Foundations of Software Technology and Theoretical Computer Science (Bangalore) 2008.
Editors: R. Hariharan, M. Mukund, V. Vinay; pp 352-356

# About Models of security protocols (abstract)

**Hubert Comon-Lundh**
ENS Cachan and Research Center for Information Security
Advanced Industrial Science and Technology (AIST), Tokyo
h.comon-lundh@aist.go.jp

## Framework

It is clear for everybody that security protocols need to be proven. Indeed, a security breach in one of these distributed programs, may have a dramatic impact. A typical example are the (electronic) voting protocols, which we fear will be soon widely used.

But what does "proven" mean? A formal proof requires a formal model, both for the protocol executions and for the security properties. There are however several such models and it may happen that a protocol is secure in one model and insecure in the other. Furthermore, there is no clear hierarchy in such models as the most accurate ones are not well-suited for formal proofs, because they require much too complicated proof steps, that are always performed in a very sketchy way.

There are two main classes of security models: the first one, later named *symbolic*, has been developed over years by the community of automated theorem proving and concurrency theory [22, 19, 3, 14, 2, 21]. The second one, later named *computational*, is more recent; it is an extension to protocols of the well-developed area of "provable security"[10, 7, 20, 17]. Symbolic models are much simpler, as an attacker is only given a fixed finite set of functionalities and there is no probabilistic choices or complexity issues. They are therefore better suited for (automated) formal proofs. However, because they are simpler, they might miss some attacks that rely on other attacker's capibilities, which are only considered in the computational model.

The relationship between these two classes of protocol models has been investigated in a series of recent works, starting with M. Abadi and Ph. Rogaway [4]. The idea is to explicit under which computational assumptions the symbolic models are *sound*; the soundness theorems show that reasoning in the symbolic model is sufficient: the extra attacker's capabilities are useless for mounting attacks.

Soundness results were first proven in the passive attacker case: such an attacker is not allowed to forge messages nor to send fake messages. Depending on the security primitives and the computational assumptions, there are several soundness results such as [4, 1, 9].

For protocol verification, it is however more relevant to consider an active attacker, who may control the communications. There is then a series of results showing a *trace mapping* property [18, 13, 15]. Roughly speaking, they show that a sequence of events in the computational model is, with overwhelming probability, also represented by a sequence of events in the symbolic model. Such trace mappings are showed for particular primitives and particular symbolic and computational models and assume computational properties



FSTTCS 2008
IARCS Annual Conference on
Foundations of Software Technology and Theoretical Computer Science
http://drops.dagstuhl.de/opus/volltexte/2008/1766

of the security primitives. Furthermore, on the security properties side, these results show that we may reason at the symbolic level, but only for some specific security properties, typically "trace properties".

Another series of soundness results are obtained in a series of papers on *simulatability* [7, 6, 8]. Simulatability implies trace mapping, as pointed out in [5], but the converse might not be true. Roughly speaking, what is missing in trace mapping is the *adaptative soundness* [16].

Finally, an even stronger result (which might be equivalent to simulatability) is the *soundness of observational equivalence* [11]: for a given set of cryptographic primitives, the authors show that trace mapping together with *tree soundness* implies that computational indistinguishability can be soundly abstracted as observational equivalence. This shows in particular, though in a different setting, what is missing in [5], in order to get a converse implication.

### A few remarks on the state of the art

In all these works, the computational attacker is a polynomial time randomized interactive Turing machine. However, as discussed in [8, 17], there are several possible complexity notions for such machines. Also, using a Turing machine model yields often quite sketchy proofs, as the machines (in particular the simulators) are never constructed explicitly. Finally, there is no evidence that worst case polynomial time is an adequate complexity class, though it is convenient because of composition properties. It is actually not clear that Turing machines are an appropriate model. As a conclusion: we would like to abstract from this particular computation model.

On the formal model side, there are many schools, each promoting its own process calculus. There are also issues concerning the expressivity: are the result still valid if we consider protocols with branching tests? recursive protocols? We would like to avoid committing to a particular process calculus, while keeping the features that are essential for security definitions.

Concerning the relationships between the models, is there a general way of defining relationships between models? Is it possible to state something useful, independently of the models considered? Is there a methodology to decompose the tasks when proving a mapping property between two models?

### Models of security protocols

In this paper, mostly consisting of definitions, we revisit the models of security protocols: we show that the symbolic and the computational models (as well as others) are instances of a same generic model. Our definitions are also parametrized by the security primitives, the notion of attacker and, to some extent, the process calculus.

We rely on a set of function symbols, predicate symbols and names (representing any randomized input), which are interpreted in some algebra. This can be a term algebra, or a computational algebra (as defined in [9]), whose domain is the set of bitstrings (or any other first-order structure on the given vocabulary).

A thread (also called a protocol role or a lightweight process) is any sequential program, generating data, receiving inputs from an environment and sending messages to the environment. The operational semantics of threads is defined through predicates that relate two consecutive states and a message (whether received or emitted). Again, this can be interpreted in a symbolic or computational world, depending on the interpretation structure that we consider.

Threads can be composed, using parallel composition (they may run concurrently), replication (a same program may be executed several times), name hiding and external inputs. This yields protocols.

Next, attackers are simply (deterministic) stateful functions that compute a message from a sequence of messages. They could be symbolic or computational and we may impose some restrictions, such as polynomial time computation bounds: this is again a model choice and we do not commit to any one in particular.

In order to define some asymptotic security notions, we need to include in the model families of probability distributions for the interpretation of names. In case of symbolic models, such distributions will be trivial: probabilities of events are either 0 or 1.

Finally, we define *indistinguishability*, without committing to any particular model: this yields for instance static equivalence in the case of a symbolic interpretation. This is also generalized to tree-indistinguishability, for families of term sequences.

### Relationships between models

In this general setting, we define trace mapping: this is a relation between any two interpretations, each of which yielding some notion of possible sequences of events. In both interpretations, the attacker computes fake messages and send them to the network. However, he does not schedule the events according to his computation results. (The attacker is not "adaptative"). Among the models, the symbolic one has a universal property: there is always a trace mapping from the symbolic model to any other model. The converse implication depends however on the assumptions on the function interpretations.

The *tree soundness* property also relates two interpretations $\mathcal{M}_1$ and $\mathcal{M}_2$. This states that, if two trees, labeled with sequences of terms, are indistinguishable in the model $\mathcal{M}_1$, then they are also indistinguishable in the model $\mathcal{M}_2$. For this soundness notion, the attacker is adaptative, but cannot compute his own fake messages: he may only choose among the available directions.

In the most general case, the attacker is both allowed to compute fake messages and to schedule adaptatively the events. In that case, two programs (or protocols) are *observationally equivalent* if there is no attacker that can distinguish them. We show that relating observational equivalence in two models can be reduced to trace mapping and tree soundness in that models:

Trace mapping + Tree soundness $\Rightarrow$ Soundness of observational equivalence

This has been shown in [11], for a particular pair of models and process calculus. In [11] we further proved the trace mapping and the tree soundness for symmetric encryption, under some strong security assumptions.

### Conclusion

We hope that revisiting the definitions will clarify what is relevant. We also believe that trace mapping and tree soundness are two (independent) relevant properties: this could be a guideline when trying to reduce security proofs in some model to symbolic security proofs. As a clue, the computational assumptions are often different for tree soundness and for trace mapping [12].

The main issue now is to decompose further the trace mapping property and the tree soundness property into more elementary tasks. Typically, we would like to get composition results, allowing to merge two sets of function symbols, instead of a having to restart from scratch each time we add a new primitive (which is the case in all current models).

### Acknowledgments

I thank David Nowak for fruitful discussions.

## References

[1] M. Abadi, M. Baudet, and B. Warinschi. Guessing attacks and the computational soundness of static equivalence. In L. Aceto and A. Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 2006.

[2] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, January 2001.

[3] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 148(1), 1999.

[4] M. Abadi and P. Rogaway. Reconciling two views of cryptography: the computational soundness of formal encryption. In *Proc. 1rst IFIP International Conference on Theoretical Computer Science*, volume 1872 of *Lecture Notes in Computer Science*, Sendai, Japan, 2000.

[5] M. Backes, M. Drmuth, and R. Ksters. On simulatability soundness and mapping soundness of symbolic cryptography. In *Proceedings of 27th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, December 2007.

[6] M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable dolev-yao style cryptographic library. In *Proc. IEEE Computer Security Foundations workshop*, 2004.

[7] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proc. 10th ACM Concerence on Computer and Communications Security (CCS'03)*, 2003.

[8] M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability (rsim) framework for asynchronous systems. *Information and Computation*, 205(12), 2007.

[9] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 652–663. Springer, July 2005.

[10] R. Canetti. Universal composable security: a new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[11] H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *Proc. ACM Conf. Computer and Communication Security (CCS)*, 2008.

[12] H. Comon-Lundh, Y. Kawamoto, and H. Sakurada. Symbolic and computational anonymity in an unbounded network. Submitted for publication.

[13] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171, 2005.

[14] F. T. Fabrega, J. Herzog, and J. Guttman. Strand spaces: Proving security protocol correct. *Journal of Computer Security*, 7:191–230, 1999.

[15] R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 172–185. Springer, 2005.

[16] S. Kremer and L. Mazaré. Adaptive soundness of static equivalence. In J. Biskup and J. Lopez, editors, *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS'07)*, volume 4734 of *Lecture Notes in Computer Science*, pages 610–625, Dresden, Germany, Sept. 2007. Springer.

[17] R. Küsters and M. Tuengerthal. Joint state theorems for public-key encryption and digital signature functionalities with local computations. In *Proc. IEEE Computer Security Foundations (CSF'08)*, 2008.

[18] D. Micciancio and B. Warinschi. Soundness of formal encryption in presence of an active attacker. In *Proc. Theory of Cryptography Conference (TCC'04)*, volume 2951 of *LNCS*, 2004.

[19] J. Millen and H. Rueß. Protocol independent secrecy. In *Proc. IEEE Symposium on Security and Privacy*, 2000.

[20] J. Mitchell, A. Ramanathan, and V. Teague. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *Theoretical Comput. Sci.*, 353:118–164, 2006.

[21] A. Roy, A. Datta, A. Derek, J. C. Mitchell, and J.-P. Seifert. Secrecy analysis in protocol composition logic. In *Proc. 11th Asian Computing Science Conference*, volume 4435 of *Lecture Notes in Computer Science*, Tokyo, Japan, Dec. 2006. Springer-Verlag.

[22] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *The Modelling and Analysis of Security Protocols*. Addison Wesley, 2000.