Exercise 1: Constraint solving

We consider the BAN-Yahalom protocol as described informally below. The purpose of this protocol is to establish a fresh session key \( K_{ab} \) between two participants \( A \) and \( B \). This is done through a server \( S \) who shares a long-term symmetric key with each participant. The key \( K_{as} \) (resp. \( K_{bs} \)) is a symmetric key shared between \( A \) (resp. \( B \)) and \( S \).

1. \( A \to B: A, N_a \)
2. \( B \to S: B, N_b, \{A, N_a\}_{K_{bs}} \)
3. \( S \to A: N_b, \{B, K_{ab}, N_a\}_{K_{as}}, \{A, K_{ab}, N_b\}_{K_{bs}} \)
4. \( A \to B: \{A, K_{ab}, N_b\}_{K_{bs}}, \{\ \} \)

We consider the constraint system \( C \) given below with \( T_0 = \{a, b, n, k\} \) and the classical inference system \( I_{DY} \) to deal with symmetric encryption and pair.

\[
\begin{align*}
T_1 & \overset{\text{def}}{=} T_0, \ (a, n_a) \not\vdash (a, x) \\
T_2 & \overset{\text{def}}{=} T_1, (b, \{n_b, \text{senc}(\langle a, x, k_{bs}\rangle)\}) \not\vdash (a, x') \\
T_3 & \overset{\text{def}}{=} T_2, (b, \{n'_b, \text{senc}(\langle a, x', k_{bs}\rangle)\}) \not\vdash (\text{senc}(\langle a, y, n_b\rangle, k_{bs}), \text{senc}(n_b, y)) \\
T_3 & \not\vdash y
\end{align*}
\]

1. Explain the scenario encoded by the constraint system \( C \). Who are the agents involved in this scenario? What are the roles played by each agent? How many instances of each role are they playing? What is the security property under study?

2. Check that the substitution \( \sigma = \{x \mapsto n_i; x' \mapsto (k_i, n_b); y \mapsto k_i\} \) is a solution of \( C \), i.e. give proof trees witnessing that \( \sigma \) is a solution of the constraint system \( C \). Explain the underlying attack on the protocol using the informal Alice & Bob notation.

3. Solve the deducibility constraint system \( C \), using the simplification rules of the lectures, and give all the solutions of the constraint system \( C \). You may notice that the simplification rules can always be applied to the first unsolved deducibility constraint, according to the completeness proof. Moreover, when using the rules \( R_2 \) and \( R_3 \), you may assume that \( t, u, t_1, t_2 \) are neither variables nor of the form \( \langle v_1, v_2 \rangle \). Completeness is still true under these hypotheses. This avoids unnecessary branching.

4. We propose to modify the protocol by adding tags. We consider public constants \( c_1, c_2, \ldots \) and we add such a constant in each ciphertext. For instance, messages 2 and 3 will be modified as follows:

\[
\begin{align*}
2. \ B \to S: B, N_b, \{c_1, A, N_a\}_{K_{bs}} \\
3. \ S \to A: N_b, \{c_2, B, K_{ab}, N_a\}_{K_{as}}, \{c_3, A, K_{ab}, N_b\}_{K_{bs}}
\end{align*}
\]
Of course, when an agent receives such a message, he will check (after decrypting the ciphertext) that the plaintext starts with the expected constant. Modify the constraint system $C$ to reflect the changes done on the protocol. Let $C_{\text{tag}}$ the resulting constraint system. Is $C_{\text{tag}}$ satisfiable? Could you comment on the usefulness of such a tagging mechanism from a security point of view?

**Exercise 2 : Blind signatures**

We want to study the intruder deduction problem for the inference system $I_{\text{sign}}$ given below:

$$
\begin{array}{cccc}
  x & y & \text{blind}(x, y) & \text{sign}(x, y) \\
  \text{blind}(x, y) & \text{sign}(x, y) & \text{blind}(x, y) & y \\
  \text{sign}(\text{blind}(x, y), z) & y & \text{sign}(x, z)
\end{array}
$$

We consider two binary function symbols $\text{sign}$ and $\text{blind}$. Intuitively, the term $\text{sign}(m, k)$ represents the signature of the message $m$ with the key $k$, and the term $\text{blind}(m, r)$ represents the message $m$ hidden with the random factor $r$. The blind signature primitive has the following property: given the signature of a blind message, i.e. $\text{sign}(\text{blind}(m, r), k)$, and the blinding factor $r$, it is possible to compute the signature of the message $m$, i.e. $\text{sign}(m, k)$. This additional ability is also given to the intruder. This is the purpose of the last inference rule of the system $I_{\text{sign}}$.

1. Show that the inference system $I_{\text{sign}}$ is not local w.r.t. the notion of syntactic subterm that we have seen during the lectures.

   We consider a notion of extended subterms defined as follows: $st_{\text{ext}}(t)$ is the smallest set such that $st(t) \subseteq st_{\text{ext}}(t)$ and

   $$
   \text{if } \text{sign}(\text{blind}(u_1, u_2), u_3) \in st_{\text{ext}}(t) \text{ then } \text{sign}(u_1, u_3) \in st_{\text{ext}}(t).
   $$

   This notion is extended to sets of terms as follows: $st_{\text{ext}}(T) = \bigcup_{t \in T} st_{\text{ext}}(t)$.

2. Let $T_0 = \{ \text{blind}(m, r), k, r \}$ and $u = \text{sign}(m, k)$. Give two different proof trees $\Pi_1$ and $\Pi_2$ of $T \vdash u$ in $I_{\text{sign}}$ having minimal size (size = number of nodes).

3. Show that the inference system $I_{\text{sign}}$ is local w.r.t. the notion of extended subterms.

4. Show that the intruder deduction problem is decidable

   Input : a finite set $T$ of terms, and a term $u$;

   Output : Is $u$ deducible from $T$ in $I_{\text{sign}}$?

   Justify the termination of your algorithm.

5. We consider the simplification rules seen during the lectures where the underlying deduction relation used in $R_1$ is $I_{\text{sign}}$, and the notion of subterm used in $R_2$ and $R_3$ is the notion of extended subterm mentioned above. We consider the following theorem:

   **Termination** : There is no infinite chain $C \leadsto_{\sigma_1} C_1 \ldots \leadsto_{\sigma_n} C_n$.

   **Correctness** : If $C \leadsto_{\sigma}^* C'$ for some constraint system $C'$ and some substitution $\sigma$ and if $\theta$ is a solution of $C'$ then $\sigma \theta$ is a solution of $C$.

   **Completeness** : If $\theta$ is a solution of $C$, then there exist a solved constraint system $C'$ and substitutions $\sigma$, $\theta'$ such that $\theta = \sigma \theta'$, $C \leadsto_{\sigma}^* C'$ and $\theta'$ is a solution of $C'$.

   Say whether each of these statements is true or not. You will provide a short explanation to justify a positive answer, and a counter-example to illustrate a negative answer.
Exercice 3 : Encrypted Key Exchange protocol

The EKE protocol is designed to solve the problem of authenticated key exchange while being resistant against dictionary attacks. EKE is a password-only protocol: the password $pw$ is the only shared data between the two participants $A$ and $B$.

1. $A \rightarrow B$ : $\{\text{pkey}\}_{pw}$
2. $B \rightarrow A$ : $\{\{R\}_\text{pkey}\}_{pw}$
3. $A \rightarrow B$ : $\{N_a\}_R$
4. $B \rightarrow A$ : $\{N_a, N_b\}_R$
5. $A \rightarrow B$ : $\{N_b\}_R$

First, $A$ generated a fresh private/public key pair, and then sends the public key encrypted with the password $pw$ shared with $B$. Then $B$ extracts the public key, generates a fresh session key $R$, encrypts it with the public key, and encrypts the result with the password. $B$ sends this message to $A$. Then, nonces $N_a$ and $N_b$ are exchanged to perform the “hand-shaking” necessary to defend against replay attacks.

1. Give a signature $\mathcal{F}$ and an equational theory $\mathcal{E}$ suitable to model this protocol in a reasonable way. We will assume that the operations symmetric encryption and decryption are commutative, i.e. $\text{senc}(\text{sdec}(x, y), y) = x$ is one of the equations in $\mathcal{E}$. In the following, all the function symbols will be assumed to be public. So, your model is supposed to be reasonable in this setting.

2. Write the processes $P_A(pw)$ and $P_B(pw)$ to model one session of the role $A$ and one session of the role $B$.

In the following, we consider the frame $\phi_0 = \text{new} \ pw. \text{new} \tilde{n}. \sigma_0$ obtained after a normal execution of one session this protocol. In other words, the attacker does not try to intercept, modify, or inject some messages during this execution.

3. Write the substitution $\sigma_0$, and the names $\tilde{n}$ that represent such an execution. We are interested in the following static equivalence (modulo the theory $\mathcal{E}$ you defined above):

$$\text{new} \tilde{n}. \sigma_0 \overset{?}{\sim} \text{new} \ pw. \text{new} \tilde{n}. \sigma_0$$

Does this static equivalence hold or not? Justify your answer.

Hint: You could rely on the algorithm seen during the lectures.

4. Using the result obtained at the previous question, deduce whether $\phi_0$ is resistant to dictionary attack against $pw$ or not? Justify your answer.

5. Now, we assume that the asymmetric cryptosystem that is used to implement this protocol is which-key revealing, allowing an attacker to deduce if two ciphertexts were encrypted under the same key. Propose a signature $\mathcal{F}^+$ and a set of equations $\mathcal{E}^+$ to reflect this new attacker model. Is $\phi_0$ resistant to dictionary attacks against $pw$ (considering the signature $\mathcal{F}^+$ and the equational theory $\mathcal{E}^+$)?
Exercise 1: Constraint solving

1. Actually a is involved to send the first message and then only b is involved in this scenario (in 2 different sessions). The agent b plays two instances of the role B (and only the two first actions for the second instance). This role is made up of one input (message 1) followed by one output (message 2), and then the reception of a last message (message 4). If the message has the expected form, then the agent accepts the key he received during this exchange. This constraint system encodes the secrecy of the key $K_{ab}$ as received by the agent b who plays the role B.

2. We apply the substitution $\sigma$ on $C$ and we check that each rhs is indeed deducible from the lhs.

\[ \Pi^2_{n_b} = \left\{ \begin{array}{ll}
{b,\langle n_b, \text{senc}(\langle a, k_i \rangle), k_{bs}\rangle)} \\
{\langle n_b, \text{senc}(\langle a, k_i \rangle), k_{bs}\rangle)} \\
\end{array} \right\} \]

We have the following prooftrees:

The attack can be informally described as follows:

1. $I(A) \rightarrow B : A, N_i$
2. $B \rightarrow (S) : B, N_b, \{A, N_i\} K_{bs}$

1. $I(A) \rightarrow B : A, (K_i, N_b)$
2. $B \rightarrow (S) : B, N'_b, \{A, (K_i, N_b)\} K_{bs}$

4. $I(A) \rightarrow B : \{A, (K_i, N_b)\} K_{bs}, \{N_b\} K_i$

3. Starting with $C$, and following the strategy suggesting in the question, we work on the first constraint and apply $R_f$ followed by $R_1$. This leads to the constraint system $C_1$:

\[ C_1 = \left\{ \begin{array}{ll}
T_1 \vdash x \\
T_2 \vdash \langle a, x' \rangle \\
T_3 \vdash \text{senc}(\langle a, \langle y, n_b \rangle\rangle, k_{bs}), \text{senc}(n_b, y) \\
T_3 \vdash y \\
\end{array} \right\} \]

Then, since the first constraint is now solved, we work on the second one and we apply again $R_f$ followed by $R_1$. This leads us to the system $C_2$:

\[ C_2 = \left\{ \begin{array}{ll}
T_1 \vdash x \\
T_2 \vdash x' \\
T_3 \vdash \text{senc}(\langle a, \langle y, n_b \rangle\rangle, k_{bs}), \text{senc}(n_b, y) \\
T_3 \vdash y \\
\end{array} \right\} \]
Then, we apply $R_f$ on the third constraint and we obtain $C_3$:

$$C_3 = \begin{cases} T_1 \vdash x \\ T_2 \vdash x' \\ T_3 \vdash \text{senc}(n_b, y) \\ T_3 \vdash \text{senc}(\langle a, \langle y, n_b \rangle \rangle, k_{bs}) \\ T_3 \vdash y \end{cases}$$

Again, the only option is to apply $R_f$ on the third constraint and after applying $R_1$ to get rid of $T_3 \vdash n_b$, we obtain $C_4$:

$$C_4 = \begin{cases} T_1 \vdash x \\ T_2 \vdash x' \\ T_3 \vdash y \\ T_3 \vdash \text{senc}(\langle a, \langle y, n_b \rangle \rangle, k_{bs}) \end{cases}$$

Then, on the fourth constraint, there are four possible options. We may apply:

(a) $R_2$, and we obtain $C_{41}$ (where $\sigma = \{x' \mapsto \langle y, n_b \rangle\}$);
(b) $R_f$, and we obtain $C_{42}$;
(c) $R_2$, and we obtain $C_{43}$ (where $\sigma = \{x \mapsto \langle y, n_b \rangle\}$);
(d) $R_3$, and we obtain $C_{44}$ (where $\sigma = \{x' \mapsto x\}$)

We only develop the two first cases. Actually, after some steps, we $C_{43}$ and $C_{44}$ are reduced to $\bot$ (no solution).

$$C_{41} = \begin{cases} T_1 \vdash x \\ T_2 \vdash \langle y, n_b \rangle \\ T_3\sigma \vdash y \\ T_3\sigma \vdash \text{senc}(\langle a, \langle y, n_b \rangle \rangle, k_{bs}) \end{cases}$$

$$C_{42} = \begin{cases} T_1 \vdash x \\ T_2 \vdash x' \\ T_3 \vdash y \\ T_3 \vdash k_{bs} \\ T_3 \vdash \langle a, \langle y, n_b \rangle \rangle \end{cases}$$

Since, no rule can be applied on the first unsolved constraint of $C_{42}$, we deduce that $C_{42}$ has no solution. Then, we apply $R_f$ on the second constraint of $C_{41}$ followed by $R_1$ on the resulting constraint. We get $C_5$:

$$C_5 = \begin{cases} T_1 \vdash x \\ T_2 \vdash y \\ T_3\sigma \vdash y \\ T_3\sigma \vdash \text{senc}(\langle a, \langle y, n_b \rangle \rangle, k_{bs}) \end{cases}$$

Then, we may apply $R_1$ on the two last constraints, and we obtain at the end

$$T_1 \vdash x \land T_2 \vdash y$$

Thus, any substitution such that $x$ is instantiated by a term $t_x$ deducible from $T_1$, $y$ is instantiated by a term $t_y$ deducible from $T_2\{x \mapsto t_x\}$, and $x'$ is mapped to $\langle t_y, n_b \rangle$ is a solution. These are the only solution of such a constant system $C$.
4. The constraint system $C_{\text{tag}}$ is as follows where $T_0^+ = T_0 \cup \{c_1, c_2, c_3, c_4\}$.

$$C_{\text{tag}} = \left\{ \begin{array}{l}
T_1^+ \overset{\text{def}}{=} T_0^+, \langle a, n_a \rangle \vdash \langle a, x \rangle \\
T_2^+ \overset{\text{def}}{=} T_1^+, \langle b, \langle n_b, \text{senc}(\langle c_1, \langle a, x \rangle \rangle) \rangle, k_{b_a} \rangle \vdash \langle a, x' \rangle \\
T_3^+ \overset{\text{def}}{=} T_2^+, \langle b, \langle n_b', \text{senc}(\langle c_1, \langle a, y \rangle \rangle) \rangle, k_{b_a} \rangle \vdash \langle \text{senc}(\langle c_3, \langle a, \langle z, n_b \rangle \rangle \rangle), k_{b_a} \rangle, \text{senc}(\langle c_4, n_b \rangle, z) \rangle \\
T_3^+ \overset{\text{def}}{=} T_2^+, \langle b, \langle n_b', \text{senc}(\langle c_1, \langle a, y \rangle \rangle) \rangle, k_{b_a} \rangle \vdash \langle \text{senc}(\langle c_3, \langle a, \langle y, n_b \rangle \rangle \rangle), k_{b_a} \rangle
\end{array} \right\}
$$

Following the same strategy as the one described previously, we reach the system $C_{\text{tag}}^4$

$$C_{\text{tag}}^4 = \left\{ \begin{array}{l}
T_1^+ \vdash ? \ x \\
T_2^+ \vdash ? \ x' \\
T_3^+ \vdash ? \ y \\
T_3^+ \vdash \text{senc}(\langle c_3, \langle a, \langle y, n_b \rangle \rangle \rangle), k_{b_a})
\end{array} \right\}
$$

The option of applying $R_2$ is not possible anymore. The only possibility is to apply $R_1$, and this leads to a constraint system that is not satisfiable. Hence, we conclude that $C_{\text{tag}}$ is not satisfiable. The presence of such tags improves the security of the protocol by avoiding type confusion attacks as the one described in this exercise. This avoids confusion between different ciphertexts that are used at different places in the protocol.

**Exercise 2 : Blind signatures**

1. Let $T = \{\text{sign}(\text{blind}(\text{blind}(m, r_1), r_2), k); r_2; r_1\}$ and $u = \text{sign}(m, k)$. We have that $T \vdash u$ and any proof of this fact used the term $\text{sign}(\text{blind}(m, r_1), k)$ as an intermediate node. The term $\text{sign}(\text{blind}(m, r_1), k)$ is not a subterm of $T \cup \{u\}$.

2. We consider the proof trees $\Pi_1$ and $\Pi_2$ below:

$$\Pi_1 = \frac{\text{blind}(m, r)}{\text{sign}(m, k)} \quad \frac{m}{k} \quad \Pi_2 = \frac{\text{blind}(m, r)}{\text{sign}(m, k)} \quad \frac{\text{sign}(\text{blind}(m, r), k)}{r}$$

3. Given a proof tree $\Pi$, we define its size as its number of nodes plus the number of instances of the “special” rule that occurs in it (e.g. $\text{size}(\Pi_1) = 5$ whereas $\text{size}(\Pi_2) = 6$).

Let $T$ be a set of terms, and $u$ be a term such that $T \vdash u$. Let $\Pi$ be a proof tree witnessing this fact whose size is minimal. We show by induction on $\Pi$ that $\Pi$ only contains terms in $st_{\text{ext}}(T \cup \{u\})$. Moreover, if $\Pi$ is reduced to a leaf or ends with a “decomposition rule” (i.e. one of the two last rules), then $\Pi$ only contains terms in $st_{\text{ext}}(T)$. We do the proof by case analysis on the last rule of the proof tree.

- $\Pi$ is reduced to a leaf: the result trivially holds.
- $\Pi$ ends with an instance of the first inference rule: $u = \text{blind}(u_1, u_2)$ and we denote $\Pi_1$ and $\Pi_2$ the two direct sub-prooftrees of $\Pi$. By induction hypothesis, we have that $\Pi_1$ (resp. $\Pi_2$) only contains terms in $st_{\text{ext}}(T \cup \{u_1\})$ (resp. $st_{\text{ext}}(T \cup \{u_2\})$), and thus $\Pi$ only contains terms in $st_{\text{ext}}(T \cup \{u\})$ since $u = \text{blind}(u_1, u_2)$ and $u_1, u_2 \in st_{\text{ext}}(u)$. The case of the second inference rule is similar.
- $\Pi$ ends with an instance of the third inference rule with $\text{blind}(u, v)$ and $v$ as hypotheses. We denote by $\Pi_1$ and $\Pi_2$ the two direct sub-prooftrees of $\Pi$. By minimality, we know that $\Pi_1$ is either reduced to a leaf or ends with an instance of third rule. Thus, by induction
hypothesis, we have that $\Pi_1$ only contains terms in $st_{ext}(T)$, and thus $\text{blind}(u, v) \in st_{ext}(T)$. By induction hypothesis on $\Pi_2$, we deduce that $\Pi_2$ only contains terms in $st_{ext}(T \cup \{v\}) \subseteq st_{ext}(T)$ (since $\text{blind}(u, v) \in st_{ext}(T)$). We have also that $u \in st_{ext}(T)$, and this allows us to conclude that $\Pi$ only contains $st_{ext}(T)$.

- $\Pi$ ends with an instance of the fourth rule with $\text{sign}(\text{blind}(u_1, v), u_2)$ and $u_2$ as hypotheses. We denote by $\Pi_1$ and $\Pi_2$ the two direct sub-prooftrees of $\Pi$. We have that $\Pi_1$ is either reduced to a leaf, or ends with an instance of the 2nd, 3rd, or 4th inference rule. Actually, thanks to minimality, an instance of the 2nd rule is not possible (indeed a smaller proof will be possible in this case). Thus, we have that $\Pi_1$ only contains terms in $st_{ext}(T \cup \{\text{sign}(\text{blind}(u_1, v), u_2)\})$, thus $v$ and $u = \text{sign}(u_1, u_2)$ are in $st_{ext}(T)$, and this allows us to conclude that $\Pi$ only contains terms in $st_{ext}(T)$.

Of course, this result allows us to conclude.

The saturation algorithm seen during the lectures applies. Correction comes form the fact that we only add deducible terms in the saturation set. Completeness is derived from the locality result. Regarding termination, we have to ensure that the set $st_{ext}(T \cup \{u\})$ is finite. We have that $|st_{ext}(t)| \leq |t| + |t|_{\text{blind}}$ (where $|t|$ is the number of symbols occurring in $t$ and $|t|_{\text{blind}}$ is the number of occurrence of the symbols $\text{blind}$ in $t$). This allows us to conclude.

Regarding termination, the same measure as the one seen during the lectures allows us to conclude ($\#\vars(\Pi), \Sigma_{u \in \text{ehs}(\Pi)}|u|$) with a lexicographical order. Regarding correctness, the same arguments also apply. In particular, correctness for the rule $R_1$ can be established as during the lectures. However, completeness is wrong:

$$m, r \vdash x \quad \text{sign}(x, k), r \vdash \text{sign}(m, k) \quad \text{with} \quad \theta = \{x \mapsto \text{blind}(m, r)\}.$$ 

There is no simplication rule that we can applied to progress towards this solution and the system is not in solved form.

**Exercise 3 : Encrypted Key Exchange protocol**

1. We consider the signature $\mathcal{F} = \{\text{senc}/2, \text{sdec}/2, \text{aenc}/2, \text{adec}/2, \text{pk}/1(\cdot)/2, \text{proj}_1/1, \text{proj}_2/1\}$, and the equational theory generated by the following equations:

\[
\text{sdec} (\text{senc} (x, y), y) = x \quad \text{adec} (\text{aenc} (x, y), y) = x \quad \text{proj}_1 (\langle x, y \rangle) = x \quad \text{proj}_2 (\langle x, y \rangle) = y
\]

Note: we may consider in addition the equation $\text{senc} (\text{sdec} (x, y), y) = x$.

2. $P_A(pw) = \text{new} \ sk. \text{out} (c, \text{senc} (\text{pk} (sk), pw)). \text{in} (c, x_1)$.

\[
\text{let} \ x_R = \text{adec} (\text{sdec} (x_1, pw), sk) \text{ in}
\text{new} \ n_a. \text{out} (c, \text{senc} (n_a, x_R)). \text{in} (c, x_2).
\]

if proj$_1(\text{sdec}(x_2, x_R)) = n_a$ then out$(c, \text{senc}(\text{proj}_2(\text{sdec}(x_2, x_R)), x_R))$.

\[
P_B(pw) = \text{in} (c, y_1). \text{let} \ y_{\text{pub}} = \text{sdec} (y_1, pw) \text{ in}
\text{new} \ r. \text{out} (c, \text{senc} (\text{aenc} (r, y_{\text{pub}}), pw)). \text{in} (c, y_2).
\]

\[
\text{let} \ y_{n_b} = \text{sdec} (y_2, r) \text{ in} \text{new} \ n_b. \text{out} (c, \text{senc} (n_a, n_b, r)) \text{ in} (c, y_3). \text{if} \ y_3 = \text{senc} (n_b, r) \text{ then} 0.
\]

3. We have that

\[
\phi_0 = \text{new} \ pw. \text{new} \ sk, r, n_a, n_b. \{x_1 \mapsto \text{senc} (\text{pk} (sk), pw); x_2 \mapsto \text{senc} (\text{aenc} (r, \text{pk} (sk)), pw); x_3 \mapsto \text{senc} (n_a, r); x_4 \mapsto \text{senc} (n_a, n_b, r); x_5 \mapsto \text{senc} (n_b, r)\}
\]

Any test that is satisfied by $\text{new} \ pw. \text{new} \ n.C \phi_0$ will be also true in $\text{new} \ n.C \phi_0$. Thus, to check whether static equivalence holds between these two frames, we will focus on computing
the sets sat(\(\phi\)) and Eq(\(\phi\)) when \(\phi = \text{new } \tilde{n}.\sigma_0\). Then, we will check whether all these tests are also valid in \(\text{new } pw.\text{new } \tilde{n}.\sigma_0\).

We list below the elements in sat(\(\phi\)) together with their associated recipe, and the rule applied according to the definition of frame saturation.

\[
\begin{align*}
senc(pk(sk), pw) & \quad x_1 \quad \text{(1)} \\
senc(aenc(r, pk(sk)), pw) & \quad x_2 \quad \text{(1)} \\
senc(n_a, r) & \quad x_3 \quad \text{(1)} \\
senc((n_a, n_b), r) & \quad x_4 \quad \text{(1)} \\
senc(n_b, r) & \quad x_5 \quad \text{(1)} \\
pw & \quad pw \quad \text{(3)} \\
\text{pk(sk)} & \quad \text{sdec}(x_1, pw) \quad \text{(3)} \\
\text{aenc}(r, pk(sk)) & \quad \text{sdec}(x_2, pw) \quad \text{(3)}
\end{align*}
\]

In case, we have the equation senc(sdec(x, y), y) = x in our model, all the equalities that we can infer are actually trivial, and thus static equivalence holds. Otherwise, there is a test senc(sdec(x_1, pw), pw) = x_1 which holds in \(\phi\), and does not hold in new pw.\(\phi\) (note that we have first to rename \(\text{pw}\) with \(\text{pw}'\)).

4. Actually, we have new \(\tilde{n}.\sigma \sim \text{new } pw.\text{new } \tilde{n}.\sigma\) if, and only if, new pw.\text{new } \tilde{n}.\sigma is resistant to dictionary attack against \(\text{pw}\) with the definition seen during the lectures, i.e. if and only if:

\[
\text{new } pw.\text{new } \tilde{n}.(\sigma \cup \{x \mapsto \text{pw}\}) \sim \text{new } pw.\text{new } \tilde{n}.(\sigma \cup \{x \mapsto \text{pw}'\})
\]

We show the following equivalence:

\[
\text{new } \tilde{n}.\sigma \sim \text{new } pw.\text{new } \tilde{n}.\sigma
\]

if, and only if,

\[
\text{new } \tilde{n}.\sigma \sim \text{new } pw'.\text{new } \tilde{n}.(\sigma\{\text{pw} \mapsto \text{pw}'\}) \text{ by } \alpha\text{-conversion.}
\]

if, and only if,

\[
\text{new } pw, \text{new } \tilde{n}.(\sigma \cup \{x \mapsto \text{pw}\}) \sim \text{new } pw.\text{new } \tilde{n}.(\sigma \cup \{x \mapsto \text{pw}'\}) \text{ by } \alpha\text{-conversion.}
\]

Note that for this equivalence, it is easy to see that any test that distinguishes the two frames can be transformed in another test that distinguishes the two resulting frames (and conversely) by replacing the use of \(\text{pw}\) with \(x\) (or the converse).

Lastly, this equivalence holds if, and only if,

\[
\text{new } pw, \text{new } \tilde{n}.(\sigma \cup \{x \mapsto \text{pw}\}) \sim \text{new } pw'.\text{new } \tilde{n}.(\sigma \cup \{x \mapsto \text{pw}'\}) \text{ by } \alpha\text{-conversion.}
\]

Hence we conclude that \(\phi_0\) is resistant to dictionary attacks against \(\text{pw}\) if, and only if, the static equivalence studied at the previous question holds.

5. We consider two additional function symbol samekey/2 and ok/0, as well as, the equation samekey(senc(x_1, y), senc(x_2, y)) = ok. Clearly, \(\phi_0\) is not resistant against dictionary attack on \(\text{pw}\). It suffices to consider the test samekey(x_1, senc(n, x)) \(\not\equiv\) ok. This test holds on the frame new pw.\text{new } \tilde{n}.(\sigma \cup \{x \mapsto \text{pw}\}) but does not hold in new pw.\text{new } pw'.\text{new } \tilde{n}.(\sigma \cup \{x \mapsto \text{pw}'\}).