The Protocol Derivation Assistant

Matthias Anlauff[1], Dusko Pavlovic[1], and Stephen Westfold[1]

Kestrel Institute, Palo Alto, CA, USA,
{ma,dusko,westfold}@kestrel.edu,
WWW home page: http://www.kestrel.edu/software/pda

Overview

The Protocol Derivation Assistant or, for short PDA, provides tool support for the derivational approach to protocols as described in [12, 6, 4]. We will give a brief overview of the capabilities of PDA; for further information please see [2]. The design of PDA reflects the basic ideas of the derivational approach to protocol design by providing (i) a rich, graphical user interface for entering protocol derivations, (ii) support for refining models that correspond to these protocols, and (iii) automated support for incrementally proving security properties of the protocols and their models. We will briefly sketch these three aspects in the following.

Protocol Derivations

The research area of security has generated a surprisingly wide range of models and approaches. Even the basic paradigm of security comes in three different flavors: computational (initiated by Diffie and Hellman [7]), information-theoretic (based on Shannon's work [14]), and symbolic (due to Dolev and Yao [8]). Interestingly, the most abstract (and hence the least precise) model is the most recent, and efforts to relate the three paradigms are of an even later date [1, 11]. Since its inception, computer science has been preoccupied with such state machines as Turing machines and automata. Consequently, computations were defined as possible executions, i.e., sequences of actions, and reasoning about computation has been in terms of predicates over such sequences. With the advent of the Internet, and of computer networks in general, this simple paradigm of computation becomes insufficient. The interesting computational processes nowadays do not occur within a state machine, but are distributed through interactions of many state systems. The problem arising from protocol computation is that traditional logical systems are not made to support distributed reasoning, which is its very essence. Our objective is to develop modeling and logical methodologies, tools, and interfaces to facilitate interaction with the novel and unintuitive logical situations of distributed reasoning. To describe a protocol, one usually specifies its process model, its logical properties, and its conceptual components, predecessors, versions, and descendants. The Protocol Derivation Assistant extends over these three dimensions as well. The process model is a version of partially ordered multisets (pomsets) [13], adapted for security, as in [5, 12].

Being implemented as an application on top of the Eclipse platform and its Graphical Editing Framework (GEF), PDA provides a rich set of features that support the developers in their tasks of creating and manipulating protocols and protocol derivations. Protocols are specified using a graphical editing pane by drawing the desired run of the protocol similar to the representation of protocols found in academic research articles. Figure 1 illustrates that using the basic challenge-response protocol: protocol definition given as label "`CR[A,B](c,r)`" specifies the two roles: initiator (`A`) and responder (`B`) as well as the generic challenge and response functions (`c` and `r`, respectively). In general, PDA's input consists of a graphical description of the protocols' processes, and a textual description of the protocols' semantics in form of the specification of basic entities and properties about the protocols. From protocol definitions like this, PDA users can derive more concrete protocols using instance creation, where function parameters (e.g., `c` and `r` in Figure 1) are instantiated with more concrete functions. PDA then automatically generates the protocol graphics with the values for the functions substituted in messages and internal actions. On the semantics side, PDA allows the user to incrementally tackle the security properties for the derived protocols, which makes the process of protocol verifications a manageable task even for more complex protocols. Traditional approaches lack this ability, because they usually try to verify the final protocol as a whole either resulting in proof obligations being too complex to be discharged by any existing theorem provers or forcing the human verifier to abstract away certain technical issues, which in turn can be the very cause of a security leak. For more sophisticated transformations, PDA offers the concept of *rules*, which can be used to specify arbitrary transformations and/or compositions of protocols in order to construct a refined protocol out of already derived or defined ones. This concept of rules is very powerful and has already allowed for expressing crucial transformation steps in the derivations of popular security protocols like GDOI [12], MQV [10], etc. PDA also provides means to keep the protocol derivations organized by allowing the user to split larger derivations into multiple files and by providing a *derivation browser* that displays the structure of the derivations regardless of their division into diagrams.

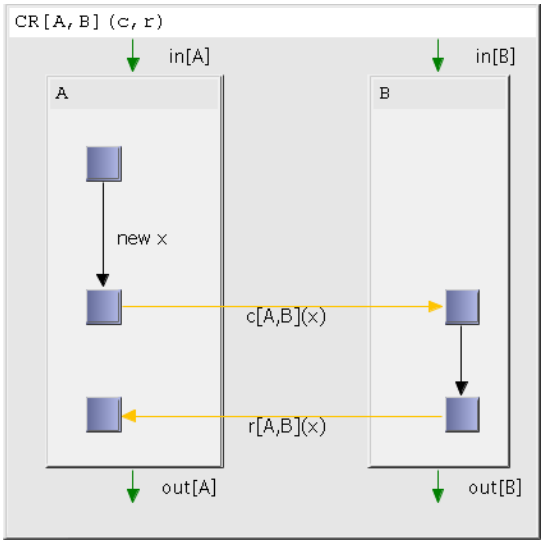

Fig. 1: Specifying a protocol in PDA

Model Refinement and Automated Support

In parallel with the (graphical) protocol derivations, PDA supports the specifications and refinement of protocol models. In order to not dictate a specific logic and interpretation of the protocols, the core PDA system defines interfaces for plugging in arbitrary specification frameworks. However, in its current version, PDA is shipped with a plugin for Kestrel's Specware specification language [9], which provides powerful functionality to support model refinement. In order to make use of the model refinement support in PDA, the user specifies the semantics of functions used in the protocol alongside with basic axioms in the spec-part of the protocol, which is accessible from the graphical user interface. Using this and the process graph defined for the protocol, the Specware-plugin generates proof obligations for authenticity properties. These proof obligations can be discharged (or not) from within the PDA-GUI using prove commands, which trigger calls to the SNARK theorem prover [15] integrated into the Specware system. In case the proof went through, the corresponding conjecture will be transformed into a theorem for refinements of the protocol for which it has been proven. This concept manifests the power of the incremental approach, because in subsequent derivation steps these proofs don't need to be repeated, and thus decrease the complexity of the correctness proofs to a level where there are manageable and in most cases comprehensible. Again, a more detailed description of these features is beyond the scope of this paper; Figure 2 shows a screenshot involving the use of the theorem prover in a protocol derivation.

References

1. Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *J. Cryptol.*, 15(2):103–127, 2002.
2. Matthias Anlauff and Dusko Pavlovic. The Pda Homepage, 2006. URL `http://www.kestrel.edu/software/pda`.
3. Matthias Anlauff, Dusko Pavlovic, and Asuman Suenbuel. Deriving secure network protocols for enterprise services architectures. In *Proceedings of the IEEE International Conference on Communications (ICC 2006), Istanbul*, June 2006.
4. Matthias Anlauff, Dusko Pavlovic, Richard Waldinger, and Stephen Westfold. Proving authenticaton properties in Pda. In *Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis*, 2006. to appear.
5. Iliano Cervesato, Catherine Meadows, and Dusko Pavlovic. An encapsulated authentication logic for reasoning about key distribution protocols. In Joshua Guttman, editor, *Proceedings of the 18th IEEE Computer Security Foundations Workshop*, pages 48–61. IEEE, 2005.
6. Anupam Datta, Ante Derek, John Mitchell, and Dusko Pavlovic. A derivation system and compositional logic for security protocols. *J. of Comp. Security*, 13:423–482, 2005.
7. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
8. Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(2):198–208, 1983.

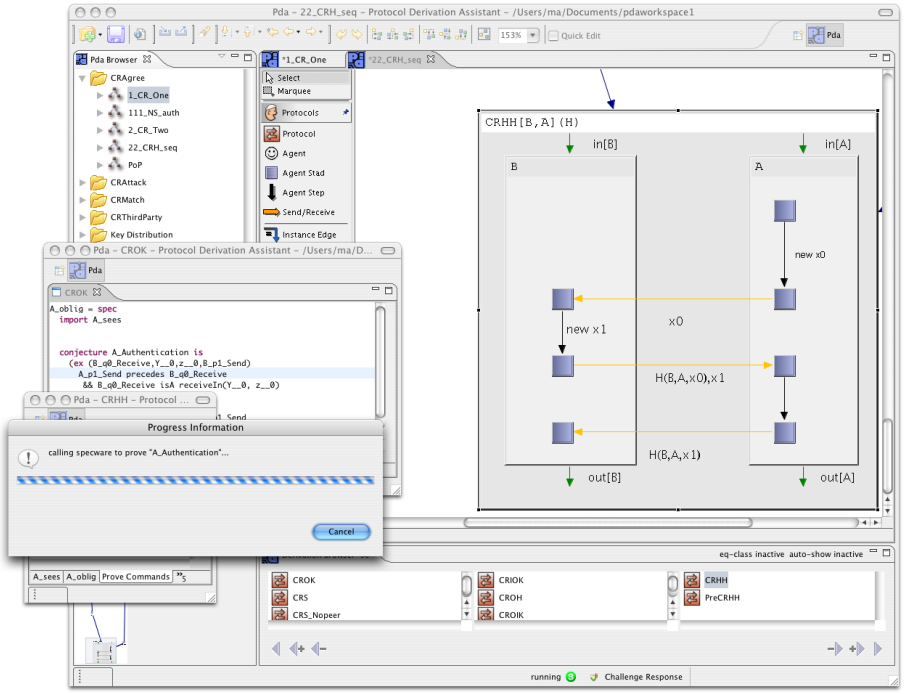

Fig. 2. Running PDA session involving theorem prover invocation

9. Kestrel Institute. *Specware System and Documentation*, 2004. http://www.specware.org/.
10. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement, 1998.
11. Ueli Maurer. Information-theoretic cryptography. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 47–64. Springer-Verlag, August 1999.
12. Catherine Meadows and Dusko Pavlovic. Deriving, attacking and defending the gdoi protocol. In Peter Ryan, Pierangela Samarati, Dieter Gollmann, and Refik Molva, editors, *Proceedings of ESORICS 2004*, volume 3193 of *Lecture Notes in Computer Science*, pages 53–72. Springer Verlag, 2004.
13. Vaughan Pratt. Modelling concurrency with partial orders. *Internat. J. Parallel Programming*, 15:33–71, 1987.
14. Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28:656–715, 1949.
15. M. Stickel, R. Waldinger, and V. Chaudhri. *A guide to* SNARK. SRI International, 2000.