Formal Analysis of Real-World Security Protocols

Prof. Dr. Cas Cremers, Dr. Aleksi Peltonen
Alexander Dax, Niklas Medinger

`https://cms.cispa.saarland/farwsp24/`

Exercise Sheet 2

due 11.11.2024 at 14:00

Instructions:

You should submit exercises in groups of **two**. Submissions **must** include the name and matriculation number of **both** students. Please submit only once per group. Solutions must be submitted in PDF format to CMS. You should provide solutions in English. If there are additional files, they can be downloaded from the material section in CMS.

Exercise 1: Unification (30 points)

In the following exercise, we take a more formal view on how to unify two terms. Two terms s and t are said to be unifiable if there exists a substitution σ such that $s\sigma = t\sigma$; the substitution σ is then called an unifier of s and t. The unifier σ is called *most general unifier*, written $\sigma = \mathrm{mgu}(s,t)$, if any other unifier τ of s and t it can be represented as $\tau = \sigma\tau'$, for some substitution τ'.

As an example, consider $s = f(x,a)$ and $t = f(b,y)$. Here, the *mgu* of s and t would be $\sigma = \{x/b, y/a\}$.

We additionally lift this notion to sets of equations $Eq = \{s_1 = t_1, ...s_n = t_n\}$ and say that σ is a unifier for Eq if for all $i \in \{1,...,n\}$: $s_i\sigma = t_i\sigma$.

You can assume that $x, y, z..$ are variables and $a, b, ..$ are atomic/ground terms.

(a) (8 points) Compute the most general unifier for the following unification problems (if possible):

(1) $\{enc(x, h(x,y)) = enc(enc(y,z), h(y,z'))\}$
(2) $\{h(enc(x,x)) = z, h(h(enc(a,y))) = h(z)\}$
(3) $\{h(enc(x,x)) = z, h(h(enc(a,y))) = h(z), mac(x,y) = v\}$
(4) $\{prf(x,y) = prf(x',y'), y' = enc(x,a), enc(h(a),z) = y\}$

(b) (8 points) Check whether the following equations can be unified. If a unifier exists, present it.

(1) $\{prf(enc(x,y),z) = z_1, z_1 = prf(y_1, h(x_2,a)), x_2 = enc(a,b)\}$
(2) $\{prf(enc(a,y),z) = z_1, z_1 = x_1, x_1 = prf(enc(y_1,a),z_2), enc(y_1,a) = enc(b,x_2)\}$
(3) $\{prf(z, enc(x,y)) = prf(x_1,x_1), x = h(y_1,y_1), y = h(z_1,z_1)\}$
(4) $\{prf(z, enc(x,y)) = prf(x_1,x_1), x = h(y_1,y_1), y = h(x_2,z)\}$

(c) (6 points) Check for all s and t whether (i) σ is an unifier of s and t (ii) σ is the *most general unifier*:

s	t	σ
$sign(a, h(x), y)$	$nice(v, h(h(z)), b)$	$\{v/a, x/h(z), y/b\}$
$sign(x, enc(a,z), h(x))$	$sign(h_1(h_1(v)), y, h(w))$	$\{x/h_1(h_1(v)), y/enc(a,z), w/h_1(h_1(v))\}$
$enc(kdf(x), h(y))$	$enc(z, h(v))$	$\{x/a, z/kdf(a), y/v\}$

(d) (8 points) Prove or find a counterexample: If $|V(s)| > |V(t)|$ then there is no substitution σ with $t\sigma = s$. where V computes the set of variables from a term.

Exercise 2: Term Deduction (30 points)

For this exercise, assume all equational theories as they were defined in the lecture.

(a) (12 points) Let T be a set of terms defined as follows:

$$T = \{senc(k_2, \langle k_1, senc(k_1, k_3)\rangle), \langle k_1, k_1\rangle, senc(senc(k_1, k_3), k_1)\}$$

(1) Show that k_1 and k_2 are deducible from T.
(2) Show that k_3 is not deducible from T.
(3) Among the subterms of the terms in T, give those that are deducible.

(b) (8 points) Let T be a set of terms defined as follows:

$$T = \{senc(k_4, \langle k_1, senc(k_3, k_4)\rangle), \langle k_1, k_2\rangle, senc(senc(k_3, k_4), k_2)\}$$

(1) Show step by step that all keys in T are deducible.

(c) (10 points) Consider the following protocol:
1: $A \rightarrow B :< aenc(k_1, pk_B), aenc(k_2, pk_B) >$
2: $B \rightarrow A : senc(k_1, k_2)$
The keys k_1 and k_2 are freshly generated by A at the beginning of each session.

(1) Show how a network attacker can construct a term knowledge set T from which you can deduce k_1. Explain how the attacker achieves this and show the deduction.
(2) Does this work with k_2 as well?

Exercise 3: Modeling Protocols (40 points)

(a) (6 points) **Natural Numbers**

Recall the definition of natural numbers from the last exercise sheet.

For this and the following exercises, assume that natural numbers are defined by:

```
functions: Z/0, S/1
equations:
```

where $Z()$ represents the number 0 (zero) and $S(x)$ represents the successor of x, where x is a natural number. For instance, the number 2 would be equal to `S(S(Z()))`.

Using the defined function symbols, create a rule that

a) sends a zero value out.
b) takes any value, adds one to it, and sends it out.
c) takes any value bigger than zero, substracts one from it, and sends it out.

If you introduce new facts or function symbols, explain them.

(b) (16 points) **NAXOS Key Exchange (Variant)**

In this exercise, we will gradually convert a variant of the NAXOS key exchange protocol [1] into multiset rewriting rules using Tamarin's built-in fact symbols. Give a clear name to each rule you create.

Rule 1: Generate Public Key Pair

- Define a rule that generates a public key pair, (x, g^x), and stores the pair in a persistant system state.

Rule 2: Initiator Sends Public Keys

Define a rule for the Initiator:

- Retrieve a key pair from a system state.
- Send the public key to the network.
- Generate a fresh ephemeral key material
- Use a hash function $h/2$ to construct a **combined private key** $hskI$ from the ephemeral key material and the private key from the input.
- Construct the public key of $hskI$ and send in also to the network
- Store all key materials in a non-persistant state

Rule 3: Responder Computes Shared Key

Define a rule for the Responder:

- Receive the two public keys from the network.
- Retrieve a key pair from the system state.
- Create fresh ephemeral key material and construct $hskR$ similar to $hskI$ using h.
- Compute the shared secret key using the key pair, the received public keys, and a key derivation function, KDF/3.
  The inputs to the kdf are:
  - $(\text{A's public key})^{hskR}$
  - $(\text{A's combined public key})^{\text{B's secret key}}$
  - $(\text{A's combined public key})^{hskR}$
- Store the shared key in the system state and send the Responder's public keys to the network.

Rule 4: Initiator Computes Shared Key and Sends Message

Define a rule for the Initiator:

- Receive two public keys from the network.
- Retrieve the Initiator's stored key materials from the state.
- Compute the shared key analogous to the Responder. (Hint: the three inputs to the kdf are all Diffie-Hellman-style keys.)
- Store the shared key in the system state.
- Create a fresh message m and encrypt it with the shared key.
- Send the encrypted message to the network.

[1] `https://eprint.iacr.org/2006/073.pdf`

(c) (18 points) **Another Public Key Exchange**

Consider the following protocol in Alice & Bob notation.

$$\begin{aligned}
A \to B &: aenc(\langle A, N_A \rangle, pk_B) \\
B \to A &: aenc(N_B, pk_A), MAC(\langle B, A \rangle, K_0) \\
A \to B &: MAC(\langle A, B \rangle, K_0)
\end{aligned}$$

Again, A and B are the respective identities of A and B. For the MAC you may use a function symbol with out any equational theory. To verify the MAC use pattern matching (see course book, Chapter 3.2.3 for an example).

Assume both parties already have a private/public key in some fact $KeyA(sk, pk, A, B)$ and $KeyB(sk, pk, B, A)$.

First, A sends their identity and a nonce N_A encrypted under $B's$ public key to B. Then, B generates the nonce N_B and sends it encrypted under $A's$ public key. Additionally, B computes a key $K_0 = H(N_A, N_B)$ and uses it to create a MAC of the identies of A and B. Now, A decrypts the nonce, uses it to derive K_0, and verifies the MAC. Finally, A sends out another MAC (but with switched parameters) and B verifies this MAC. To finish the protocol, both A and B compute

$$K = MAC(MAC(\langle A, B \rangle, K_0), K_0).$$

Transform this protocol into multiset rewriting rules.