Extending Formal Cryptographic Protocol Analysis Techniques for Group Protocols and Low-Level Cryptographic Primitives

Catherine Meadows
Center for High Assurance Computer Systems
Code 5543
Naval Research Laboratory
Washington, DC 20375
meadows@itd.nrl.navy.mil

We have recently seen the development of a number of new tools for the analysis of cryptographic protocols. Many of them are based on state exploration, that is, they try to find as many paths through the protocol as possible, in the hope that, if there is an error, it will be discovered. But, since the search space offered by a cryptographic protocol is infinite, this search alone cannot guarantee security if no attack is found. However, some state exploration tools do offer the ability to prove security results as well as find flaws by the use of theoretical results about the system that they are examining. In particular, the NRL Protocol Analyzer [4] allows its user to interactively prove lemmas that limit the size of its search space. If the resulting search space is finite, then it too can guarantee that a protocol is secure by performing an exhaustive search.

However, the ability to make such guarantees brings with it certain limitations. In particular, most of the systems developed so far model only a very limited set of cryptographic primitives, often only encryption (public and shared key) and concatenation. They also avoid low-level features of cryptographic algorithms, such as the commutativity and distributivity properties of RSA.

Most importantly, there has as yet been no attempt to use state exploration techniques to reason about group protocols, that is, protocols that involve communication among a set of principals which could be arbitrarily large. The reason why this is the case is not hard to see; in a group protocol even a legal execution can take an unbounded number of steps. This makes state exploration and the avoidance of state explosion even more difficult. Thus it is no surprise that so far, the only successful formal verification of group cryptographic protocols that has been done to date has been done involving the use of theorem provers [5, 2]. However, although theorem provers can give the assurance that a protocol is correct, they do not give the same assistance in pinpointing problems as the direct generation of counterexamples.

In our work we are seeking to fill this gap by applying the NRL Protocol Analyzer to the problem of verifying group security protocols and protocols that use other cryptographic primitives than those used in the standard model. Since the NRL Protocol Analyzer already includes a limited theorem-proving capability, we believe that it is well-suited for scaling up to these types of protocols by extending the power and scope of that capability. Once this is done, we can use the NRL Protocol Analyzer as before to explore the remaining search space.

We have recently started an analysis of the suite of protocols developed for the CLIQUES project [7]. These are a set of group key distribution protocols that use a version of the Diffie-Hellman key exchange protocol. Moreover, it includes not only the standard operations associated with Diffie-Hellman, but also the concepts of repeated exponentiation and exponentiation by an inverse, so that it requires a richer model than that we have used previously.

We are performing an analysis of the A-GDH.2 protocol from [1]. This is a protocol in which a key is shared among a group of arbitrary size. We chose this because it is the basis of all the protocols used in the CLIQUES project, and because its simplicity makes it a useful subject of experimentation. This protocol guarantees secrecy in the sense that, if no member of the group is dishonest, then the key is not compromised. Our approach was first to make some minor changes to the NRL Protocol Analyzer that made the specification and compilation of group protocols possible. We then proved some initial lemmas that were defined and proved pretty much in the same way as in two or three-party protocols.

Finally, we set out to prove a theorem stating that, if a principal accepts a key K as a group key, and all the principals in the group are honest, then the key is not compromised. In order to do this, we found out that we had to devise a new construct which we call a *parameterized language*.

Briefly, a *language* is a construct used by the NRL Protocol Analyzer to define invariants [3]. Languages are so-called because they are defined using a BNF grammar, but actually most languages as they are generated by the NRL Protocol Analyzer can be defined is the smallest set $\mathbf{L}$ containing a fixed set S that is closed under a set of operations by the intruder involving members of $\mathbf{L}$ and arbitrary terms. One uses $\mathbf{L}$ to prove secrecy results by showing that the intruder's not knowing members of $\mathbf{L}$ is invariant under state transition.

It can be seen that languages are closely related to strand space ideals [8] and to rank functions [6]. Languages are more general than ideals in that the set of operations available to the intruder is left open and can be defined in the specification itself, while in strand space ideals the operations are restricted to encryptions and concatenation. On the other hand, ideals are more general than languages in that is possible to restrict the keys used for the encryption operation to the members of a particular set, which is not possible for NRL Protocol Analyzer languages. Parameterized languages put conditions, not only on the keys used for encryption and the messages encrypted, but on any input to any operation defined in the language. Thus parameterized languages can be thought of as a generalization of both NRL Protocol Analyzer languages and strand space ideals. We find that the fact that the analysis of the A-GDH.2, at least so far, seems to require only this relatively straightforward generalization of languages an encouraging one.

We are currently continuing our verification of A-GDH.2, and plan to expand our investigation to all the CLIQUES protocols. We expect to continue refining and extending the use of parameterized languages and other tools as our analysis progresses.

References

[1] G. Ateniese, M. Steiner, and G. Tsudik. Authenticated group key agreement and friends. In *ACM Conference on Computer and Communications Security*. ACM, November 1998.

[2] J. Bryans and S. Schneider. CSP, PVS, and a recursive authentication protocol. In *DIMACS Workshop on Formal Verification of Security Protocols*, September 1997.

[3] C. Meadows. Language generation and verification in the NRL Protocol Analyzer. In *Proceedings of the 9th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1996.

[4] C. Meadows. The NRL Protocol Analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[5] L. C. Paulson. Mechanized proofs for a recursive authentication protocol. In *Proceedings of the 10th Computer Security Foundations Workshop*, pages 84–95. IEEE Computer Society Press, June 1997.

[6] S. Schneider. Verifying authentication protocols with CSP. In *Proceedings of the 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1997.

[7] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A new approach to key agreement. In *IEEE International Conference on Distributed Computing Systems*. IEEE Computer Society Press, May 1998.

[8] J. Thayer, J. Herzog, and J. Guttman. Honest ideals in strand spaces. In *Proceedings of the 11th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1998.