1 
\end{cases}
\]

Now, \( C'_{16} \) and \( C'_{26} \) become unsatisfiable and are removed: there are only two symbolic traces that correspond to \( C'_{15} \) and \( C'_{25} \). If we add now

\[
a, b, c, \text{sk}(c), \text{aenc}(\langle x_3, \langle n_B, b \rangle \rangle, a), \text{aenc}(\langle y_3, \langle n'_B, a \rangle \rangle, c) \vdash n_B
\]

to the constraints, both become unsatisfiable since there is no rule that can be applied to this last constraint.

It follows that the weak secrecy of \( n_B \) is preserved in this scenario.

5. (a) Consider \( \phi = vn.va.a, \text{aenc}(n, a) \). \( n \) is weakly secret: there is no local proof of \( n \) from the two assumptions. \( vn.va.vn'.a, \text{aenc}(n, a), n' \neq vn.va.vn'.a, \text{aenc}(n', a), n' \): consider the two recipes \( C_1 = x_2 \) and \( C_2 = \text{aenc}(x_3, x_1) \). \( C_1\phi_1 \neq C_2\phi_1 \downarrow, \) while \( C_1\phi_2 \downarrow = C_2\phi_2 \downarrow \).

(b) Assume \( \phi \vdash n \). Then there is a recipe \( C_1 \) such that \( C_1\phi \downarrow = n \). If we let \( \phi' \) be the frame, in which \( n \) is replaced with \( n' \), then \( C_1\phi' \downarrow = n' \) (this can be proved by a simple induction on the number of rewriting steps: there is no rewrite rule that depends on a name). Consider then the two recipes: \( C_1 \) and \( C_2 = x_{k+1} \). If \( \phi_1 = vn'.\phi, n' \) and \( \phi'_1 = vn'.\phi', n' \), then \( C_1\phi_1 \downarrow = n \neq C_2\phi \downarrow = n' \) while \( C_1\phi'_1 \downarrow = n' = C_2\phi_2 \downarrow \). This prove the contrapositive of the question.