Brief Announcement: Lower and Upper Bounds for Attacks on Authentication Protocols

Scott D. Stoller
Computer Science Dept., Indiana University, Bloomington, IN 47405-7104 USA
stoller@cs.indiana.edu   http://www.cs.indiana.edu/~stoller/

Many authentication protocols are intended to work correctly in the presence of an adversary that can intercept messages, perform an unbounded number of encryptions and other operations while fabricating messages, and prompt honest principals to engage in an unbounded number of concurrent (*i.e.*, interleaved) runs of the protocol. The amount of local state maintained by a single run of an authentication protocol is bounded. This suggests the existence of upper bounds on the resources needed to attack a protocol. Such bounds provide a rigorous basis for automated verification. We sketch a Language for Authentication Protocols (LAP), based on [WL93], and establish an exponential lower bound on the worst-case number of concurrent runs needed in a successful attack on a LAP protocol. Details appear in [Sto98a]. An exponential upper bound would be too large to enable automated verification. This shows the need to impose additional restrictions on the class of protocols, as done in [Sto98b], which gives a polynomial upper bound.

The relevant kinds of statements (slightly simplified) in LAP are: NewValue(v), which generates a unique value (*e.g.*, a nonce or session key) and binds variable v to it; Send(x, t), which sends a message t to x; and Receive(pat), which receives a message m and binds the unbound variables in pattern pat to the corresponding subterms of m. The Receive statement attempts pattern-matching between a candidate message m and the pattern. A pattern can express that the message should be a ciphertext produced with a given key. If m is encrypted with the given key (if any) and there exist bindings for the unbound variables of pat such that pat with those bindings equals m, then the Receive statement executes and establishes those bindings. The Receive statement blocks until this condition is satisfied. A *local protocol* is a finite sequence of statements satisfying some well-formedness requirements. A *protocol* is, roughly, a set of local protocols, one for each role (or participant) in the protocol. A *secrecy* requirement asserts that certain values are not revealed to the adversary.

Theorem 1. There exists a family of LAP protocols Π^ℓ and a secrecy property ϕ such that the minimum number of concurrent runs in an execution of Π^ℓ that violates ϕ is $\Omega((\ell/2-4)^{(\ell/2-4)})$, where ℓ is the maximum number of Send statements in a local protocol of Π^ℓ.

Proof sketch: Protocol Π^ℓ involves three local protocols: P_I, P_R, and P_S. Intuitively, an execution of Π^ℓ performs two depth-first traversals of a conceptual ℓ-ary tree of height ℓ before violating ϕ. Each non-leaf node of the tree corresponds to a run of a local protocol. A run of P_I corresponds to the root. Runs of P_R correspond to non-root non-leaf nodes. Runs of P_S correspond to leaves. The protocol involves *two* depth-first traversals in order to force all the runs of P_R to be concurrent. Values generated by NewValue are used to ensure that each node in the tree corresponds to a distinct run. By design, the secrecy requirement ϕ is violated iff P_I runs to completion, and P_I can do this only in executions containing $\Omega((\ell/2-4)^{(\ell/2-4)})$ concurrent runs of P_R. ∎

References

[Sto98a] Scott D. Stoller. Justifying finite resources for adversaries in automated analysis of authentication protocols. Tech Report 506, C. S. Dept., Indiana U., March 1998 (revised Feb. 1999).

[Sto98b] Scott D. Stoller. Reductions for automated analysis of authentication protocols. Tech Report 520, C.S. Dept., Indiana U., Dec. 1998.

[WL93] Thomas Y. C. Woo and Simon S. Lam. A semantic model for authentication protocols. In *Proc. 14th IEEE Symposium on Research in Security and Privacy*, pages 178–194, 1993.