WRLA 2000 Preliminary Version

Applications of Term Rewriting to Cryptographic Protocol Analysis

J. Millen

Computer Science Laboratory
SRI International
Menlo Park, CA 94025
USA
Email: millen@csl.sri.com

Abstract

Cryptographic protocols for key distribution and authentication play an important role in Internet security. Certain flaws in these protocols can be discovered using term rewriting models to express the protocol, the malicious network environment, and vulnerability search strategies.

1 Cryptographic Protocol Security

Cryptographic protocols are short, simple exchanges of messages whose purpose is to distribute secret keys to communicating parties. These keys can be used to transmit confidential information or to authenticate messages.

For example, consider the following protocol:

```
A -> B: {A,Na}pk(B)
B -> A: {Na,Nb}pk(A)
A -> B: {Nb}pk(B)
```

Each line, of the form $A \to B : M$, means that A sends the message M to B, and each one changes state as a result. The expression {A,Na}pk(B) represents the encryption of the concatenated fields A and Na with the public key of B. N_a and N_b are session-specific values called "nonces" that are, in this protocol, kept secret and could be used later as session keys.

Specifications of this type are called "Alice-and-Bob" specifications because of the names conventionally given to the participants A and B in examples. Note, however, that an actual protocol user, like Alice, can start any

[1] Supported in part by DARPA through SPAWAR Contract N66001-00-C-8014.

This is a preliminary version. The final version will be published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

number of processes, and some of them can play the A role while others play the B role.

This protocol is a portion of a longer one published in 1978; it was found to be insecure in 1996 [8]. It is possible for a third party ("Mallory") to masquerade as Alice to Bob, while carrying on a concurrent session of the protocol (as Mallory in the B role) with Alice. The vulnerability can be discovered from the abstract presentation above, given only an idealized notion of encryption and a simple attacker model, and all of these can be represented using term rewriting rules.

The security properties most often stated and analyzed for cryptographic protocols are secrecy and authentication. Session keys and user data should be kept secret, and the recipient of a session key or important data (say, a contract) should be able to confirm the identity of the sender. These are safety properties, i.e., state invariants. Some protocols have more complex security goals, but there is no consensus on how to state them. Liveness properties are usually considered performance rather than security properties.

2 Protocol Modeling

The earliest mathematical model of a class of cryptographic protocols, by Dolev and Yao [7], was a rewrite rule model. That paper studied "ping-pong" protocols between a pair of legitimate parties in which the first message is an encrypted secret, and each party receiving a message applies a state-dependent sequence of public key operators to it and returns the result. State transition rules might, for example, rewrite a received message x to $E_B(D_A(x))$, where E_A means encryption with A's public key. There are reduction rules like $D_x(E_x(y)) \rightarrow y$. Honest parties have to follow the procotol, while a malicious party Z could intercept any message, apply any combination of E_x for any x and the malicious party's decryption operator D_Z, and forward the result to either party in any session. The attack objective was to expose the secret that was sent encrypted in the first message. The security problem is decidable for this very restricted model and also for minor extensions of it.

The term "Dolev-Yao" model is now often used to refer to any protocol model in which the attacker can intercept and redirect messages, and modify their content using axiomatically defined operations and data available to the attacker.

To make the cryptographic protocol problem undecidable, we need only assume, first, that two terms can be concatenated into a longer message, and, second, that there is an unbounded set of items of some type, usually called "nonces." Nonces serve as unique identifiers to distinguish different sessions [3].

There has been some effort to identify restricted classes of protocols for which it is sufficient to examine a bounded number of sessions. For example, there is Lowe's "small system" result, in which only one instance of each role in the procotol is enough to reveal possible flaws [9]. But the conditions for this result are very restrictive. State exploration approaches generally perform a bounded, iteratively expanding search.

Mitchell's recent multiset rewrite rule (MSR) model [3], inspired by linear logic, is both simple and general enough for extension to real protocols. In the MSR model, a system state is a finite collection of "facts," which are ground terms of three kinds: states of protocol parties, such as $A_0(a, b)$ (A is in state 0 of a protocol between a and b); messages transmitted to the network, such as $N(\{a, b\})$; and message components memorized by the attacker, such as $M(a)$.

A protocol rule shows how states are updated and messages are produced, for example

$$A_0(A, B) \to (\exists K)A_1(A, B, K), N(\{A\}_K),$$

where the quantifier has the special meaning that the key variable K is to be instantiated with a nonce. A state fact in the system state multiset that matches the left side is removed and replaced by the similarly instantiated right side. (The system state is a multiset because certain rules, such as session initialization or message creation rules, permit any number of copies of some terms to be generated.)

The model also includes attacker rules, by which, for example, the attacker can decrypt encrypted terms and add the result to its memory, such as

$$M(\{A\}_K), M(K) \to M(\{A\}_K), M(K), M(A).$$

Note that the attacker model is fixed and universal, given the selection of encryption and other computational functions used by the protocol.

There are other models, such as specialized epistemic authentication logics [2] and process algebra representations [15,1], but they have less of a term rewriting flavor.

3 Protocol Analysis

There are techniques for proving that cryptographic protocols satisfy security properties. An influential paper by Paulson demonstrated an inductive technique to prove confidentiality or authentication invariants [14], and there have been others, using models similar in spirit to the MSR model. Proofs using authentication logics are useful for understanding protocols, but they work by "idealizing" the protocol into logical statements in a way that is not algorithmic and which loses some information kept by state-transition models.

Proving protocol security is most useful when the protocol is in fact secure. If the protocol has a vulnerability, as most do, at least in their initial versions, it is productive to implement some form of state search.

3.1 Using Prolog

The oldest approaches to cryptographic protocol analysis use Prolog. It is easy to express the state of a protocol party as a Prolog term, and a message is also a term, so that the global state of the environment is a list of such states and messages. A global state transition updates some party's state and may add a new message.

The Interrogator, a Prolog program for protocol analysis [12], was a set of Prolog rules expressing a relation `reachable(H,Q)` meaning that global state Q was reachable via the message sequence H. In a typical reachability query, Q was partly instantiated to an insecure state, and H was a variable, to be instantiated by a successful Prolog seach for an attack. Prolog's built-in resolution algorithm performs backward search from the insecure goal state to the initial state. The challenge for this approach was termination, not only because of the inherent undecidability, but because of the need to rein in Prolog's depth-first search. A partial answer was to make the program interactive, to take advantage of user guidance.

One of the improvements introduced by Meadows' NRL Protocol Analyzer [11], also a Prolog program, was the explicit use of narrowing. Narrowing solves equations modulo canonical sets of rewrite rules; and the rules for reducing symbolic encryption, such as $\mathtt{dec}(K,\mathtt{enc}(K,X)) = X$, are canonical.

One way to use narrowing, adopted also in later versions of the Interrogator, is to find predecessor states. Suppose the protocol includes a transition rule $Q, M \rightarrow Q'$, meaning that there can be a network state transition from Q to Q' when message M is received; and the attacker's goal is the partly instantiated state Q_a. Now suppose narrowing succeeds in solving $Q' = Q_a$ with the substitution σ. Then the attacker's job is reduced to reaching the state σQ and generating the message σM.

The use of narrowing brings to light another challenge, when the protocol makes use of non-canonical operations, such as commutative and associative operations. For example, bitwise modulo-two addition (exclusive-or) is used in many protocols as an encryption operator.

3.2 Application of Model-Checking Tools

There have been many recent successess in the application of existing model checking tools, which had been developed for other purposes such as processor "hardware" design, to cryptographic protocol analysis. The stimulus was a paper by Roscoe on the use of FDR [15], followed by Lowe's result on the Needham-Schroeder protocol [8]. Other researchers realized that similar techniques could be used with their own tools [10,13]. The combination of modern computing power, model checking advances such as BDDs, and a few protocol-specific optimizations, made it possible to find previously undiscovered vulnerabilities in a reasonable execution time, with no user interaction after the initial encoding and setup.

The lessons learned about modelling and search optimizations could then be applied in other ways. One remarkable result is the ability of Maude, a rewriting logic system rather than a model checker, to become a model checker. Using meta-level excution strategies, it performs iterative breadth-first state search and takes advantage of state reduction optimizations suitable for the protocol problem. In this way it has achieved performance comparable to model checking tools [5,4].

It is also possible to program a new model checker that is customized for the protocol problem. An unusual approach to this was taken by Athena [16], using the strand space model [17]. In this model, the state of the network is a single structure called a *bundle*, consisting of nodes that are partially ordered by causality. A *strand* is a string of adjacent nodes ordered by execution sequence from the same process. A protocol specifies possible legal strands with *parameterized* strands, i.e., strand expressions with variable parameters, and these can be connected into *semibundles* by joining transmitted and received messages. Athena attempts to expand a small semibundle containing a security violation into a complete bundle. It is efficient because of its use of partial ordering and because the state expansion introduces an entire strand at a time. A semibundle could be regarded as a large complex term. It would be interesting to see if a general term-rewriting system could be adapted to follow this approach.

4 Conclusions

There are natural and practical ways to apply term rewriting concepts and tools to the security analysis of cryptographic protocols. Term rewriting can also be applied to this area in other ways beyond the specific ones mentioned above, such as to implement part of a translator from a high level protocol language to a rule representation, as was done using Maude in the CAPSL translator [6]. We might also look forward to benefits from advances in related areas such as narrowing algorithms, and perhaps from extensions such as graph rewriting.

References

[1] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the Spi calculus. Technical Report SRC-149, Digital Systems Research Center, 1998.

[2] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[3] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *12th IEEE Computer Security Foundations Workshop*, pages 55–69. IEEE Computer Society, 1999.

[4] G. Denker. Design of a CIL connector to Maude. In H. V. andN. Heintze and E. Clarke, editors, *Workshop on Formal Methods and Comuter Security*. Carnegie Mellon University, July 2000.

[5] G. Denker, J. Meseguer, and C. Talcott. Protocol specification and analysis in Maude. In *Formal Methods and Security Protocols*, 1998. LICS '98 Workshop.

[6] G. Denker and J. Millen. CAPSL integrated protocol environment. In *DARPA Information Survivability Conference (DISCEX 2000)*, pages 207–221. IEEE Computer Society, 2000.

[7] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29:198–208, 1983. Also STAN-CS-81-854, May 1981, Stanford U.

[8] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of TACAS*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.

[9] G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(2/3):89–146, 1999.

[10] W. Marrero, E. Clarke, and S. Jha. Model checking for security protocols. In *DIMACS Workshop on Design and Verification of Security Protocols*. Rutgers U., 1997.

[11] C. Meadows. A system for the specification and verification of key management protocols. In *IEEE Symposium on Security and Privacy*, pages 182–195. IEEE Computer Society, 1991.

[12] J. Millen, S. Clark, and S. Freedman. The Interrogator: protocol security analysis. *IEEE Transactions on Software Engineering*, SE-13(2):274–288, February 1987.

[13] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murϕ. In *IEEE Symposium on Security and Privacy*, pages 141–154. IEEE Computer Society, 1997.

[14] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85–128, 1998.

[15] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *8th IEEE Computer Security Foundations Workshop*, pages 98–107. IEEE Computer Society, 1995.

[16] D. Song. Athena: a new efficient automatic checker for security protocol analysis. In *12th IEEE Computer Security Foundations Workshop*, pages 192–202. IEEE Computer Society, 1999.

[17] J. Thayer, J. Herzog, and J. Guttman. Honest ideals on strand spaces. In *11th IEEE Computer Security Foundations Workshop*, pages 66–78. IEEE Computer Society, 1998.