Probabilistic Polynomial-Time Equivalence and Security Analysis

P. Lincoln *, [1], J. Mitchell **, [2], M. Mitchell ***, [2], and A. Scedrov †, [3]

[1] Computer Science Laboratory, SRI International
[2] Department of Computer Science, Stanford University
[3] Department of Mathematics, University of Pennsylvania

Abstract. We use properties of observational equivalence for a probabilistic process calculus to prove an authentication property of a cryptographic protocol. The process calculus is a form of π-calculus, with probabilistic scheduling instead of nondeterminism, over a term language that captures probabilistic polynomial time. The operational semantics of this calculus gives priority to communication over private channels, so that the presence of private communication does not affect the observable probability of visible actions. Our definition of observational equivalence involves asymptotic comparison of uniform process families, only requiring equivalence to within vanishing error probabilities. This definition differs from previous notions of probabilistic process equivalence that require equal probabilities for corresponding actions; asymptotics fit our intended application and make equivalence transitive, thereby justifying the use of the term "equivalence." Our security proof uses a series of lemmas about probabilistic observational equivalence that may well prove useful for establishing correctness of other cryptographic protocols.

1 Introduction

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. Over the past decade or two, a variety of methods have been 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], as well as theorem proving [Pau97a, Pau97b] and model-checking methods using general purpose tools [Low96, Mea96, MMS97, Ros95, Sch96].

* Partially supported by DoD MURI "Semantic Consistency in Information Exchange," ONR Grant N00014-97-1-0505.
** Additional support from NSF CCR-9629754.
*** Additional support from Stanford University Fellowship.
† Additional support from NSF Grant CCR-9800785.

In two previous papers [LMMS98, MMS98], we outlined a framework for protocol analysis employing assumptions different from those used in virtually all other formal approaches. Specifically, most formal approaches use a basic model of adversary capabilities which appears to have developed from positions taken by Needham and Schroeder [NS78] and a model presented by Dolev and Yao [DY83]. This set of modeling assumptions treats cryptographic operations as "black-box" primitives, with plaintext and ciphertext treated as atomic data that cannot be decomposed into sequences of bits. Furthermore, as explained in [MMS97, Pau97a, Sch96], there are limited ways for an adversary to learn new information. For example, if a 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 coupled with nondeterministic computation on the part of the adversary, provide an idealized setting in which protocol analysis becomes relatively tractable. However, this model reduces the power of the adversary relative to real-world conditions. As a result, it is possible to prove a protocol correct in this standard model, even when the protocol is vulnerable to simple deterministic attacks.

Our goal is to establish a framework that can be used to analyze protocols (and, potentially, other computer security components) under the standard assumptions of complexity-based cryptography. In [LMMS98], we outlined a refinement of spi-calculus [AG97] that requires a calculus of communicating probabilistic polynomial-time processes and an asymptotic form of observational equivalence. We proposed basic definitions of the key concepts and discussed the potential of this framework by examining some extremely simple protocols. The sequential fragment of our calculus is developed in more detail in [MMS98], where a precise correspondence is proved between a modal-typed lambda calculus and probabilistic polynomial-time computation. In the present paper, we test our basic definitions by considering further applications and develop a more refined probabilistic semantics. Using our improved semantics, we sketch a proof of correctness for a less trivial protocol. Specifically, we prove correctness of a mutual authentication protocol proposed by Bellare and Rogaway [BR94]. This security proof involves some reasoning about a specific form of asymptotic probabilistic observational equivalence for our process calculus. Since, to the best of our knowledge, there has been no previous work on process equivalence up to some error tolerance, this argument and the difficulties we have encountered motivate further investigation into resource-bounded probabilistic semantics and information hiding.

In addition to relying on the basic relation between observational equivalence and security properties developed in the spi-calculus [AG97], we have drawn inspiration from the cryptography-based protocol studies of Bellare and Rogaway [BR94, BR95]. In these studies, a protocol is represented as a set of oracles, each corresponding to one input-output step by one principal. These oracles are each available to the adversary, which is represented by a probabilistic polynomial-time oracle Turing machine. There are some similarities to our setting, since an adversary has access to each input-output step by a principal by sending and receiving data on the appropriate ports. However, there are some significant technical and methodological differences. In our setting, the protocol and the adversary are both expressed in a formal language. The use of a formal language allows for proof techniques that are based on either the syntactic structure of the protocol or on the semantic properties of all expressible adversaries. We have found the specification method we have adopted from spi-calculus to be relatively natural and more systematic than the specifications used by Bellare and Rogaway. In particular, it appears that our specification of authentication is stronger than the one used in [BR94], requiring us to prove more about the observable properties of a protocol execution. Finally, by structuring our proof around observational equivalence, we are led to develop general methods for reasoning about probabilistic observational equivalence that should prove useful in analyzing other protocols.

2 Process Calculus for Protocol Analysis

A protocol consists of a set of programs that communicate over some medium in order to achieve a certain task. Typically, these programs are parameterized by a *security parameter* k, with the idea that increasing the value of k makes the protocol more secure. Often, k is just the length of the keys used in the protocol since it is expected that longer encryption keys make decryption more difficult.

For simplicity, we will consider only those protocols that require some fixed number of communications, independent of the security parameter. In other words, the number of messages sent back and forth before the protocol completes does not increase, even as the security parameter is "cranked up," although the length of the keys used throughout the protocol will increase. This simplification is appropriate for most handshake protocols, key-exchange protocols and authentication protocols. (Many widely-used protocols, including he authentication phase of SSL, serve as examples of "real-world" protocols where the number of messages remains fixed, even as the security parameter is increased.) We are in the process of extending our process calculus to allow looping, which will allow us to deal with more complex protocols, such as those used to prove zero-knowledge. In the present paper, however, we present methods for reasoning about asymptotic observational equivalence that rely on having a fixed bound on the depth of the concurrent process execution tree, and are therefore inappropriate for protocols where the number of messages depends on the security parameter.

Following the work of Abadi and Gordon [AG97], we express security properties of a protocol P by writing an idealized protocol Q which is "patently secure." (Typically, Q requires magic machinery not available in real computational environments, such as perfect random number generators or perfectly secure communication channels.) Then, we endeavor to show that, for any adversary, the interactions between the adversary and P have the same observable behavior as the interactions between the adversary and Q. If this condition holds, we can replace the ideal protocol Q with the realizable protocol P, without compromising security.

The adversary may then be thought of as a *process context*, at which point the task of reasoning about security is reduced to the task of reasoning about observational equivalence (also called observational congruence). Our framework is a refinement of the spi-calculus approach in that we replace nondeterministic computation with probabilistic polynomial-time computation while simultaneously shifting from a standard observational equivalence to an asymptotic form of observational equivalence.

2.1 Syntax

The syntax of our probabilistic polynomial-time calculus consists of *terms* and *processes.* The process portion of the language is a bounded subset of asynchronous π-calculus. However, readers familiar with the traditional π-calculus will note the absence of scope extrusion, or the ability to pass channel names. These omissions are purposeful, and necessary, in order that the expressive power of the calculus correspond to what is commonly believed reasonable in the cryptographic community. It is best to think of the calculus presented here as a notationally familiar means of expressing parallelism and communication, rather than to compare it directly to more traditional forms of π-calculus.

The term portion of the language is used to express all data dependent computation. All terms have natural number type, so the only values communicated from process to process are natural numbers. We do not present a formal grammar or semantics for the term calculus (although we did so in [MMS98]). For the purposes of this paper, the important consideration is that the term language be able to express precisely the probabilistic polynomial time functions from integers to integers. (Therefore, an alternative formalism to that employed in [MMS98] would be probabilistic Turing machine descriptions, together with explicit polynomial time limits, and the understanding that a Turing machine computation that exceeds its time limit outputs zero.) Because the syntax of the term language is not critical for the level of analysis presented in this paper, we use pseudo-code to express terms throughout the paper.

In the grammar below, P varies over processes, T over terms, x over term variables, and c over a countably infinite set C of channel names. The set of well-formed processes is given by the following grammar:

$$
\begin{array}{rlr}
P ::= & 0 & \text{(termination)} \\
 & (\nu \mathsf{c}).P & \text{(private channel)} \\
 & \mathsf{c}(x).P & \text{(input)} \\
 & \bar{\mathsf{c}}\langle T \rangle & \text{(output)} \\
 & [T = T].P & \text{(match)} \\
 & P \mid P & \text{(parallel composition)}
\end{array}
$$

To simplify the presentation of our probabilistic scheduling conventions, we partition the set C of channel names into two disjoint subsets: the *private channel names* and the *public channel names*. Any name c bound with $(\nu \mathsf{c})$ must be a "private" channel name.

Communication between separate principals of a protocol will normally take place across public channels. Private channels are used to communicate between processes that are considered part of a single principal. Typically, these processes would be running on a single machine, and their communication would therefore be invisible to other machines on the network. Private channels are also used to express offline initialization steps in a protocol, such as the exchange of keys prior to the beginning of some communication.

Private channels are also used in writing specifications; there, they are used to transfer information between processes in a way that is secure by fiat. Often this information is transmitted in encrypted form in the actual implementation; the statement of observational equivalence expresses the fact that encrypted communication should behave similarly to totally private communication.

We often sweeten our process descriptions with a little syntactic sugar. We write $!_n P$ to mean the n-fold parallel composition of P with itself. We write $\mathtt{LET}\ x = y\ \mathtt{IN}\ P$ as shorthand for $(\nu \mathsf{c}).(\bar{\mathsf{c}}\langle y \rangle \mid \mathsf{c}(x).P)$ where c is some private channel not occurring in P. As usual, we say P is closed if all variables in P are bound.

2.2 Probabilistic Scheduling

Traditional process calculi assume nondeterministic scheduling of computations. In particular, when there are several steps which could be chosen next, the one actually chosen is selected nondeterministically. One motivation for this point of view is derived from failure analysis. If one wishes to prove a mission-critical system to be one hundred percent reliable, it is important to consider all possible interleavings of computation in order to see whether any of them yield an unacceptable outcome.

However, from the point of view of realistic security analysis, nondeterminism gives too much power to the adversary. For example, an adversary allowed to choose a number nondeterministically may just select the key required to decrypt the message. Security analysis, however, is founded on the notion that such an event can happen only with negligible probability, and is therefore of limited concern.

For the sake of concreteness, and to provide an introduction to our notation and methodology, we consider a simple protocol in which one party, A, wishes to send a message securely to another party, B, using public-key encryption. A will attempt to accomplish this feat by encrypting the message with B's k-bit public key (K_b). In the notation commonly found in the literature, this protocol would be expressed as:

$$A \to B : \{\mathsf{msg}\}_{K_b}$$

The notation $A \to B$ indicates a message from A to B, while $\{x\}_y$ is commonly used to indicate a message (plaintext) x encrypted under key y. In our system, we would describe the protocol as

$$\overline{\scriptstyle AB}\langle encrypt(K_b, \mathsf{msg})\rangle$$

The channel name ${\scriptstyle AB}$ is used to indicate that the message is being sent from A to B. Of course, an adversary might intercept or modify the message, so the channel name serves only as a mnemonic.

We assume that an evil adversary wishes to discover the message msg. If we allow the adversary to consist of the parallel composition of 3 processes E_0, E_1 and E, scheduled nondeterministically, then the message can be discovered. Specifically, we let

$$\begin{aligned}
E_0 &= !_k \overline{\scriptstyle E}\langle 0\rangle \\
E_1 &= !_k \overline{\scriptstyle E}\langle 1\rangle \\
E \ &= {\scriptstyle E}(b_0).\ \ldots {\scriptstyle E}(b_{k-1}).\ {\scriptstyle AB}(x). \\
&\quad\ \overline{\scriptstyle Public}\langle decrypt(conc(b_0, \ldots, b_{k-1}), \mathsf{msg})\rangle
\end{aligned}$$

Processes E_0 and E_1 each send k bits on the same channel. The intruder E reads the message from A to B, nondeterministically reads the bits from E_0 and E_1 in such an order so as to obtain B's private key, and decrypts the message. Although at first one might think that eliminating nondeterminism from the term calculus would be sufficient, this example demonstrates that nondeterminism cannot be allowed even at the level of the process calculus.

Our probabilistic operational semantics is in the same spirit as Milner's reaction relation approach [Mil92], which was inspired by the Chemical Abstract Machine of Berry and Boudol [BB90]. Simply put, the reduction step is the "reaction" between a process ready to output on a channel and a process ready to receive input on that same channel. Our operational semantics provides a means of calculating, at any point, which processes are eligible to interact, and then of choosing probabilistically from among this set.

There are actually two sources of randomness in the execution of a process. The first, just discussed, is in the choice of which processes will execute next. The second comes in the computation of terms, which themselves perform probabilistic computations. Specifically, for any closed term T there is a finite set of possible values $T_1, \ldots, T_k$ such that the probability of T evaluating to T_i is p_i and $\sum_{i=1,k} p_i = 1$.

Our goal is to devise an intuitively plausible probabilistic semantics that refines the standard nondeterministic semantics of π-calculus and allows us to model faithfully the security phenomena of interest. Subject to these two primary goals, we would also like to have as many natural equivalences as possible. For example, all other things being equal, we would like parallel composition to be associative. The first goal, refining the standard nondeterministic semantics, means that our operational semantics will induce some probability distribution on the set of execution sequences allowed by the nondeterministic semantics.

We assign probabilities in a "local" manner, independent of the history of prior steps leading to any process state.

That communications on private channels be unobservable is of the utmost importance in our framework. For example, it is vital that

$$P \quad \cong \quad (\nu \mathsf{c}).(\,\overline{\mathsf{c}}\langle 0\rangle \mid \mathsf{c}(x).P\,)$$

when x does not occur free in P. In words, we want any process P to be equivalent to the process that transmits some value on a private channel, discards the result of that communication, and then proceeds as P. After all, given that private channels are considered private, out-of-band communication mechanisms, there is no way that an adversary should be able to observe the private communication.

In order to guarantee unobservability, we must ensure not only that the contents of the transmission are unavailable to the adversary, but also that the existence of the transmission cannot become known. In particular, the communication must not skew the probability of other actions in the process as the adversary might otherwise be able to distinguish the processes by sampling the behavior of the (supposedly equivalent) processes.

Consider the following concrete example:

$$\overline{\mathsf{A}}\langle 0\rangle \mid \overline{\mathsf{A}}\langle 1\rangle.$$

We would like this process to be equivalent to:

$$((\nu \mathsf{c}).(\overline{\mathsf{c}}\langle 0\rangle \mid \mathsf{c}(x).\overline{\mathsf{A}}\langle 0\rangle)) \mid \overline{\mathsf{A}}\langle 1\rangle.$$

However, if our operational semantics were to select from all possible next steps with equal probability, then the first process would output a 0 followed by a 1 one half of the time, and the sequence $1, 0$ the other half of the time. The introduction of the private channel in the second process would bias the computation so that the $0, 1$ sequence would occur with only twenty-five percent probability while the $1, 0$ sequence would occur with seventy-five percent probability. In other words, the most obvious probabilistic scheduling rules yield a situation in which the introduction of a silent action changes the behavior of the process as a whole. The solution we have chosen is to give priority to silent actions, allowing them to occur before any reductions involving public channels. We thereby keep the probability of scheduling silent actions from interfering with the scheduling of observable actions.

2.3 Operational Semantics

The probabilistic operational semantics of processes is given in three parts. The first part is essentially a structural equivalence, in the same sprit as Milner's structural-equivalence-based semantics for π-calculus [Mil92], presented using multisets of subprocesses instead of a syntactic transformation of process expressions. The second part determines the set of eligible process combinations, each consisting of either a matching output and input, or a single process that requires evaluation of a conditional test. The third part of the operational semantics is the definition of the execution graph of a process, which consists of edges between sets of schedulable processes, each edge labeled with a probability and any visible communication that may appear as a result of the computation step.

The basic computation steps of our process calculus are either a communication, consisting of an output and matching input, or a guarded process whose guard is to be evaluated. Therefore, a *schedulable process* is a process of one of the following three forms: $\mathsf{c}(x).Q$, called an *output process*, $\bar{\mathsf{c}}\langle T\rangle$, called an *input process*, or $[T_1 = T_2].Q$, called a *guarded process*.

If P is any process, then the multiset $S(P)$ of *schedulable subprocesses of* P is defined inductively as follows:

$$S(P) = \begin{cases} \emptyset & \text{if } P = 0 \\ S(Q) & \text{if } P = (\nu \mathsf{c}).Q \\ \{P\} & \text{if } P \text{ is } \textit{schdulable} \\ S(P_1) \cup S(P_2) & \text{if } P = P_1 | P_2 \end{cases}$$

Since the semantics of a process P is determined by $S(P)$, we can think of P and Q as structurally equivalent if $S(P)$ and $S(Q)$ are identical multisets.

The execution of a process consists of selecting one computation step from the set of schedulable subprocesses. To simplify a later definition, we say a singleton $[T_1 = T_2].Q$ is an *eligible computation*, and a pair $(\bar{\mathsf{c}}\langle T\rangle, \mathsf{c}(x).Q)$ is an *eligible computation*. Due to priority of internal actions over external ones, communication will only occur if there are no eligible guarded processes, and communication on public channels will occur only if there are no eligible computations on a private channel.

The *execution graph* $G(P)$ of process P has nodes that are multisets of schedulable processes and arcs labeled with probabilities. In addition to the probability that a transition will occur, an arc may also be labeled with a channel name and value, if the transition involves communication on a public channel. The probabilities will depend on both the probabilistic scheduling of subprocesses, and the probabilistic evaluation of terms (explained in [MMS98]). Intuitively, if $S(P) = \{P_1, \ldots, P_n\}$, we select uniformly at random from among the eligible computations of the highest priority. More specifically, the root of $G(P)$ is the multiset $S(P)$ of schedulable processes of P. The arcs of $G(P)$ are determined as follows:

- There is an arc from multiset $R = R' \cup \{[T_1 = T_2].Q\}$ to $R \cup S(Q)$ with probability p if there are k eligible guarded processes in R, the probability that T_1 and T_2 evaluate to the same value is r, and $p = r/k$.
- There is an arc from multiset $R = R' \cup \{\bar{\mathsf{c}}\langle T\rangle, \mathsf{c}(x).Q\}$ to $R \cup S([v/x]Q)$ with probability p if there are no guarded processes in R, there are k eligible communications along private channels in R, the probability that T evaluates to v is r, and $p = r/k$.

– There is an arc from multiset $R = R' \cup \{\bar{c}\langle T\rangle, c(x).Q\}$ to $R \cup S([v/x]Q)$, labeled with probability p and pair $\langle c, v\rangle$ if there are no guarded processes in R, there are no eligible communications along private channels, there are k eligible communications along public channels in R, the probability that T evaluates to v is r, and $p = r/k$.

The nodes of $G(P)$ are all sets of schedulable processes that lie along a path from the root of $G(P)$.

Intuitively, we assume that public communications are externally observable, while the evaluation of guards and private communication are not. Formally, an *observation* is a sequence of pairs $\langle c, v\rangle$, each consisting of a public channel name and a value. We say an observation o *occurs* in the execution of P if o is a subsequence of the labels along a path in $G(P)$. For example, the observation $\langle c, 1\rangle, \langle d, 2\rangle$ occurs in the execution of $\bar{c}\langle 1\rangle|\bar{c}\langle 2\rangle|\bar{d}\langle 2\rangle$. The probability of a path in $G(P)$ is the product of all the probabilities along the path, and the *probability of observation* o in the execution of P is the sum of the probabilities of all paths in $G(P)$ on which o occurs.

3 Process Equivalence

Two processes 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. To make this more precise for a specific programming language $\mathcal{L}$, we assume the language definition gives rise to some set of *program contexts,* each context $\mathcal{C}[\ ]$ consisting of a program with a "hole" (indicated by empty square brackets $[\ ]$) in which to insert a phrase of the language, and some set *Obs* of concrete *observable actions,* such as integer or string outputs. We also assume that there is some semantic evaluation relation $\overset{eval}{\leadsto}$, with $M \overset{eval}{\leadsto} v$ meaning that evaluation or execution of the program M produces the observable action v.

We perform an *experiment* on a process P by placing it in a context $\mathcal{C}[\ ]$, running the resulting process, and seeing whether or not a particular observable v occurs.The main idea underlying the concept of observational equivalence is that the properties of a program that matter are precisely the properties that can be observed by experiment. Although we presented the basic form of observational equivalence below in [LMMS98], we repeat the basic motivation here for completeness. We now commit ourselves to uniform families of processes and contexts (in which all members of a family represented by the same expression, but parameterized by some natural number), mirroring the usual assumptions regarding protocols and security parameters mentioned earlier.

3.1 Definition of Equivalence

For the process language considered in this paper, we are interested in contexts that distinguish between processes. (We will not need to consider observational equivalence of terms.) Therefore, the contexts of interest are process expressions with a "hole", given by the following grammar

$$\begin{array}{c} \mathcal{C}[\;] ::= [\;] \quad | \quad n(x).\,\mathcal{C}[\;] \quad | \quad P|\mathcal{C}[\;] \quad | \quad \mathcal{C}[\;]|Q \quad | \\ (\nu \mathsf{c}).\mathcal{C}[\;] \quad | \quad \bar{\mathsf{c}}\langle T\rangle \quad | \quad [M = N]\mathcal{C}[\;] \end{array}$$

A process observation will be a communication event on a public channel. More specifically, we let Obs be the set of all possible observations, *i.e.*, the set of pairs $\langle n, m\rangle$, where n is a public channel name and m is an integer. We write $P \overset{eval}{\leadsto} o$ if evaluation of process expression P results in the observation $o \in Obs$.

Intuitively, given program phrases P and Q, context $\mathcal{C}[\;]$ and observation o, it seems reasonable to compare the probability that $\mathcal{C}[P] \overset{eval}{\leadsto} o$ to the probability that $\mathcal{C}[Q] \overset{eval}{\leadsto} o$. However, since a probability distribution is an infinite entity, it is not clear how to "observe" a distribution. We might run $\mathcal{C}[P]$ some number of times, count how many times o occurs, and then repeat the series of experiments for $\mathcal{C}[Q]$. If the probabilities are very different, then we might be able to observe this difference (with high confidence) by a few runs of each program. However, if the probabilities are very close, then it might take many more runs of both programs to distinguish them.

As a first step toward developing a workable notion of observable equivalence, we define computational indistinguishability within factor ϵ by saying that $P \simeq_\epsilon Q$ if

$$\forall \mathcal{C}[\;].\ \forall v \in Obs.\ |\mathrm{Prob}[\mathcal{C}[P] \overset{eval}{\leadsto} v] - \mathrm{Prob}[\mathcal{C}[Q] \overset{eval}{\leadsto} v]| \leq \epsilon$$

An immediate difficulty with $\simeq_\epsilon$ is that it is not a transitive relation. Moreover, it is not clear how to differentiate large ϵ from small ϵ. Specifically, we would like to draw a distinction between sets of processes that are "close" in behavior from those that are "far apart." Intuitively, the distinction should have something to do with running time, since it takes more trials to distinguish random variables that differ by a small amount than to distinguish random variables that differ by a large amount.

We can bring concepts from asymptotic complexity theory to bear on the situation if the processes P and Q under consideration are actually families of processes indexed by natural numbers. This point of view fits our intended application, since, as mentioned earlier, cryptographic primitives and security protocols are generally defined with some security parameter that may be increased if greater resistance to tampering is required.

A *process family* P is a process of the form $\mathsf{n}(n).P'$. If P is a process family, we write P_n for $(\nu \mathsf{n}).(\bar{\mathsf{n}}\langle n\rangle \mid P)$. A context family is the analogous construction, but with a single hole permitted in the body of P. Let us assume that $P = \{P_n\}_{n\geq 0}$ and $Q = \{Q_n\}_{n\geq 0}$ are process families and $\mathcal{C}[\;] = \{\mathcal{C}_n[\;]\}_{n\geq 0}$ a family of contexts. We assume that the running times of P_n, Q_n and $\mathcal{C}_n[\;]$ are bounded by polynomials in n. Then for function f, we define *asymptotic equivalence* *within* f for two process families P and Q by writing $P \simeq_f Q$ if

$$\forall \mathcal{C}[\,].\, \forall o \in \mathit{Obs}.\, \exists n_0.\, \forall n \geq n_0.$$
$$|\mathrm{Prob}[\mathcal{C}_n[P_n] \overset{eval}{\rightsquigarrow} o] - \mathrm{Prob}[\mathcal{C}_n[Q_n] \overset{eval}{\rightsquigarrow} o]| \leq f(n)$$

In words, P and Q are asymptotically equivalent within f if, for every computational experiment given by a context family and an observable value, the difference between experimental observation of P_n and experimental observation of Q_n is bounded by $f(n)$, for all sufficiently large n.

Since we consider polynomial factors "small", we define *observational equivalence of probabilistic processes* by

$$P \simeq Q \quad \text{if} \quad P \simeq_{1/p} Q \text{ for every polynomial } p.$$

We sketch below the proof that this relation 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 and Q are two process families which are not observationally equivalent because the probability that $\mathcal{C}_n[P_n] \overset{eval}{\rightsquigarrow} o$ occurs with a noticeably different probability than $\mathcal{C}_n[Q_n] \overset{eval}{\rightsquigarrow} o$, then we call $\mathcal{C}$ the *distinguishing context* and call o the *distinguishing observation*.

3.2 Properties of Observational Equivalence

We note first that observational equivalence is indeed an equivalence relation.

Lemma 1. *Observational equivalence is reflexive, symmetric, transitive, and is an equivalence.*

Proof. Reflexivity and symmetry are immediate from the definition. Suppose now that $P \simeq_{1/p} Q$ and $Q \simeq_{1/q} R$. Let o be an arbitrary element of Obs. Let n_0^P be the least value such that the observational equivalence condition holds for P and Q. Define n_0^Q similarly, but for Q and R. Then, for all $n \geq \max(n_0^P, n_0^Q)$

$$|\mathrm{Prob}[\mathcal{C}_n[P_n] \overset{eval}{\rightsquigarrow} o] - \mathrm{Prob}[\mathcal{C}_n[R_n] \overset{eval}{\rightsquigarrow} o]| \leq 1/p(n) + 1/q(n).$$

Assuming, without loss of generality, that $p(n) \leq q(n)$, we may bound this value by $1/(p(n)/2)$. Since o was chosen arbitrarily, we may conclude that $P \simeq R$, as required. Equivalence follows by similar reasoning.

In order to analyze changes in the probability of observable actions, we inductively define a mapping from processes to directed graphs. For any process P, the vertices of the graph $G(P)$ are all processes to which P reduces in zero or more steps. The edges of the graph are labeled with probabilities, with an edge with label p from node Q to R if and only if $Q \rightsquigarrow_p R$. Note that for any process family P, the number of nodes and edges in $G(P_n)$ is independent of n, since there are no constructs in the process calculus that depend on the values of term variables. An *execution* σ of P is a path through $G(P)$. The probability p_σ is the product of the weights of the edges that make up σ. It is clear that $\sum_\sigma p_\sigma = 1$.

Lemma 2. *For any process P, $G(P)$ is acyclic, and the only node with no incoming edges is that corresponding to P.*

Proof. All processes have finite representations, and that reduction always shortens that representation.

If $\mathcal{C}[P] \rightsquigarrow_p R$ we say that this computation step does not *touch* P if there exists a context $\mathcal{D}$ such that $R = \mathcal{D}[P]$ and such that for all processes Q we have $\mathcal{C}[Q] \rightsquigarrow_p \mathcal{D}[Q]$. In other words, the computation step does not touch P if the step does not in any way depend on P. We say that a path σ touches P if any reduction along σ touches P.

Lemma 3. *Suppose $P \ncong Q$. Let $\mathcal{C}$ be a distinguishing context and o the corresponding distinguishing observation. Then, there exists a path σ through either $G(\mathcal{C}[P_n])$ or $G(\mathcal{C}[Q_n])$ such that $o = o_\sigma$ and such σ touches P_n or Q_n. Furthermore, p_σ must be at least $1/p(n)$ for some polynomial p.*

Proof. Assume, without loss of generality, that o occurs more often in $\mathcal{C}[P_n]$ than in $\mathcal{C}[Q_n]$. Then, there must exist a path σ through $G(\mathcal{C}[P_n])$ such that $o_\sigma = o$. Choose σ so that p_σ is maximal, from the set of paths such that $o_\sigma = o$. Denote the total number of possible paths through $G(\mathcal{C}[P_n])$ by c. Note that c is independent of n, by the comments above. Therefore, $p_\sigma \geq 1/cf(n)$ where f is the distinguishing polynomial for P and Q.

Lemma 4. *For any process family P, and any context family $\mathcal{C}$,*

$$\mathcal{C}[[x = y].P] \cong \mathcal{C}[P]$$

if

$$\mathcal{C}[[x = y].\bar{\mathsf{c}}\langle 1\rangle] \cong \mathcal{C}[\bar{\mathsf{c}}\langle 1\rangle] .$$

Proof. Suppose that $\mathcal{C}[[x = y].P] \cong \mathcal{C}[P]$. Let $\mathcal{D}$ be the distinguishing context. Then, by the previous lemma, there must exist a path through $\mathcal{D}[\mathcal{C}[[x = y].P]]$ or through $\mathcal{D}[\mathcal{C}[P]]$ touching one of the two processes with probability at least $1/p(n)$ for some polynomial p. But, by the hypothesis, this path also exists through the other process with probability at least $1/2p(n)$

Lemma 5. *Let $\mathcal{C}$ be an arbitrary context family with two holes, and let c be a private channel name unused in $\mathcal{C}$. Then, $(\nu \mathsf{c}).\mathcal{C}[\bar{\mathsf{c}}\langle T\rangle][\mathsf{c}(\_)] \cong \mathcal{C}[0][[1 = 1]]$ whenever the structure of $\mathcal{C}$ ensures that the second hole is in $S(\mathcal{C})$ only when the first hole is as well.*

In other words, private channels may be freely introduced, provided the data sent on them is never used, and provided that the output is always available by the time the input is requested.

Proof. By Lemma 3, it is sufficient to show simply that for any process family P, $P \cong (\nu \mathsf{c}).(\bar{\mathsf{c}}\langle \mathrm{T}\rangle \mid \mathsf{c}(\_).\mathrm{P})$ and to consider only contexts of the form $Q \mid []$. Now, suppose that $E(Q)$ contains no private channel communications. Then, the only one-step reduction from $\mathtt{Q} \mid (\nu\mathtt{c}).(\bar{\mathtt{c}}\langle\mathtt{T}\rangle \mid \mathtt{c}(\_).\mathtt{P})$ is to $\mathtt{Q} \mid \mathtt{P}$, so the result is clear. In general, any computation path which touches the hole will result in the reduction on $\mathtt{c}$. Because private communications are given priority by the scheduling model, any observable action of Q will be delayed until that reduction has occurred.

4 Analysis of the Bellare-Rogaway Authentication Protocol

We now turn our attention to an authentication protocol proposed by Bellare and Rogaway [BR94]. This protocol does not provide secrecy, but provides authentication based on a shared secret between the two parties. The secret is the index of one function from a pseudo-random family of functions. A pseudo-random family of functions is a set of functions, indexed by a natural number. Each function should behave unpredictably on successive inputs: there should be no polynomial time tests which reliably distinguish between a random function and these pseudo-random functions. Based only on this shared secret, the two parties can recognize computations that are performed by each other, but the same ability is not afforded to any observer or adversary who does not know the secret function.

Let f be a pseudo-random function family. Suppose that A and B share a secret natural number t, and thereby a secret pseudorandom function f_t. The protocol consists of three steps, here expressed in the notation usually found in the literature:

1. $A \rightarrow B\,\{N_a\}$
2. $B \rightarrow A\,\{K_b, K_a, N_a, N_b, f_t(\langle K_b, K_a, N_a, N_b\rangle)\}$
3. $A \rightarrow B\,\{K_a, N_b, f_t(\langle K_a, N_b\rangle)\}$

Here, N_a and N_b are randomly chosen "nonces," and K_a and K_b are numbers associated with known principles A and B (informally, one might consider these to be the public keys associated with those identities.) New nonces are chosen each time the protocol is executed. At the conclusion of the protocol, A knows N_b, B knows N_a, and both can be assured that these values came from the other. Intuitively, they have this assurance because nobody else can compute the values $f_t(\langle K_b, K_a, N_a, N_b\rangle)$ and $f_t(\langle K_a, N_b\rangle)$.

4.1 Expression in Process Calculus

We can express the protocol more formally in our calculus. Our calculus requires more precision than the informal notation above. In particular, the behavior of participants who receive ill-formatted messages is made explicit; they immediately halt.

We construct the expression for this protocol in a modular fashion. First, we give a formal description of a random number server which will play the role of the pseudo-random function. We use the following expression, parameterized by a number N, for the N^{th} random number server. In particular, let $R(N)$ be the fragment given by:

$$
\begin{array}{l}
\mathsf{RS_N}(\mathtt{x}).(\ \mathtt{LET\ r} = \mathtt{f_t}(\mathtt{k})\ \mathtt{IN} \\
\qquad\quad \overline{\mathsf{RND_N}}\langle \mathtt{r}\rangle \\
\qquad\quad\ \mid \mathsf{RCHK_N}(\mathtt{y},\mathtt{z}).[\mathtt{y} = \mathtt{x}\ \mathtt{AND}\ \mathtt{z} = \mathtt{r}].\overline{\mathsf{ROK_N}}\langle 1\rangle)
\end{array}
$$

Here, it is understood that $R(1)$, say, will be the fragment given above, but with the N in the channel names replaced with 1, so that, for example, the first read will occur on $\mathsf{RS_1}$. The so-called random number server is a code fragment that will associate a random number (r) with a value (x), and remember the association so that it may be queried later. In this case, of course, there is no real randomness, since $\mathtt{f_t}$ is deterministic. However, the specification of the protocol will involve replacing the pseudo-random function with real randomness, and this change will make necessary keeping track of what random numbers were assigned to what values. This construction is a specific example of a more general construct; namely, it is a means of sharing a function between multiple processes in the same way that LET shares a value between multiple processes.

Next we describe the actions of the actual parties themselves. The fragment for A is given by:

$$
\begin{array}{l}
\mathtt{LET\ N_a} = \mathtt{rand}(\mathtt{k})\ \mathtt{IN} \\
\quad \overline{\mathsf{AB_1}}\langle \mathtt{N_a}\rangle \\
\quad\ \mid \mathsf{BA}(\mathtt{w},\mathtt{x},\mathtt{y},\mathtt{N_b},\mathtt{z}).[\mathtt{w} = \mathtt{K_b}\ \mathtt{AND}\ \mathtt{x} = \mathtt{K_a}\ \mathtt{AND}\ \mathtt{y} = \mathtt{N_a}]. \\
\qquad\qquad (\ \overline{\mathsf{RCHK_1}}\langle\{\mathtt{K_b},\mathtt{K_a},\mathtt{N_a},\mathtt{N_b}\}\,,\mathtt{z}\rangle \\
\qquad\qquad\ \mid \mathsf{ROK_1}(\_).(\ \overline{\mathsf{RS_2}}\langle\{\mathtt{K_a},\mathtt{N_b}\}\rangle \\
\qquad\qquad\qquad\qquad\quad \mid \mathsf{RND_2}(\mathtt{r}).\overline{\mathsf{AB_2}}\langle \mathtt{K_a},\mathtt{N_b},\mathtt{r}\rangle \\
\qquad\qquad\qquad\qquad\quad \mid \overline{\mathsf{AOK}}\langle 1\rangle))
\end{array}
$$

Here $\mathtt{rand(k)}$ is a function that returns a truly random k-bit number. The third line checks that the message from B in the third step of the protocol is reasonable. In particular, we check that B sends back the nonce sent by A, and that the hash value $f_t(\{K_b, K_a, N_a, N_b\})$ is correct. The last step is the one that is supposed to be authenticating B, since presumably (with high probability) only A and B can compute $f_t(\{K_b, K_a, N_a, N_b\})$.

Finally, the fragment for B is given by:

$$
\begin{array}{l}
\mathtt{LET\ N_b} = \mathtt{rand}(\mathtt{k})\ \mathtt{IN} \\
\quad \mathsf{AB_1}(\mathtt{N_a}).\ (\ \overline{\mathsf{RS_1}}\langle\{\mathtt{K_b},\mathtt{K_a},\mathtt{N_a},\mathtt{N_b}\}\rangle \\
\qquad\qquad\quad \mid \mathsf{RND_1}(\mathtt{r}).\overline{\mathsf{BA}}\langle \mathtt{K_b},\mathtt{K_a},\mathtt{N_a},\mathtt{N_b},\mathtt{r}\rangle \\
\qquad\qquad\quad \mid \mathsf{AB_2}(\mathtt{x},\mathtt{y},\mathtt{z}).[\mathtt{x} = \mathtt{K_a}\ \mathtt{AND}\ \mathtt{y} = \mathtt{N_b}].\overline{\mathsf{RCHK_2}}\langle\{\mathtt{K_a},\mathtt{N_b}\}\,,\mathtt{z}\rangle \\
\qquad\qquad\quad \mid \mathsf{ROK_2}(\_).\overline{\mathsf{BOK}}\langle 1\rangle)
\end{array}
$$

The process family $\mathtt{P}$, then, is given by:

$$
\begin{array}{l}
\mathsf{k}(\mathtt{k}).\ \mathtt{LET\ t} = \mathtt{rand}(\mathtt{k})\ \mathtt{IN} \\
\quad (\nu \mathsf{RS_1},\mathsf{RND_1},\mathsf{RCHK_1},\mathsf{ROK_1},\mathsf{RS_2},\mathsf{RND_2},\mathsf{RCHK_2},\mathsf{ROK_2}). \\
\quad (\mathtt{RS}(1) \mid \mathtt{RS}(2) \mid \mathtt{A} \mid \mathtt{B})
\end{array}
$$

Note that all communication to and from the random number servers is done via private channels, representing the fact that f_t is a shared secret between A and B. Thus, the only messages sent on publicly available channels correspond precisely to those present in the original informal description of the protocol.

4.2 Specification

The specification of the protocol is similar to the original protocol. However, we wish to express the fact that the two parties have authenticated each other. To this end, each process transmits the value it *believes* to be the other's nonce back to the originating party on a private channel. The recipient of this message then compares the nonce it sent to this value, thereby ensuring that no messages have been altered by the adversary in transit.

Note that in this particular protocol there is no attempt to ensure the secrecy of any data. Instead, the goal is authenticity. Therefore, the only way an adversary can "win" is to alter some of the messages passing back in forth, and thereby cause both A and B to send messages on the AOK and BOK channels when they should not. If the adversary were able to accomplish such a feat, it would indicate that that A and B could not distinguish a conversation with one-another from a conversation with a malicious adversary. In other words, they could not be assured of the authenticity of the messages they received.

We modify the above protocol fragment for A to become $\mathtt{A}_{\mathtt{spec}}$ as follows:

$$
\begin{array}{l}
\mathtt{LET}\ \mathtt{N_a} = \mathtt{rand(k)}\ \mathtt{IN} \\
\quad \overline{\mathsf{AB}_1}\langle \mathtt{N_a} \rangle \\
\quad \mid \mathsf{BA}(\mathtt{w}, \mathtt{x}, \mathtt{y}, \mathtt{N_b}, \mathtt{z}).[\mathtt{w} = \mathtt{K_b}\ \mathtt{AND}\ \mathtt{x} = \mathtt{K_a}\ \mathtt{AND}\ \mathtt{y} = \mathtt{N_a}]. \\
\qquad\qquad (\ \overline{\mathsf{RCHK}_1}\langle \{\mathtt{K_b}, \mathtt{K_a}, \mathtt{N_a}, \mathtt{N_b}\}, \mathtt{z} \rangle \\
\qquad\qquad \mid \mathsf{ROK}_1(\_).(\ \overline{\mathsf{RS}_2}\langle \{\mathtt{K_a}, \mathtt{N_b}\} \rangle \\
\qquad\qquad\qquad\qquad \mid \mathsf{RND}_2(\mathtt{r}).\overline{\mathsf{AB}_2}\langle \mathtt{K_a}, \mathtt{N_b}, \mathtt{r} \rangle \\
\qquad\qquad\qquad\qquad \mid \mathsf{BAP}(\mathtt{u}).[\mathtt{u} = \{\mathtt{K_b}, \mathtt{K_a}, \mathtt{N_a}\}].\overline{\mathsf{AOK}}\langle 1 \rangle \\
\qquad\qquad\qquad\qquad \mid \overline{\mathsf{ABP}}\langle \{\mathtt{K_a}, \mathtt{N_b}\} \rangle))
\end{array}
$$

Similarly, $\mathtt{B}_{\mathtt{spec}}$ is:

$$
\begin{array}{l}
\mathtt{LET}\ \mathtt{N_b} = \mathtt{rand(k)}\ \mathtt{IN} \\
\quad \mathsf{AB}_1(\mathtt{N_a}).\ (\overline{\mathsf{RS}_1}\langle \{\mathtt{K_b}, \mathtt{K_a}, \mathtt{N_a}, \mathtt{N_b}\} \rangle \\
\qquad\qquad \mid \mathsf{RND}_1(\mathtt{r}).\overline{\mathsf{BA}}\langle \mathtt{K_b}, \mathtt{K_a}, \mathtt{N_a}, \mathtt{N_b}, \mathtt{r} \rangle \\
\qquad\qquad \mid \overline{\mathsf{BAP}}\langle \{\mathtt{K_b}, \mathtt{K_a}, \mathtt{N_a}\} \rangle \\
\qquad\qquad \mid \mathsf{AB}_2(\mathtt{x}, \mathtt{y}, \mathtt{z}).[\mathtt{x} = \mathtt{K_a}\ \mathtt{AND}\ \mathtt{y} = \mathtt{N_b}].\overline{\mathsf{RCHK}_2}\langle \{\mathtt{K_a}, \mathtt{N_b}\}, \mathtt{z} \rangle \\
\qquad\qquad \mid \mathsf{ROK}_2(\_).\mathsf{ABP}(\mathtt{u}).[\mathtt{u} = \{\mathtt{K_a}, \mathtt{N_b}\}].\overline{\mathsf{BOK}}\langle 1 \rangle)
\end{array}
$$

The full process family $\mathtt{P}_{\mathtt{spec}}$ is then:

$$
\begin{array}{l}
\mathsf{k}(\mathtt{k}).\ \mathtt{LET}\ \mathtt{t} = \mathtt{rand(k)}\ \mathtt{IN} \\
\quad (\nu \mathsf{RS}_1, \mathsf{RND}_1, \mathsf{RCHK}_1, \mathsf{ROK}_1, \mathsf{RS}_2, \mathsf{RND}_2, \mathsf{RCHK}_2, \mathsf{ROK}_2, \mathsf{ABP}, \mathsf{BAP}). \\
\quad (\mathtt{RS}(1) \mid \mathtt{RS}(2) \mid \mathtt{A}_{\mathtt{spec}} \mid \mathtt{B}_{\mathtt{spec}})
\end{array}
$$

The key idea in the specification process is that $\mathtt{A_{spec}}$ and $\mathtt{B_{spec}}$ send the values they receive on public channels back to each other on private channels. In this way, the recipient of the private message can verify that the message she sent earlier was accurately received. Thus, the specification ensures that any tampering on the part of the adversary will cause the protocol to fail. This technique is our standard approach for writing specifications for authentication protocols. For other protocols, one can specify secrecy alone, or specify secrecy and authenticity together [LMMS98].

4.3 Equivalence

The observational equivalence of the implementation $\mathtt{P}$ is to its specification $\mathtt{P_{spec}}$ may be shown in two stages. On the one hand, it may be shown that $\mathtt{P_{spec}}$ is observationally equivalent to $\mathtt{P'_{spec}}$, which is just like $\mathtt{P_{spec}}$, but without the checks introduced on the private channels. On the other hand, it may be shown that $\mathtt{P'_{spec}}$ is observationally equivalent to $\mathtt{P}$. We briefly illustrate both arguments.

Suppose $\mathtt{P_{spec}}$ is not observationally equivalent to $\mathtt{P'_{spec}}$. Let $\mathcal{C}$ be a distinguishing context. Consider, for instance, an observation $\overline{\mathsf{AOK}}\langle 1\rangle$. Suppose that $\mathtt{u} \neq \{\mathtt{K_b}, \mathtt{K_a}, \mathtt{N_a}\}$ in the check introduced in $\mathtt{A_{spec}}$. Then, since BAP is private, it must be the case that the value received on $\mathsf{AB_1}$ by $\mathtt{B_{spec}}$ was some value $\alpha \neq \mathtt{N_a}$ or that at least one of the first two components of $\mathtt{u}$ is not a correct public key. We consider only the former case; the latter case is similar. Since $\mathtt{A_{spec}}$ checks that $w = K_b$ and $x = K_a$ and $y = N_a$, and since $\mathtt{B_{spec}}$ faithfully relays α on BA, it must be the case that the context has intercepted this message from $\mathtt{B_{spec}}$, and replaced it with a new message. It may be shown that this contradicts the main hypothesis on f_t, since the chance that the context is able to find $\delta, \delta', \beta, \gamma$ such that $\gamma = f_t(\delta, \delta', \alpha, \beta)$ is at most $1/2^k$. A similar argument applies to the check present in $\mathtt{B_{spec}}$.

An important step in obtaining the equivalence between $\mathtt{P'_{spec}}$ and $\mathtt{P}$ is that, up to exponentially small probability of guessing the keys provided by the servers, the writes on private channels ABP and BAP are ready whenever the reads are. This fact is derivable from examination of the data flow in $\mathtt{P_{spec}}$.

5 Conclusion and Future Directions

The work presented here sits between the existing disciplines of protocol analysis, where cryptography is presumed perfect and is treated as a black box, and cryptography, where exact requirements coming from specific protocols are not addressed. Our work bridges this gap, providing a structure within which protocol security can be reduced directly to well-known cryptographic assumptions. Our work, building on the spi-calculus and Dolev-Yao adversary model, expands the modeled capabilities of the attacker to include operations such as guessing a secret key. In order to restrain such attackers from naive but impractical guessing attacks on exponentially large keyspaces, we must restrict them to polynomial time. Furthermore, since there is a chance the attacker may guess correctly, we must reason about the probability of successful attack.

Our framework uses a process calculus for defining probabilistic polynomial-time processes communicating over a network in such a way as to allow an adversarial process access to read and manipulate the various communications. Security properties of a given protocol may be formulated in our framework by writing another, idealized protocol and showing that the environment behaviors, which represent definable adversaries, have the same observable interactions with either protocol. For this purpose we propose a definition of observational equivalence for probabilistic programs that is based on the view that large differences in probability are easier to observe than small differences. When we distinguish between "large" and "small" using asymptotic behavior, we arrive at a definition of observational equivalence that coincides with a standard concept from cryptography, namely, indistinguishability by polynomial-time statistical tests [Yao82], and which enjoys certain important properties such as transitivity.

The steps taken in this paper and those before form the basis for a larger program that we hope to see carried out over the next few years. In part following the program established in the study of spi-calculus [AG97], we hope to develop methods for reasoning about observational equivalence (or some approximation to observational equivalence such as probabilistic trace equivalence or bisimulation) and use these methods to establish security properties of various protocols. We hope that an effective set of principles and proof rules are developed in this vein. We also hope that certain foundational questions about probabilistic process calculus can be addressed in the near term. Work like that of Volpano and Smith [VS98] and Kozen [Koz81] may be of direct relevance here. We also hope to generalize our main results to include contexts with multiple holes and contexts with bounded replication. These generalizations would provide more coverage of feasible attack scenarios. Finally, we have begun to use our framework to provide rigorous, uniform definitions of traditional cryptographic concepts, like chosen plaintext attacks, chosen ciphertext preprocessing attacks, and chosen ciphertext postprocessing attacks.

In sum, this paper provides a framework for reasoning about more detailed properties of protocols than previous analysis tools and methods have allowed. We have shown how this framework can be applied to a recent authentication protocol of Bellare-Rogaway. This paper also presents a detailed model of observational equivalence which refines earlier research on spi-calculus, and presents several key properties of our framework which we employ in the analysis of the Bellare-Rogaway example.

Acknowledgements: Thanks to M. Abadi, D. Boneh, C. Dwork, S. Kannan, C. Meadows, and M. Naor for helpful discussions.

References

[AG97] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. In *Proc. 4th ACM Conference on Computer and Communications*

Security, pages 36–47, 1997. Revised and expanded versions to appear in *Information and Computation* and 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.

[BB90] G. Berry and G. Boudol. The chemical abstract machine. In *Proc. 17th ACM Symp. Principles of Programming Languages*, pages 81–94, 1990.

[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.

[BR95] M. Bellare and P. Rogaway. Provably secure session key distribution - the three party case. In *Proc. 27th ACM Symposium on the Theory of Computing*, 1995.

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

[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.

[Koz81] D. Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22:328–350, 1981.

[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.

[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.

[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.

[Mil92] R. Milner. Functions as processes. *Math. Structures in Computer Science*, 2(2):119–141, 1992.

[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.

[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.

[VS98] D. Volpano and G. Smith. Probabilistic noninterference in a concurrent language. In *11th IEEE Computer Security Foundations Workshop*, pages 34–43. IEEE Computer Soc Press, 1998.

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