CryptoVerif - Practical Session 1

June 2023

1 The encrypt-then-MAC construction

1.1 encrypt-then-MAC is IND-CPA

Create a file *enc-then-MAC-IND-CPA.ocv*, which should contain a model of an Encrypt-then-Mac construction based on an `enc` function and a `mac` function, with an left-right encryption oracle modeling the IND-CPA test based on a boolean b, that should be proven secret.

You should first try it without any other guidance. If you feel lost, you can check the step-by-step instructions below. And otherwise call us if it does not help.

1. Declare types for the mac keys, the encryption keys, declare the number of possible queries to the LR oracle, and the propabilities of breaking the underlying `enc` scheme and `mac` scheme.
2. Expand the *IND_CPA_sym_enc* and *SUF_CMA_det_mac* library for the `enc` and `mac` function.
3. Declare using the `letfun` construct the function that takes as argument a message, an encryption key for `enc`, a mac key for `mac`, and produces the actual encrypt then MAC cyphertext. You may beed to declare an additional function symbol.
4. Declare a LeftRight macro process that based on the secret boolean, an encryption key and a mac key, should declare an arbitrary number of an encryption oracle that can be queried by the attacker with two messages and then return the encryption of either one based on the secret bit.
5. Declare the main process, sampling the secret bit, an encryption key, a mac key, and that runs the macro process.
6. Make CryptoVerif auto prove the query.

1.2 encrypt-then-MAC is IND-CTXT

Create a file *enc-then-MAC-INT-CTXT.ocv*, copy-pasting from the previous file the preamble declarations for types, proba, expand, fun and letfun.

Model the INT-CTXT property and use CryptoVerif to prove it.

1. Define an additional `letfun` modeling the full decryption of a cyphertext, given the corresponding decryption key and mac key. In case of failure, the decryption should return the builtin constant `bottom`.
2. Define a basic encryption macro process for some given keys, that will spawn an arbitrary number of encryption oracles for a given message.

3. Define a macro process for some given keys, that should spawn an arbitrary number of decryption test oracles corresponding to the INT-CTXT test.

4. Specify the security property with an event `bad` that must never occur. You may need to declare a table, and slightly modify the basic encryption oracle as well.

5. Define the main process launching the macro processes in parallel with a fixed couple of keys.

6. Make CryptoVerif auto prove the query.