Poster: A Symbolic Logic with Concrete Bounds for Cryptographic Protocols

Anupam Datta*, Joseph Y. Halpern†, John C. Mitchell‡, Arnab Roy§ and Shayak Sen*
*Carnegie Mellon University, Pittsburgh, USA
{danupam,shayaks}@cmu.edu
†Cornell University, Ithaca, USA
halpern@cornell.edu
‡Stanford University, Stanford, USA
mitchell@stanford.edu
§Fujitsu Research, Sunnyvale, USA
arnabr@gmail.com

I. Introduction

Large and complex cryptographic protocols form the backbone of internet security today. A lot of attention has focused on developing formal reasoning principles for proving the correctness of cryptographic protocols ([1], [2], [3], [4]). Formal techniques for proving correctness of cryptographic protocols have largely focused on the asymptotic computational model, where security guarantees are violated with atmost some negligible probability, given that keys chosen are long enough, and the adversary's runtime is bounded by some polynomial. A concrete probability, key length or polynomial is not specified. However, at the scale of the internet, protocols use keys to communicate with millions of agents over extended periods of time, with increasingly powerful adversaries attempting to break them. In this scenario, evaluating security in a quantitative sense with respect to key lengths and adversary runtimes is critical. Concrete security is a practice-oriented approach to cryptography that requires that proofs of a security property be accompanied by a concrete probability with which the property is false.

In this paper, we present Quantitative Protocol Composition Logic (QPCL), a new program logic for reasoning about concrete bounds of the security of cryptographic protocols. QPCL allows reasoning about concrete probability bounds in a computational model of cryptography. A typical assertion in QPCL is a concrete security statement of the form $B^{\epsilon}(\varphi)$, where φ is temporal trace formula that represents a security property of the execution of a protocol in the presence of probabilistic polynomial time adversary, and ϵ is a probability that is a function of η, the the security parameter governing the length of keys and nonces, and t, the concrete runtime of the adversary.[1] This assertion is true if φ holds with probability at least $1-\epsilon(\eta,t)$. Typically, ϵ also depends on functions that represent the probability of an adversary violating the security of the specific cryptographic primitives used in a protocol.

QPCL assertions are interpreted over traces generated by a simple probabilistic, concurrent programming language executing in the presence of an adversary. The operational semantics of the programming language allows reasoning about concrete probabilities and runtimes of traces. Importantly, the adversary's program is allowed to be any program that runs in polynomial time, therefore allowing the soundness proofs of our logic to be valid in a computational model.

Rules for reasoning in QPCL's proof system can be broadly categorized into the following forms: (1) axioms about cryptographic primitives, (2) invariants and postconditions of protocol programs, (3) first-order probabilistic belief assertions, and (4) temporal ordering of events.

QPCL axioms about cryptographic primitives such as signature schemes and pseudorandom-number generators introduce a probabilistic error bound. These axioms have formal soundness proofs that show whenever an axiom is false on a trace, an attack on the corresponding primitive can be constructed. Therefore, these probabilistic error bounds are closely related to the security of the underlying primitive.

To prove invariants and postconditions of protocol programs, we borrow the syntax and style of reasoning from Protocol Composition Logic (PCL) [5], a formal logic for stating and proving security properties of network protocols, which has been used to prove properties of protocols such as SSL/TLS, IEEE 802.11i (WPA2), and extensions of Kerberos. The Hoare Logic style assertion $\varphi_1[P]_X\varphi_2$ is used to state pre/post-conditions about program fragments. Additionally, we have a rule to lift local reasoning about program fragments to global invariants about traces. An interesting consequence of placing concrete runtime bounds on traces is that all programs terminate when the runtime bound has elapsed, so global invariants can never mention events in the future, resulting in all assertions provable in QPCL being safety properties.

We reason about probabilistic belief in QPCL using an approach introduced by Halpern [6], which in turn is based on the *ϵ-semantics* of Goldszmidt, Morris, and Pearl [7]. Halpern's logic has a binary operator $\rightarrow^{\epsilon}$, where, roughly speaking, $\psi \rightarrow^{\epsilon} \varphi$ means that the conditional probability of φ given ψ is at least $1-\epsilon$. $B^{\epsilon}(\varphi)$ is an abbreviation of *true* $\rightarrow^{\epsilon} \varphi$. In our setting, the uncertainty ϵ is not a constant,

[1] ϵ has other parameters as well, which we omit here.

but a function of the security parameter η, the adversary's runtime t. Traces generated by protocol programs serve as an instance of the more general semantic models of Halpern, allowing us to embed the sound and complete proof system from [6] into our logic.

We illustrate the use of the QPCL reasoning principles by proving a "matching-conversations" property for a simple initiator-responder protocol, which states that the precise order in which messages are sent and received by the participants of the protocol cannot be altered by an adversary.

II. Overview of the Formal System

QPCL is a probabilistic program logic with concrete security bounds for temporal trace properties of cryptographic protocols. A concrete security bound on an assertion in our logic expresses the probability with which a computationally constrained adversary can violate the assertion in terms of the length of keys, the runtime of the adversary and the number of concurrent sessions of the protocol.

A. Programming Model

The models on which the logic is interpreted are traces generated by a simple protocol programming language executing in the presence of an adversary. The requirements of the logic neccessitate some features of the operational semantics of the protocol programming language, as the operational semantics defines how traces are generated from protocol programs interacting in the presence of an adversary. We now motivate some of these features.

- *Concrete Runtime.* Reasoning about concrete runtime requires accounting for the runtime of every computation in an execution. This runtime includes that of the adversary as well protocol programs. The operational semantics is defined by a transition system where transitions are labeled by these runtime costs.
- *Concrete Probabilities.* Computing probabilistic bounds requires considering probabilities of sets of traces. The transition system is probabilistic and the probabilistic branches induce an execution tree, where each path in the tree is an execution trace. We use the tree to compute the relevant probabilities.
- *Computational completeness.* For proofs to be valid in a computational model, we cannot restrict the adversary to a symbolic model. However, modeling the adversary explicitly in a Turing complete language makes the language very complicated. For proofs of security, which are generally proofs via reduction to known hard problems, it is possible to ignore the representation of the adversary. Therefore, we do not explicitly restrict the adversary's program to a particular language, but reason only about its input-output behavior.
- *Concurrency and Adversarial Scheduling.* Programs execute concurrently; where the scheduler that switches between programs is controlled by the adversary. The scheduler is resource-constrained as well. The cost of the computation required by the adversary to make scheduling decision is accounted for in the runtime of a trace.

B. Logic and Proof System

QPCL allows declarative specifications of temporal trace properties, and the proof system supports reasoning in a manner very similar to first-order reasoning. Essentially, axioms related to cryptography have error bounds associated with them, and these error bounds are propagated throughout the proof to derive a bound on the final assertion. To aid reasoning about cryptographic protocols, QPCL supports the following features:

- *Temporal Trace Properties.* Protocols, such as those involving authentication, often require properties that specify the precise order in which certain events occur. To allow reasoning about properties such as message ordering, QPCL supports temporal operators. Traditionally, such properties are specified using games [1], which imply that the security proofs typically involve the harder task of proving equivalence of games.
- *Concrete Security Bounds.* Concrete security bounds on assertions in QPCL depend on the protocol being reasoned about in a number of ways. First, protocols specify the implementations of cryptographic primitives to be used; these define the concrete bounds on the axioms related to cryptography. Second, concrete bounds on the cryptographic primitives usually also depend on the number of queries made to the primitive, and to bound this number on any particular trace, we use the number of times the cryptographic primitive is called in a protocol.
- *Safety Properties.* Finally, since concrete security involves computing the probability of a property being violated as a function of the time available to the adversary, the total runtime of a trace is bounded; traces end when the total time elapsed is greater than the specified time bound. Thus, we cannot talk about liveness properties (i.e., things that will happen eventually happen) in QPCL.

References

[1] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Advances in Cryprtology - Crypto '93 Proceedings*. Springer-Verlag, 1994, pp. 232–249.

[2] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols," in *Proceedings of 30th Annual Symposium on the Theory of Computing*. ACM, 1998, pp. 419–428.

[3] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Science*, vol. 28, pp. 270–299, 1984.

[4] V. Shoup, "On formal models for secure key exchange (version 4)," IBM Research, Tech. Rep. RZ 3120, 1999.

[5] A. Datta, J. Mitchell, A. Roy, and S. Stiller, "Protocol composition logic," In V. Cortier and S. Kremer (Editors), Formal Models and Techniques for Analyzing Security Protocols, 2010, to be published by IOS Press. http://www.andrew.cmu.edu/user/danupam/dmrs-pcl2010.pdf.

[6] J. Y. Halpern, "From qualitative to quantitative proofs of security properties using first-order conditional logic," in *Proc. Twenty-Third National Conference on Artificial Intelligence (AAAI '05)*, 2008, pp. 454–459.

[7] M. Goldszmidt, P. Morris, and J. Pearl, "A maximum entropy approach to nonmonotonic reasoning," *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 15, no. 3, pp. 220–232, 1993.