Probabilistic Polynomial-Time Process Calculus and Security Protocol Analysis

John C. Mitchell

Stanford University
Stanford, CA 94305
http://www.stanford.edu/~jcm

Abstract. We propose a formal framework for analyzing security protocols. This framework, which differs from previous logical methods based on the Dolev-Yao model, is based on a process calculus that captures probabilistic polynomial time. Protocols are written in a restricted form of π-calculus and security is expressed as a form or *observational equivalence,* a standard relation from programming language theory that involves quantifying over possible additional processes that might interact with the protocol. Using an asymptotic notion of probabilistic equivalence, we may relate observational equivalence to polynomial-time statistical tests. Several example protocols have been analyzed. We believe that this framework offers the potential to codify and automate realistic forms of protocol analysis. In addition, our work raises some foundational problems for reasoning about probabilistic programs and systems.

1 Summary

This invited lecture for ESOP ′01 will describe an approach to security protocol analysis based on a probabilistic polynomial-time process calculus and asymptotic observational equivalence. The work has been carried out in collaboration with P. Lincoln, M. Mitchell, A. Scedrov, A. Ramanathan, and V. Teague. Some of the basic ideas are described in [LMMS98], with a description of a simplified form of the process calculus appearing in [MMS98] and further example protocols considered in [LMMS99]. The closest technical precursor is the Abadi and Gordon spi-calculus [AG99,AG98] which uses observational equivalence and channel abstraction but does not involve probability or computational complexity bounds; subsequent related work is cited in [AF01], for example. Prior work on CSP and security protocols, e.g., [Ros95,Sch96], also uses process calculus and security specifications in the form of equivalence or related approximation orderings on processes. Slides from this talk will be available on the author's web site at http://www.stanford.edu/~jcm.

2 Protocols

Protocols based on cryptographic primitives are commonly used to protect access to computer systems and to protect transactions over the Internet. Two well-known examples are the Kerberos authentication scheme [KNT94,KN93], used to manage encrypted passwords, and the Secure Sockets Layer [FKK96], used by Internet browsers and servers to carry out secure internet transactions. In recent years, a variety of methods have developed for analyzing and reasoning about such protocols. These approaches include specialized logics such as BAN logic [BAN89], special-purpose tools designed for cryptographic protocol analysis [KMM94], and theorem proving [Pau97a,Pau97b] and model-checking methods using general purpose tools [Low96,Mea96,MMS97,Ros95,Sch96].

Although there are many differences among existing formal approaches, most use the same basic model of adversary capabilities. This model, apparently derived from [DY83] and views expressed in [NS78], treats cryptographic operations as “black-box” primitives. For example, encryption is generally considered a primitive operation, with plaintext and ciphertext treated as atomic data that cannot be decomposed into sequences of bits. In most uses of this model, as explained in [MMS97,Pau97a,Sch96], there are specific rules for how an adversary can learn new information. For example, if the decryption key is sent over the network “in the clear”, it can be learned by the adversary. However, it is not possible for the adversary to learn the plaintext of an encrypted message unless the entire decryption key has already been learned. Generally, the adversary is treated as a nondeterministic process that may attempt any possible attack, and a protocol is considered secure if no possible interleaving of actions results in a security breach. The two basic assumptions of this model, perfect cryptography and nondeterministic adversary, provide an idealized setting in which protocol analysis becomes relatively tractable.

While there have been significant accomplishments using this model, the assumptions inherent in the standard model also make it possible to “verify” protocols that are in fact susceptible to attack. For example, the model does not allow the adversary to learn a decryption key by guessing, since then some nondeterministic execution would allow a correct guess, and all protocols relying on encryption would be broken. However, in some real cases, adversaries can learn some bits of a key by statistical analysis, and can then exhaustively search the remaining (smaller) portion of the key space. Such an attack is simply not considered by the model described above, since it requires both knowledge of the particular encryption function involved and also the use of probabilistic methods.

Our goal is to develop an analysis framework that can be used to explore interactions between protocols and cryptographic primitives. We are also interested in devising specifications of cryptographic primitives such as oblivious transfer and selective decommittment. Our framework uses a language for defining communicating probabilistic polynomial-time processes [MMS98]. We restrict processes to probabilistic polynomial time since the adversary is represented by an arbitrary context, written in the process calculus. Limiting the running time of an adversary allows us to lift other restrictions on the behavior of an adversary. Specifically, an adversary may send randomly chosen messages, or perform arbitrary probabilistic polynomial-time computation on messages overheard on the network. In addition, we treat messages as sequences of bits and allow specific encryption functions such as RSA or DES to be written in full as part of a protocol. An important feature of this framework is that we can analyze probabilistic as well as deterministic encryption functions and protocols. Without a probabilistic framework, it would not be possible to analyze an encryption function such as ElGamal [ElG85], for example, for which a single plaintext may have more than one ciphertext.

Security properties of a protocol P may be formulated by writing an idealized protocol Q so that, intuitively, for any adversary M, the interactions between M and P have the same observable behavior as the interactions between M and Q. This intuitive description may be formalized by using observational equivalence (also called observational congruence), a standard notion from the study of programming languages. Namely, two processes (such as two protocols) P and Q are observationally equivalent, written $P \simeq Q$, if any program $\mathcal{C}[P]$ containing P has the same observable behavior as the program $\mathcal{C}[Q]$ with Q replacing P. The reason observational equivalence is applicable to security analysis is that it involves quantifying over all possible adversaries, represented by the environments, that might interact with the protocol participants. In our asymptotic formulation, observational equivalence between probabilistic polynomial-time processes coincides with the traditional notion of indistinguishability by polynomial-time statistical tests [Lub96,Yao82], a standard way of characterizing cryptographically strong pseudo-random number generators.

The remainder of this short document presents the key definitions, as reference for the author's invited talk.

3 Process Calculus

The protocol language consists of a set of *terms,* or sequential expressions that do not perform any communication, and *processes,* which can communicate with one another. The process portion of the language is a restriction of standard π-calculus [MPW92]. All computation done by a process is expressed using terms. Since our goal is to model probabilistic polynomial-time adversaries by quantifying over processes definable in our language, it is essential that all functions definable by terms lie in probabilistic polynomial time. Although we use pseudocode to write terms in this paper, we have developed an applied, simply-typed lambda calculus which exactly captures the probabilistic polynomial-time terms [MMS98].

The syntax of processes is given by the following grammar:

$$\begin{array}{rll}
P ::= & \mathbf{0} & \text{(termination)} \\
& \nu_{c_{q(|n|)}}.(P) & \text{(private channel)} \\
& c_{q(|n|)}(x).P & \text{(input)} \\
& c_{q(|n|)}\langle T\rangle & \text{(output)} \\
& [T = T].P & \text{(match)} \\
& P \mid P & \text{(parallel composition)} \\
& !_{q(|n|)}.P & \text{($q(|n|)$-fold replication)}
\end{array}$$

Polynomials appear explicitly in the syntax of processes in two places, in channel names and in replication. In a channel name $c_{q(|n|)}$, the polynomial

$q(|n|)$ associated with the channel c indicates that for some value n of the security parameter, channel c can carry values of $q(|n|)$ bits or fewer. This restriction on the size of natural numbers that are communicated from one process to another is needed to maintain the polynomial-time restriction on process computations. Replication $!_{q(|n|)}.P$ results in $q(|n|)$ copies of process P, where n is again the security parameter. For simplicity, after fixing n when we evaluate a process P, we replace all subexpressions of P of the form $!_{q(|n|)}.R$ with $q(|n|)$ copies of R in parallel. We also assume that all channel names and variable names are α-renamed apart.

The operational semantics of this process calculus is fairly intricate, due to probabilistic considerations and the desire to keep communication on a private channel from biasing the probabilities associated with externally observable communication on public channels. In brief, executing a process step begins with *outer evaluation* of any terms. In a process $[T_1 = T_2].P$, for example, we evaluate terms T_1 and T_2 before possibly performing any communication inside P. Similarly, execution of $c_{q(|n|)}\langle T\rangle$ begins with the evaluation of the term T, and execution of $P \,|\, Q$ with the outer-evaluation of both P and Q.

Once a process is outer-evaluated, a set of eligible communication pairs is selected. The set of *schedulable processes* $S(P)$ is defined inductively by

$$\begin{array}{ll} S(\mathbf{0}) & = \emptyset \\ S(\nu_{c_{p(|n|)}}.(Q)) & = S(Q) \\ S(c_{p(|n|)}(x).Q) & = \{c_{p(|n|)}(x).Q\} \\ S(c_{p(|n|)}\langle T\rangle) & = \{c_{p(|n|)}\langle T\rangle\} \\ S(Q_1 \mid Q_2) & = S(Q_1) \cup S(Q_2) \end{array}$$

Since P is outer-evaluated prior to computing $S(P)$, we do not need to consider the case $P \equiv [T_1 = T_2].Q$. Note that every process in $S(P)$ is either waiting for input or ready to output. The set of *communication triples* $C(P)$ is

$\{\langle P_1, P_2, Q_{P_1,P_2}[\ \ ]\rangle \,|\, P_i \in S(P), P_1 \equiv c_{p(|n|)}\langle a\rangle, P_2 \equiv c_{p(|n|)}(x).R, P \equiv Q_{P_1,P_2}[P_1, P_2]\}$

and the set of *eligible processes* $E(P)$ is defined by

$$E(P) = \begin{cases} C(P)|_{\text{private channels}} & \text{if there is a possible private communication} \\ C(P)|_{\text{public channels}} & \text{otherwise .} \end{cases}$$

The reason for this definition, explained intuitively in [LMMS99], is to keep communication on a private channel from biasing the probabilities associated with externally observable communication on public channels. Once a set of eligible processes have been determined, a computation step of P proceeds by selecting one communication triple from $E(P)$ at random and performing the resulting communication step.

4 Equivalence

An *observation* is a test on a specific public channel for a specific natural number. More precisely, let *Obs* be the set of all pairs $\langle i, c_{p(|n|)} \rangle$ where i is a natural number and $c_{p(|n|)}$ is a public channel. If, during an evaluation of process expression P, the scheduler selects the communication triple

$$\langle c_{p(|n|)}\langle i \rangle, c_{p(|n|)}(x).P', Q_{c_{p(|n|)}\langle i \rangle, c_{p(|n|)}(x).P'} \rangle$$

we will say that the observable $\langle i, c_{p(|n|)} \rangle \in \mathit{Obs}$ *occurs* and write $P \leadsto \langle i, c_{p(|n|)} \rangle$.

A process P may contain the security parameter n, as described above. We will write P_m to signify that the parameter n is assigned the natural number m. A *process family* $\mathcal{P}$ is the set $\langle P_i | \, i \in \mathbb{N} \rangle$. Since contexts may contain the process parameter n, we can define the *context family* $\mathcal{C}[\ \ ]$ analogously.

If $\mathcal{P}$ and $\mathcal{Q}$ are two process families, then $\mathcal{P}$ and $\mathcal{Q}$ are *observationally equivalent*, written write that $\mathcal{P} \cong \mathcal{Q}$, if

$$\forall q(x).\forall \mathcal{C}[\ \ ].\forall o \in \mathit{Obs}.\exists n_o.\forall n > n_o : \qquad \left|\text{Prob}(C[P] \leadsto o) - \text{Prob}(C[Q] \leadsto o)\right| \leq \frac{1}{q(n)}$$

where $\mathcal{C}[\ \ ]$ indicates a context family and $q(x)$ an everywhere-positive polynomial.

It is straightforward to check that $\cong$ is an equivalence relation. Moreover, we believe that this formal definition reasonably models the ability to distinguish two processes by feasible intervention and observation. If $P = \{P_n\}_{n \geq 0}$ is a scheme for generating pseudorandom sequences of bits, and $Q = \{Q_n\}_{n \geq 0}$ consists of processes that generate truly random bits (e.g., by calls to our built-in random-bit primitive), then our definition of observational equivalence corresponds to a standard notion from the study of pseudorandomness and cryptography (see, e.g., [Lub96,Yao82]). Specifically, $P \simeq Q$ iff P and Q pass the same polynomial-time statistical tests.

5 Applications and Future Directions

An example authentication protocol, proposed by Bellare and Rogaway [BR94], is discussed in [LMMS99]. However, the proof of security of this protocol that is presented in [LMMS99] is ad hoc, and relies on specific syntactic similarities between the protocol and its specification. In the future, we hope to develop more powerful systematic proof methods for observational congruence. Since there has been little prior work on complexity-bounded probabilistic process formalisms and asymptotic equivalence, one of our near-term goals is to better understand the forms of probabilistic reasoning that would be needed to carry out more rigorous protocol analysis.

References

[AF01] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th ACM Symposium on Principles of Programming Languages*, pages 104–115, 2001.

[AG98] M. Abadi and A. Gordon. A bisimulation method for cryptographic protocol. In *Proc. ESOP'98, Springer Lecture Notes in Computer Science*, 1998.

[AG99] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 143:1–70, 1999. Expanded version available as SRC Research Report 149 (January 1998).

[BAN89] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society, Series A*, 426(1871):233–271, 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

[BR94] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology - CRYPTO '93, Lecture Notes in Computer Science, Vol. 773*, 1994.

[DY83] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

[ElG85] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31:469–472, 1985.

[FKK96] A. Freier, P. Karlton, and P. Kocher. The SSL protocol version 3.0. `draft-ietf-tls-ssl-version3-00.txt`, November 18 1996.

[KMM94] R. Kemmerer, C. Meadows, and J. Millen. Three systems for cryptographic protocol analysis. *J. Cryptology*, 7(2):79–130, 1994.

[KN93] J.T. Kohl and B.C. Neuman. The Kerberos network authentication service (version 5). Internet Request For Comment RFC-1510, September 1993.

[KNT94] J.T. Kohl, B.C. Neuman, and T.Y. Ts'o. *The evolution of the Kerberos authentication service*, pages 78–94. IEEE Computer Society Press, 1994.

[LMMS98] P.D. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *ACM Conf. Computer and Communication Security*, 1998.

[LMMS99] P.D. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time equivalence and security protocols. In *FM'99 World Congress On Formal Methods in the Development of Computing Systems*, 1999.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In *2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*. Springer-Verlag, 1996.

[Lub96] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton Computer Science Notes, Princeton University Press, 1996.

[Mea96] C. Meadows. Analyzing the Needham-Schroeder public-key protocol: a comparison of two approaches. In *Proc. European Symposium On Research In Computer Security.* Springer Verlag, 1996.

[MMS97] J.C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murφ. In *Proc. IEEE Symp. Security and Privacy*, pages 141–151, 1997.

[MMS98] J. Mitchell, M. Mitchell, and A. Scedrov. A linguistic characterization of bounded oracle computation and probabilistic polynomial time. In *IEEE Symp. Foundations of Computer Science*, 1998.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part i. *Information and Computation*, 100(1):1–40, 1992.

[NS78] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[Pau97a] L.C. Paulson. Mechanized proofs for a recursive authentication protocol. In *10th IEEE Computer Security Foundations Workshop*, pages 84–95, 1997.

[Pau97b] L.C. Paulson. Proving properties of security protocols by induction. In *10th IEEE Computer Security Foundations Workshop*, pages 70–83, 1997.

[Ros95] 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 Soc Press, 1995.

[Sch96] S. Schneider. Security properties and CSP. In *IEEE Symp. Security and Privacy*, 1996.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *IEEE Foundations of Computer Science*, pages 80–91, 1982.