Compiling Protocol Narrations into Applied Pi Processes

Hans Hüttel and Sam Sepstrup Olesen

Department of Computer Science, Aalborg University, Denmark

When designing or presenting cryptographic protocols, it is common to use informal protocol descriptions, such as *protocol narrations*, to describe the intended execution of a protocol run as a sequence or directed graph of message exchanges between the associated principals[BN07]. An example of a protocol narration can be seen below. Encryption of a message M with a key N denoted by $\{M\}_N$.

$$\begin{aligned} A &\rightsquigarrow S : \{B\}_{k_{AS}} \\ S &\rightsquigarrow A : \{k_B\}_{k_{AS}} \\ A &\rightsquigarrow B : \{m\}_{k_B} \end{aligned}$$

This narration describes a simple protocol, in which a principal A can ask a key server S for a key to communicate with a principal B. However, some assumptions about ownership of keys are made, namely that A and S share a key k_{AS} and that B knows the key k_B that is sent to A by S. Other important aspects are the actions that principals are supposed to perform on messages that are received, e.g. decryption and consistency checks are implicit. We obviously expect the principals to attempt decryption of the messages they receive. We also expect that the key-server S only sends the key k_B, if that is the key requested by A.

An important challenge is to be able to create an executable implementation of the protocol based on a narration. In [BN07] Briais and Nestmann define a translation of protocol narrations to processes in the spi calculus, which have been implemented by Briais in the `spyer` compiler[Bri08]. However, the spi calculus contains a fixed set of cryptographic primitives, and protocols that depend on other cryptographic primitive, e.g. homomorphic encryption, are therefore not supported by the compiler.

In this paper we deal with this issue by instead considering the applied pi calculus[AF01] which is an extension to the spi calculus with arbitrary cryptographic primitives. These primitives are described by means of an equational theory. For instance, the equation $\mathsf{dec}(\mathsf{enc}(x, y), y) = x$ describes how symmetric encryption and decryption of a message x with a key y should behave.

We describe how to translate protocol narrations into a version of the applied pi calculus; this method is implemented in OCaml as a tool which acts as a front-end for ProVerif.

In addition to the translation from protocol narrations to processes in the spi calculus[BN07], our work is related to that of projection from global session types to local session types[HYC08] and code generation based on protocol narrations[Mod14].

While the addition of an equational theory for deriving terms is useful for modelling arbitrary cryptographic primitives, they can be the source of high computational complexity. We show that the problem of deriving terms using an equational theory and a set of terms is NP-complete.

Definition 1 (Term Derivation Problem)**.** *Given a signature Σ, an equational theory Θ, a set of terms T, and a term t, can a term be synthesised from Σ and T that equals t in Θ?*

Theorem 1. *The term derivation problem is NP-Complete.*

References

[AF01] Martín Abadi and Cédric Fournet. "Mobile values, new names, and secure communication". In: *ACM SIGPLAN Notices* 36.3 (2001), pp. 104–115.

[BN07] Sébastien Briais and Uwe Nestmann. "A formal semantics for protocol narrations". In: *Theoretical Computer Science* 389.3 (2007), pp. 484–511.

[Bri08] Sébastien Briais. *spyer user's guide*. 2008. URL: `http://sbriais.free.fr/tools/spyer/`.

[HYC08] Kohei Honda, Nobuko Yoshida, and Marco Carbone. "Multiparty asynchronous session types". In: *ACM SIGPLAN Notices* 43.1 (2008), pp. 273–284.

[Mod14] Paolo Modesti. "Efficient Java Code Generation of Security Protocols Specified in AnB/AnBx". In: *Security and Trust Management*. Vol. 8743. Springer, 2014, pp. 204–208.