Probabilistic Bisimulation and Equivalence for Security Analysis of Network Protocols *

Ajith Ramanathan[1], John Mitchell[1], Andre Scedrov[2]**, and Vanessa Teague[1]

[1] Stanford University
[2] University of Pennsylvania

Abstract. Using a probabilistic polynomial-time process calculus designed for specifying security properties as observational equivalences, we develop a form of bisimulation that justifies an equational proof system. This proof system is sufficiently powerful to derive the semantic security of El Gamal encryption from the Decision Diffie-Hellman (DDH) assumption. The proof system can also derive the converse: if El Gamal is secure, then DDH holds. While these are not new cryptographic results, these example proofs show the power of probabilistic bisimulation and equational reasoning for protocol security.

1 Introduction

While so-called Dolev-Yao-style [9, 23] models that use nondeterminism and idealized cryptography have proven useful (*e.g.*, [6, 27, 24, 10]), combining nondeterminism with bit-level representation of encryption keys renders any encryption function insecure [18]. We therefore explore a probabilistic polynomial-time process calculus framework [18, 22, 19] for protocol analysis that is formal, yet close to the mathematical setting of modern cryptography and other recent work on compositional reasoning at a cryptographic level [7, 25], In this approach, we may reason about the security of protocols by quantifying over all "adversarial" processes definable in the language. In addition, the probabilistic process language lets us analyze probabilistic encryption functions, such as El Gamal [11], and protocols, using security requirements that have become accepted in the field of cryptography.

In the probabilistic polynomial-time calculus, security properties are specified as asymptotic observational equivalences. Specifically, $\mathcal{P} \cong \mathcal{Q}$ means that for any context $\mathcal{C}[\ \ ]$, the behavior of process $\mathcal{C}[\mathcal{P}]$ is asymptotically computationally indistinguishable from the behavior of process $\mathcal{C}[\mathcal{Q}]$. If $\mathcal{P}$ is a protocol of interest, and $\mathcal{Q}$ is an idealized form of the process that uses private channels to guarantee authentication and secrecy, then $\mathcal{P} \cong \mathcal{Q}$ is a succinct way of asserting that $\mathcal{P}$ is secure. We have found this approach, also used in [2, 7, 25], effective not only for specifying security properties of common network protocols, but

* Partially supported by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing," ONR Grant N00014-01-1-0795.

** Additional support from NSF Grant CCR-0098096.

also for stating common cryptographic assumptions. For this reason, we believe it is possible to prove protocol security from cryptographic assumptions using equational reasoning. The possibility is realized in this paper by proving security of El Gamal encryption from the standard Decision Diffie-Hellman assumption, and conversely.

Several advances over our previous efforts [18, 22, 19] were needed to make these formal equational proofs possible. First, we have refined the operational semantics of our process calculus. Most importantly, we define protocol execution with respect to any probabilistic scheduler that runs in polynomial time and operates uniformly over certain kinds of choices (to avoid unrealistic collusion between the scheduler and a protocol attacker), and we give priority to private ("silent") actions by executing private actions simultaneously in parallel before public communication. Second, we develop a form of probabilistic bisimulation that, while not a complete characterization of asymptotic observational equivalence, gives a tractable approximation. Third, we present an equational proof system and prove its soundness using bisimulation. The power of this equational proof system is shown by proving the semantic security of El Gamal encryption from the Decision Diffie-Hellman assumption.

Previous literature on probabilistic process calculi includes, *e.g.*, [20, 28]. The closest technical precursor of our process calculus is [2], which uses observational equivalence and channel abstraction but does not involve probability or computational complexity bounds; subsequent related work is cited in [1], for example. Prior work on CSP and security protocols, *e.g.*, [27], also uses process calculus and security specifications in the form of equivalence or related approximation orderings on processes. An important parallel effort with goals similar to our work, the paradigm of "universally composable security", can be found in [7]. Some connections between the probabilistic polynomial-time process calculus and universal composability are discussed in [19]. A full version of the present condensed conference paper is in preparation and will be available on the web [26].

2 The Probabilistic Process Calculus

We will write $F\colon X \times Y \to [0,1]$ for a probabilistic function from X to Y, and say that F is *stochastic* if $\forall x \in X$ the finite sum $\sum_{y \in Y} F(x,y)$ equals 1. Given an equivalence relation R over the set X, we will write $[x]_R$ for the equivalence class of x in X and write (X/R) for the set of all equivalence classes of X. For a multiset X, the equivalence class $[x]_R$ of x with respect to R is a multiset consisting of all elements equivalent to x under R, each with the same multiplicity in $[x]_R$ as in X. We refer to standard sources such as [3] for a discussion of probabilistic poly-time Turing machines and functions.

2.1 Syntax

Our probabilistic process calculus consists of a set of *terms* that do not perform any communications, *expressions* that can communicate with other expressions, and, *channels* that are used for communication. Let Var be a countable set of variable names, $Channel$ a countable set of channel names, and $Poly = \{q \colon \mathbb{N} \to \mathbb{N} | \, \forall a \in \mathbb{N} \colon q(a) > 0\}$ the set of positive polynomials in one variable. We note that each channel name has a bandwidth polynomial associated with it by a function we shall call σ. The security parameter used in cryptographic analysis is represented by a distinguished constant $\mathbf{n}$, discussed in more detail further on. We use $\equiv$ for syntactic identity.

We assume the existence of a class of *terms* Θ such that:

1. If θ is a term with k variables, then there exists a probabilistic Turing machine M_θ with k inputs and a polynomial $q_\theta(x_1, \dots, x_k)$ such that:
   (a) The term θ, with $a_1, \dots, a_k$ substituted for its k variables, reduces to a with probability p if and only if $M_\theta(a_1, \dots, a_k)$ returns a with probability p; and,
   (b) For any choice of $a_1, \dots, a_k$ we have that $M_\theta(a_1, \dots, a_k)$ halts in time at most $q_\theta(|a_1|, \dots, |a_k|)$.
2. For each probabilistic poly-time function $f \colon \mathbb{N}^m \to \mathbb{N}$, there exists a term θ such that M_θ computes f.

One example of such a set of terms is based on a term calculus called OSLR studied in [21] (based in turn on [4, 16]).

Expressions of the probabilistic process calculus (PPC) are given by the following grammar:

$$
\begin{array}{llr}
\mathcal{P} ::= & \mathbf{0} & \text{(termination)} \\
 & \nu_c(\mathcal{P}) & \text{(private channel)} \\
 & \mathtt{in}\,[c, x]\,.(\mathcal{P}) & \text{(input)} \\
 & \mathtt{out}\,[c, T]\,.(\mathcal{P}) & \text{(output)} \\
 & [T_1 = T_2].(\mathcal{P}) & \text{(match)} \\
 & (\mathcal{P} \mid \mathcal{P}) & \text{(parallel composition)} \\
 & !_{q(\mathbf{n})}.(\mathcal{P}) & \text{(bounded replication)}
\end{array}
$$

Intuitively $\mathbf{0}$ is the *zero expression* having no transitions. An input operator $\mathtt{in}\,[c, x]\,.\mathcal{P}$ waits until it receives a value on the channel c and then substitutes that value for the variable $x \in Var$ in $\mathcal{P}$. Similarly, an output $\mathtt{out}\,[c, T]\,.\mathcal{P}$ evaluates the term T, transmits that value on the channel c, and then proceeds with $\mathcal{P}$. Channel names that appear in an input or an output operation can be either public or private, with a channel being private if it is bound by a ν-operator and public otherwise. For convenience we will α-rename channel names so that they are all distinct. The match operator evaluates the expression bound to it if and only if both the terms making up the match evaluate to the same atom. Otherwise the entire match expression evaluates to the zero process. We assume that the probabilistic parallel composition operator $\mid$ associates to the left. The bounded replication operator has bound determined by the polynomial $q \in Poly$ affixed as a subscript. The expression $!_{q(\mathbf{n})}.(\mathcal{P})$ is expanded to the $q(\mathbf{n})$-fold parallel composition $\mathcal{P} \mid \cdots \mid \mathcal{P}$ before evaluation. We will write $\mathtt{out}\,[c, T]$ as an abbreviation for the expression $\mathtt{out}\,[c, T]\,.(\mathbf{0})$.

An expression $\mathcal{P}$ generated by this grammar may contain the distinguished constant $\mathbf{n}$. Substituting a value drawn from $\mathbb{N}$ for $\mathbf{n}$ gives rise to *processes*. In particular if $\mathcal{P}$ is an expression, then the process obtained by substituting i for all occurrences of $\mathbf{n}$ in $\mathcal{P}$ is denoted $P^{\mathbf{n}\leftarrow i}$ *i.e.*, $P^{\mathbf{n}\leftarrow i} \equiv [i/\mathbf{n}]\mathcal{P}$. An expression $\mathcal{P}$ can be thought to define the set of processes $\{P^{\mathbf{n}\leftarrow i} | \, i \in \mathbb{N}\}$. If we wish to denote a process without making the value of the security parameter explicit, we will just drop the superscript and write P (for a process obtained from the expression $\mathcal{P}$).

We use *Expr* for the set of expressions and *Proc* for the set of all processes. We will denote the set of all variable-closed processes by *CProc*. Let $\mathcal{P}$ be an open PPC expression and ξ a valuation of the free variables of $\mathcal{P}$ in $\mathbb{N}$. Then $\mathcal{P}(\xi)$ denotes the result of substituting, for each free variable x, the atom $\xi(x)$ for all occurrences of x in $\mathcal{P}$.

A *context* is an expression with numbered "holes" (indicated by empty square brackets $[\ \ ]_k$). The numerical subscripts serve to uniquely identify the holes. If we express protocols as expressions, then we can use contexts to express adversarial environments in which the protocols execute. As for expressions, it follows from the presence of the security parameter $\mathbf{n}$ in a context $\mathcal{C}$ that we can view $\mathcal{C}[\ \ ]_{k_1} \cdots [\ \ ]_{k_m}$ as defining the set $\{C^{\mathbf{n}\leftarrow i}[\ \ ]_{k_1} \cdots [\ \ ]_{k_m} | \, i \in \mathbb{N}\}$. We remind the reader that the security parameter appears in the polynomial bounding replication operators, in terms, and in the bandwidth polynomials associated with channels. We write *Con* for the set of all contexts.

2.2 Operational Semantics

The evaluation of a variable-closed process proceeds in three steps: reduction, selection, and communication. In the *reduction step*, all unblocked terms and matches are evaluated, where a term or match is *unblocked* if it appears in the body P of a process $\mathtt{in}\,[c, x]\,.\mathcal{P}$ requiring input.

In the *selection step*, we use a probabilistic scheduler to select an action to perform. Actions include the silent action, τ; the input action $\mathtt{in}\langle c, a\rangle$ that reads the value a from the channel c into the variable x; the output action $\mathtt{out}\langle c, a\rangle$ that places the value a on the channel c; and the simultaneous action $\alpha \cdot \beta$ obtained by using the action product $\cdot$ on α and β. The observable of an action is simply the pair of value and channel name associated with the action. We will say that two actions are of the same type, written $\alpha \sim \beta$, when the observables generated by them are the same. A *scheduler* is a stochastic probabilistic poly-time function that probabilistically maps a set of actions to type of action and always selects the silent (τ) type when the input set of actions contains a silent action. Let $\mathbf{S}$ be the set of all schedulers.

In the *communication step*, we pick an action of the chosen type uniformly at random from the set of actions of that type that the process can take. Having picked the particular action, we then perform the associated substitution of values for variables. In doing this, we take care to truncate the value according to the bandwidth polynomials associated with the channel name. By truncating the values substituted, we ensure that we never write down values that are exponentially long.

We call this three-stage procedure an *evaluation step*; and evaluation proceeds in evaluation steps until the set of schedulable actions becomes empty.

Theorem 1. *Let P be a process. Then the evaluation of P can be performed in time polynomial in the security parameter.*

The proof proceeds by constructing a machine that evaluates P. Obtaining the required time-bound relies on terms and schedulers both being probabilistic polytime Turing machines. We can also prove the converse of this theorem, omitted due to space constraints.

3 Probabilistic Bisimulation

In this section we develop a form of *weak probabilistic bisimulation*, or more simply *probabilistic bisimulation*, adapted from [28], which studies various approaches to probabilistic bisimulation and provides an elegant treatment.

Our calculus and intended application to security protocols presents several challenges that are absent from [28]. Silent actions are made simultaneous so as to avoid the $|$-operator changing the probability of silent actions. Communications across public channels yield compound public (not silent) actions since protocol communications are subject to interference by an adversary. As a result, the semantics of our $|$-operator cannot be easily reduced to the semantics given to standard nondeterministic composition, probabilistic summation, or probabilistic product.

We refer to a sequence of actions as a *path* and a sequence of silent actions terminated by an α-step as an *α-path*. If α is public, then an α-path must have length ≥ 1; if α is a silent action, then an α-path can have length 0. A zero-length path is called an *empty path* and a τ-path is called a *silent path*.

Definition 2. *Let $Paths(P, \alpha, \mathfrak{R})$ be the set of α-paths from P to some process in $\mathfrak{R}$ that are minimal with respect to $\mathfrak{R}$, i.e., the α-paths that are not proper extensions of some other α-path into $\mathfrak{R}$. For $\pi \in Paths(P, \alpha, \mathfrak{R})$, $Prob\big[\pi, S\big]$ denotes the probability of the process P taking the path π under the scheduler S and is formally defined in [26].*

We will reason about processes using a *cumulative probability distribution function* (cPDF). Intuitively, given a scheduler S, the cPDF $\mu(P, \alpha, \mathfrak{R}, S)$ measures the total probability that the process P can take an α-path under the scheduler S to reach a process in the set $\mathfrak{R}$. For details we refer the reader to [26].

Definition 3. *The cPDF $\mu\colon CProc \times Act \times 2^{Proc} \times \mathbf{S} \to [0, 1]$ is defined by*

$$\mu(P, \alpha, \mathfrak{R}, S) = \sum_{\pi \in Paths(P,\alpha,\mathfrak{R})} Prob\big[\pi, S\big] \tag{1}$$

Lemma 4. $\forall P \in CProc. \forall \alpha \in Act. \forall \mathfrak{R} \subseteq CProc. \forall S \in \mathbf{S}\colon \mu(P, \alpha, \mathfrak{R}, S) \leq 1$

The proof is by induction on the length of α-paths.

Definition 5. *An equivalence relation $R \subseteq Proc \times Proc$ is a* weak probabilistic bisimulation*, or* bisimulation*, if $(P, Q) \in R$ implies that*

$$\forall \mathfrak{U} \in (Proc/R).\forall S \in \mathbf{S}.\forall \alpha \in Act\colon \mu(P, \alpha, \mathfrak{U}, S) = \mu(Q, \alpha, \mathfrak{U}, S) \tag{2}$$

Two processes P and Q are bisimulation equivalent *(denoted $P \simeq Q$) if there exists a bisimulation R such that $(P, Q) \in R$. It immediately follows that $\simeq = \bigcup\{R \mid R \text{ is a bisimulation}\}$. We extend bisimulation to all processes by stipulating that $P, Q \in Proc$ are bisimilar iff they are bisimilar after any substitution of atoms for their free variables. We extend bisimulation equivalence to expressions by stipulating that $\mathcal{P}, \mathcal{Q} \in Expr$ are bisimilar iff $\forall i \in \mathbb{N}\colon P^{\mathbf{n} \leftarrow i} \simeq Q^{\mathbf{n} \leftarrow i}$.*

As a traditional sanity check, we can show that $\simeq$ is the *largest* bisimulation over $Proc$ using a fixed-point argument as in [20].

Theorem 6. $\forall P, Q \in Proc.\forall C[\ \ ] \in Con\colon P \simeq Q \implies C[P] \simeq C[Q]$

As in [28], the proof uses the cumulative probability distribution function μ instead of reasoning directly about the underlying transitions. The proof is structured around two inductions: one on the maximum number of free variables in $C[P]$ and $C[Q]$ and another on the structure of processes. For $\mid$ we show that $\mu(P \mid Q, \alpha, \mathfrak{R}_1 \mid \mathfrak{R}_2, S) = \sum_{\beta \cdot \gamma \sim \alpha} \mu(P, \beta, \mathfrak{R}_1, S) \cdot \mu(P, \gamma, \mathfrak{R}, S)$ by exploiting the fact that $P \simeq Q$ implies that $\forall \mathfrak{R} \in (Proc/\simeq)\colon \mu(P, \tau, \mathfrak{R}, S) = \mu(Q, \tau, \mathfrak{R}, S)$. For ν we exploit the fact that a silent action consists of simultaneously performing every concrete action (an action consisting of an input and output on the same channel and transmitting the same value) on a private channel that can go without interfering with the other private actions.

4 Asymptotic Observational Equivalence

4.1 Definition of Observational Equivalence

Intuitively, we wish to consider two closed expressions equivalent if they behave indistinguishably in the presence of any adversary, where we represent an adversary by a context and a scheduler. We are most interested in protocols that use cryptographic primitives, and cryptographic primitives generally depend on some security parameter, such as the length of the encryption key. The purpose of the security parameter, as opposed to a fixed key length, is that it can be made as large as needed for any desired degree of security. Therefore, while the adversary has control over the context (surrounding environment) and scheduler, representing a degree of "good luck", we only consider the adversary successful if it can measure an asymptotically significant difference between processes as the security parameter increases.

Definition 7. *An* observable *o is a pair $\langle c, a\rangle \in Channel \times \mathbb{N}$. Let* Obs *be the set of all observables.*

If $P \in Proc$ is a blocked process and $o = \langle c, a\rangle$ is an observable, then P *generates o under scheduler S*, written $P \rightsquigarrow_S o$, if an action equivalent to $\mathtt{in}\langle c, a\rangle \cdot \mathtt{out}\langle c, a\rangle$ is selected by S and performed during the course of the evaluation of P. "Partial actions," consisting of an input action without a matching output, or an output without a matching input, appear in the structured operational semantics and are used to prove compositionality of $\simeq$. These partial actions are not primitive observables, but can be observed in a parallel context that provides the dual action. We only consider concrete actions (with an input matched by an output) here and write $Act^{\times}$ for the set of concrete actions.

The probability that P *generates an observable o under the scheduler S* is the probability that P can take a $(\mathtt{in}\langle c, a\rangle \cdot \mathtt{out}\langle c, a\rangle)$-path, under S, plus the probability that P can, under S, take an α-path (with α concrete but not equivalent to $\mathtt{in}\langle c, a\rangle \cdot \mathtt{out}\langle c, a\rangle$) to some process R times the probability that R generates o under S. If a path produces the observable o multiple times, then its contribution to $\mathrm{Prob}\big[P \rightsquigarrow_S o\big]$ only extends to the first occurrence of that observable along that path. Since we only care about whether a path produces an observable, not the number of times that it does so, we only count to the first observable generated by that path. An easy induction shows that $\mathrm{Prob}\big[P \rightsquigarrow_S o\big] \leq 1$.

We envision process evaluation as a proceeding in stages where each stage is determined by an input communicating with an output. While probabilistic bisimulation allowed partial actions (inputs and outputs communicating with the environment), actual evaluation proceeds via concrete actions (input-output pairs). Hence, we define observable behavior with respect to just those schedulers that always select concrete actions.

Definition 8. *A* perceptible scheduler *is a scheduler that only schedules private actions and concrete actions. We write* **SP** *for the set of perceptible schedulers.*

Let $\mathbb{Q}$ be the set of positive polynomials. Let Σ be the set of valuations of free variables. If $\sigma \in \Sigma$ is a valuation, then we denote the result of performing the valuation σ on $\mathcal{P}$ by $\sigma(\mathcal{P})$.

Definition 9 (observational equivalence). *Let $\mathcal{P}$ and $\mathcal{Q}$ be two expressions. We will say that $\mathcal{P} \cong \mathcal{Q}$, or that they are* observationally equivalent*, if:*

$$\forall q(y) \in \mathbb{Q}.\forall \sigma \in \Sigma.\forall \mathcal{C}[\;\;] \in Con.\forall o \in Obs.\forall S \in \mathbf{SP}.\exists i_o \in \mathbb{N}.\forall i > i_o :$$
$$\Big|Prob\big[\sigma(C^{\mathbf{n}\leftarrow i}[P^{\mathbf{n}\leftarrow i}]) \rightsquigarrow_S o\big] - Prob\big[\sigma(C^{\mathbf{n}\leftarrow i}[Q^{\mathbf{n}\leftarrow i}]) \rightsquigarrow_S o\big]\Big| \leq \frac{1}{q(i)} \quad (3)$$

Theorem 10. $\cong$ *is a congruence.*

Using Theorem 6 and the definition of $\simeq$, we can prove

Theorem 11. $\mathcal{P} \simeq \mathcal{Q} \implies \mathcal{P} \cong \mathcal{Q}$.

4.2 A Proof System

A proof system for proving asymptotic observational equivalence is given in Figure 1. The soundness of this proof system is established in three ways. Axioms

$$\mathcal{P} \mid \mathcal{Q} \cong \mathcal{Q} \mid \mathcal{P} \tag{P1}$$

$$\mathbf{0} \mid \mathcal{P} \cong \mathcal{P} \tag{P2}$$

$$(\mathcal{P} \mid \mathcal{Q}) \mid \mathcal{R} \cong \mathcal{P} \mid (\mathcal{Q} \mid \mathcal{R}) \tag{P3}$$

$$\frac{\mathcal{P}_1 \cong \mathcal{P}_2, \mathcal{Q}_1 \cong \mathcal{Q}_2}{\mathcal{P}_1 \mid \mathcal{Q}_1 \cong \mathcal{P}_2 \mid \mathcal{Q}_2} \tag{P4}$$

$$\frac{c \notin Channel(\mathcal{P}), x \notin FreeVars(\mathcal{P})}{\mathcal{P} \cong \nu_c(\mathtt{out}\,[c, T] \mid \mathtt{in}\,[c, x]\,.\mathcal{P})} \tag{NU1}$$

$$\frac{\begin{array}{c}\mathcal{C}[\mathtt{out}\,[c, T]] \text{ is scheduler-insensitive,} \\ c \notin Channel(\mathcal{C}[\mathbf{0}]), Public(\mathcal{C}[\mathtt{out}\,[c, T]]) = \{c\}\end{array}}{\exists T_{\mathcal{C}} : \mathtt{out}\,[c, T_{\mathcal{C}}] \cong \mathcal{C}[\mathtt{out}\,[c, T]]} \tag{NU2}$$

$$\frac{\mathcal{P} \text{ has no public channels}}{\mathcal{P} \cong \mathbf{0}} \tag{ZER}$$

$$\frac{\mathcal{P} \cong \mathcal{Q}, \mathcal{C}[\ \ ] \in Con_1}{\mathcal{C}[\mathcal{P}] \cong \mathcal{C}[\mathcal{Q}]} \tag{CON}$$

$$\frac{\mathcal{P} \cong \mathcal{Q}, \mathcal{Q} \cong \mathcal{R}}{\mathcal{P} \cong \mathcal{R}} \tag{TRN}$$

$$\frac{\mathcal{P} \cong \mathcal{Q}}{\mathcal{Q} \cong \mathcal{P}} \tag{SYM}$$

$$\frac{\sigma(c) = \sigma(d)}{\nu_c(\mathcal{P}) \cong \nu_d(\mathcal{P}^{[d/c]})} \tag{R1}$$

$$\frac{\begin{array}{c}\sigma(c) = \sigma(d), \\ d \notin Channel(\mathcal{P}, \mathcal{Q}), \mathcal{P} \cong \mathcal{Q}\end{array}}{\mathcal{P}^{[d/c]} \cong \mathcal{Q}^{[d/c]}} \tag{R2}$$

$$\frac{f_T \text{ and } f_U \text{ are computationally indistinguishable}}{\mathtt{out}\,[c, T] \cong \mathtt{out}\,[c, U]} \tag{EQ1}$$

$$\frac{\forall i \in [1, k] : \mathtt{out}\,[c_i, T_i] \cong \mathtt{out}\,[c_i, U_i]}{\mathtt{out}\,[d, V(T_1, \ldots, T_k)] \cong \mathtt{out}\,[d, V(U_1, \ldots, U_k)]} \tag{EQ2}$$

$$\frac{\begin{array}{r}\forall a_1, \ldots, a_k : \mathtt{out}\,[c_i, U_i(a_1, \ldots, a_k)] \cong \mathtt{out}\,[c_i, V_i(a_1, \ldots, a_k)]\,, i \in \{1, m\} \\ FV(\mathcal{C}[\mathtt{out}\,[c_1, U_1(x_1, \ldots, x_k])] \cdots [\mathtt{out}\,[c_m, U_m(x_1, \ldots, x_k])]]) = \\ FV(\mathcal{C}[\mathtt{out}\,[c_1, V_1(x_1, \ldots, x_k])] \cdots [\mathtt{out}\,[c_m, V_m(x_1, \ldots, x_k])]]) = \\ \{x_i\}\end{array}}{\begin{array}{r}\mathtt{in}\,[d, x_i]\,.\mathcal{C}[\mathtt{out}\,[c_1, U_1(x_1, \ldots, x_k)]] \cdots [\mathtt{out}\,[c_m, U_m(x_1, \ldots, x_k)]] \cong \\ \mathtt{in}\,[d, x_i]\,.\mathcal{C}[\mathtt{out}\,[c_1, V_1(x_1, \ldots, x_k)]] \cdots [\mathtt{out}\,[c_m, V_m(x_1, \ldots, x_k)]]\end{array}} \tag{PUL}$$

Fig. 1. A Reasoning System for PPC

such as `P1`, `NU2`, *etc.*, and certain structural inference rules like `P4` are justified by an application of Theorem 11. The soundness of rules `CON`, `TRN`, and, `SYM` follow from the congruence properties of $\cong$. The three rules `EQ1`, `EQ2`, and `PUL` are justified by reasoning directly about asymptotic equivalences. Detailed soundness arguments are given in [26]. We now continue with comments on selected rules.

A *scheduler-insensitive* process family is one in which the choice of scheduler does not matter *i.e.*, a process family for which at any evaluation-step only one kind of action can possibly be taken. Rule `NU2` states that if you have a scheduler-insensitive process family with only one output on a public channel, then the entire process family can be written as a single term placed in an output on the same channel. Essentially, this rule states the silent transitions are probabilistically invisible, a property that we were not able to achieve in earlier semantics for our calculus. The first of the two rules dealing with renaming channels, `R1` states that one can arbitrarily rename private channels (as long as bandwidths are respected). In this rule, $\mathcal{P}^{[d/c]}$ is taken to mean the closed expression obtained by replacing the channel name c with the channel name d (we define a similar notation for processes).

The second rule regarding renaming, `R2`, allows us to rename public channels to a name that is not currently in use by the expression. There is an additional technical restriction that ensures that the bandwidth associated with the new name is as big as the bandwidth associated with the old name.

The rule `PUL` asserts that if two functions $f_V \colon \mathbb{N}^k \times \mathbb{N} \to [0,1]$ and $g_U \colon \mathbb{N}^k \times \mathbb{N} \to [0,1]$ induce almost the same distribution on outputs, then we can "pull out" one of the arguments into an output.

5 Cryptographic Applications

Our asymptotic notion of observational equivalence between probabilistic poly-time processes allows us to express *indistinguishability by polynomial-time statistical tests*, a standard way of characterizing cryptographically strong pseudo-random number generators [29, 12]. In what follows, we will denote an element x chosen uniformly at random from the set X by $x \in_R X$.

Throughout this section we adopt a uniform-complexity model of the adversary, see [13] Ch. 5.

5.1 Computational Indistinguishability

Definition 12 (function ensemble [29, 12]). *A* function ensemble *f is an indexed family of functions $\{f_i \colon A_i \to B_i\}_{i \in \mathbb{N}}$. A function ensemble $f \colon A_i \to B_i$ is* uniform *if there exists a single Turing machine M that computes f for all values of i i.e., $M(i,x) = f_i(x)$. A uniform function ensemble $f \colon A_i \to B_i$ is* poly-time *if there exists a polynomial q and a single Turing machine M such that $M(i,x)$ computes $f_i(x)$ in time at most $q(|i|,|x|)$. A uniform function ensemble $f \colon A_i \to B_i$ is* probabilistic poly-time *if f_i is a probabilistic poly-time function. A* poly-time statistical test *$\mathcal{A}$ is the $\{0,1\}$-valued probabilistic poly-time function ensemble $\{\mathcal{A}_i \colon \{0,1\}^{m(i)} \to \{0,1\}\}$.*

The notion of computational indistinguishability is central to cryptography; [12], in particular, has an insightful discussion.

Definition 13 (computational indistinguishability [29, 12]). *Let $q(x)$ be a positive polynomial. A uniform probabilistic poly-time function ensemble $f\colon \{\} \to \{0,1\}^{l(x)}$ is* computationally indistinguishable *from a uniform probabilistic poly-time function ensemble $g\colon \{\} \to \{0,1\}^{l(x)}$ just when for all poly-time statistical tests $\mathcal{A}$ we have:*

$$\forall q(x).\exists i_o.\forall i > i_o\colon \big|Prob\big[\mathcal{A}_i(f_i()) = \text{“1”}\big] - Prob\big[\mathcal{A}_i(g_i()) = \text{“1”}\big]\big| \leq \frac{1}{q(i)} \quad (4)$$

Theorem 14. *Let $f\colon \{\} \to \{0,1\}^{l(x)}$ $(\forall x \in \mathbb{N}\colon l(x) > k(x))$ be a uniform probabilistic poly-time function ensemble. Let $g\colon \{\} \to \{0,1\}^{l(x)}$ be another uniform probabilistic poly-time function ensemble. Let $\mathcal{F} \equiv \mathtt{out}\,[c, f]$ and $\mathcal{G} \equiv \mathtt{out}\,[c, g]$. Then, f is computationally indistinguishable from g if and only if $\mathcal{F} \cong \mathcal{G}$.*

Assume that f is not computationally indistinguishable from g but that $\mathcal{F} \cong \mathcal{G}$. Then there exists a test A distinguishing f and g. But then the context $[\ \ ] \mid \mathtt{in}\,[c, x]\,.\mathtt{out}\,[d, A(x)]$ will distinguish $\mathcal{F}$ from $\mathcal{G}$. Similarly, assume that f is computationally indistinguishable from g but that $\mathcal{F} \not\cong \mathcal{G}$. Then there exists a context $\mathcal{C}[\ \ ]$ distinguishing $\mathcal{F} \not\cong \mathcal{G}$ on the basis of the observable o under scheduler S. We construct a test A as follows. To evaluate A on the value a we evaluate,under S, the expression, $\mathcal{C}[\mathtt{out}\,[c, a]]$ and return “1” if o is generated and “0” otherwise. Clearly, A will distinguish f from g.

We can immediately obtain, as a corollary to Theorem 14, the result from [22] showing that pseudorandom number generators can be represented in PPC.

5.2 Semantic Security

Semantic security is an important cryptographic property due to [15]. We use a definition for uniform complexity based on [13, 14]. We begin by summarizing the definition of a cryptosystem that can be found in full in [13] or [14], for example.

Definition 15. *[8, 13, 14] A* public-key encryption scheme *or, more simply, an* encryption scheme *is a triple $\langle G, E, D\rangle$ comprising a probabilistic poly-time key-generation algorithm G that produces a key pair from input 1^k (the security parameter written in unary), probabilistic poly-time encryption algorithm E, and a probabilistic poly-time decryption algorithm D.*

Intuitively, an encryption scheme is semantically secure if, given a ciphertext, no polynomially-bounded adversary can reliably compute information about the associated plaintext. Semantic security can also be stated using indistinguishability: intuitively, it is infeasible for any adversary to distinguish between the encryptions of any two messages, even when it chooses the messages. For our purposes it is convenient to work with security in the sense of indistinguishability; we follow [14].

Definition 16. *An encryption scheme* $\langle G, E, D\rangle$ *is* indistinguishably secure *if for all probabilistic poly-time Turing machines* F, A*, for every polynomial* q*, for sufficiently large* k*, and for all* m*:*

$$\Big|Prob\big[A(1^k, e, \langle m_0, m_1\rangle, c) = m|\, c \in E(e, m_0)\big] - Prob\big[A(1^k, e, \langle m_0, m_1\rangle, c) = m|\, c \in E(e, m_1)\big]\Big| \leq \frac{1}{q(k)} \quad (5)$$

with $\langle m_0, m_1\rangle$ *chosen probabilistically by running* $F(1^k, e)$*.*

In words, it is impossible to efficiently generate two messages (using F) such that an attack A can reliably distinguish between their encryptions. This definition reflects adaptive chosen plaintext semantic security since the adversary, in possession of the encryption key, can generate and encrypt a polynomial number of messages. The equivalence of security in the sense of indistinguishability and semantic security is well known in cryptographic circles; [13] Ch. 5 has a detailed treatment of both directions.

Encoding the statement of indistinguishable encryptions as an observational equivalence in PPC is straightforward. In what follows, we will use the notation $\mathtt{in}\,[c, \langle x_1, \ldots, x_k\rangle]$ to mean that the input obtained on channel c should be treated as a k-tuple whose ith element is named x_i.

Definition 17. *Let* $\langle G, E, D\rangle$ *be an encryption scheme. Then* $\langle G, E, D\rangle$ *is an* observationally indistinguishable encryption scheme *iff*

$$\nu_c(\mathtt{out}\,[c, pkey(G(1^{\mathbf{n}}))] \mid \mathtt{in}\,[c, key]\,.\mathtt{out}\,[\mathtt{pub}, \langle key, 1^{\mathbf{n}}\rangle]\,.\mathtt{in}\,[\mathtt{msg}, \langle m_0, m_1\rangle]\,.\ \mathtt{out}\,[\mathtt{challenge}, \langle key, \langle m_0, m_1\rangle, E(key, m_0)\rangle]) \quad (\mathcal{L}\text{-}\mathcal{SS})$$

is observationally indistinguishable from

$$\nu_c(\mathtt{out}\,[c, pkey(G(1^{\mathbf{n}}))] \mid \mathtt{in}\,[c, key]\,.\mathtt{out}\,[\mathtt{pub}, \langle key, 1^{\mathbf{n}}\rangle]\,.\mathtt{in}\,[\mathtt{msg}, \langle m_0, m_1\rangle]\,.\ \mathtt{out}\,[\mathtt{challenge}, \langle key, \langle m_0, m_1\rangle, E(key, m_1)\rangle]) \quad (\mathcal{R}\text{-}\mathcal{SS})$$

where pkey takes a private-public key-pair and returns only the public key.

An examination of the expression $\mathcal{L}$-$\mathcal{SS}$ shows that it

1. Generates an encryption-decryption key-pair,
2. Publishes the security parameter and the public key,
3. Obtains a message pair (that could be a function of the security parameter and the public key),
4. Publishes the encryption of the first message, along with the message pair and the encryption key.

Expression $\mathcal{R}$-$\mathcal{SS}$ is similar, but encrypts the second message.

Theorem 18. *Let* $\langle G, E, D\rangle$ *be an encryption scheme. Then,* $\langle G, E, D\rangle$ *is semantically secure iff* $\mathcal{L}$-$\mathcal{SS} \cong \mathcal{R}$-$\mathcal{SS}$*.*

5.3 The Decision Diffie-Hellman Assumption

The Decision Diffie-Hellman assumption [8] is an assumption about modular exponentiation. Our development draws from [5, 13].

A group family $\mathbb{G}$ is a set of finite cyclic groups $\{G_p\}$ where the index p ranges over an infinite set. An instance generator $IG(n)$ takes security parameter n, runs in time polynomial in n and returns a random index p as well as a generator g of the group G_p.

Definition 19. *A* Decision Diffie-Hellman algorithm *A for $\mathbb{G}$ is a probabilistic polynomial time algorithm such that:*

1. *Given $\langle p, g, g^a, g^b, g^c\rangle$ the algorithm A reliably decides if $c = ab$; and,*
2. *There exists a non-constant positive polynomial $q(\cdot)$ such that $IG(n) = \langle p, g\rangle$ implies that $|\langle p, g\rangle| = \Omega(q(n))$.*

The probability is taken over the probability that the instance generator $IG(1^n)$ returns $\langle p, g\rangle$ given n, random choice of a, b, c in $[1, \mathrm{ord}\, G_p]$ and random bits used by A. The Decision Diffie-Hellman assumption *(DDHA) for $\mathbb{G}$ is that no Decision Diffie-Hellman algorithm exists.*

This assumption is believed for some group families $\mathbb{G}$, but known to be false for others; see [5].

Definition 20. *The group family $\mathbb{G}$ is* observationally DDHA-secure *if*

$$\mathtt{out}\left[\mathtt{ch}, \langle p, g, g^a, g^b, g^{ab}\rangle \,|\, a, b \in_R [1, \mathrm{ord}\, G_p]\right] \cong \mathtt{out}\left[\mathtt{ch}, \langle p, g, g^a, g^b, g^{c}\rangle \,|\, a, b, c \in_R [1, \mathrm{ord}\, G_p]\right] \quad (\mathcal{DDHA})$$

where the term $\langle p, g, g^a, g^b, g^{ab}\rangle \,|\, a, b \in_R [1, \mathrm{ord}\, G_p]$ denotes the term that computes this tuple with a, b chosen uniformly at random from $[1, \mathrm{ord}\, G_p]$.

The following theorem shows that we can express the DDHA in PPC by an observational equivalence.

Theorem 21. *The DDHA holds for the group family $\mathbb{G}$ iff $\mathbb{G}$ is observationally DDHA-secure.*

5.4 The Semantic Security of El Gamal Encryption

In this section, we use PPC to show semantic security of El Gamal encryption.

Definition 22. *Let $\cdot$ denote group multiplication and $=$ denote group equality. An* El Gamal encryption scheme *is a triple $\langle G, E, D\rangle$ of probabilistic poly-time algorithms such that:*

1. *The key generating algorithm G, on input 1^k outputs a public key $e = \langle p, g, g^a\rangle$ and a private key $d = a$ where $\langle g, p\rangle \in IG(1^k)$ and $a \in_R [1, \mathrm{ord}\, G_p]$.*

2. *An encryption algorithm E that, on input, $e = \langle p, g, g^a \rangle$ and m outputs $\langle g^b, m \cdot g^{ab} \bmod p \rangle$ as the ciphertext (where $b \in_R [1, \operatorname{ord} G_p]$).*
3. *A decryption algorithm D that, given ciphertext $c = \langle k, c' \rangle$ and decryption key d computes c'/k^d. To see why this works, we note that $k = g^a$, $c' = m \cdot g^{ab} \bmod p$, and $d = b$ for some a, b, m. Then*

$$\frac{c'}{k^d} = \frac{m \cdot g^{ab}}{\left(g^b\right)^a} = \frac{m \cdot g^{ab}}{g^{ab}} = m \tag{6}$$

Let $\mathcal{L}\text{-}\mathcal{EG}$ and $\mathcal{R}\text{-}\mathcal{EG}$ be the result of instantiating expressions $\mathcal{L}\text{-}\mathcal{SS}$ and $\mathcal{R}\text{-}\mathcal{SS}$ to El Gamal encryption.

Theorem 23. *If the Decision Diffie-Hellman assumption holds for a group family $\mathbb{G}$, then El Gamal encryption using $\mathbb{G}$ is semantically secure. Furthermore, there is a formal equational proof of the equivalence $\mathcal{L}\text{-}\mathcal{EG} \cong \mathcal{R}\text{-}\mathcal{EG}$ stating that El Gamal encryption is semantically secure from the equivalence $\mathcal{DDHA}$ stating that the DDHA holds for $\mathbb{G}$.*

A detailed proof is given in [26]. It starts with the equivalence $\mathcal{DDHA}$ and build up the equivalence $\mathcal{L}\text{-}\mathcal{EG} \cong \mathcal{R}\text{-}\mathcal{EG}$ by systematically transforming the term that outputs a challenge instance of the DDHA. The proof can be split into two distinct parts. In the first part, we use mathematical facts about the group operation $\cdot$ in the group G_p to transform the DDHA challenge $\langle p, g, g^a, g^b, g^c \rangle$ into a tuple $\langle p, g, g^a, m_0, m_1, g^b, m_i \cdot g^{ab} \rangle$ that *almost* looks like a semantic security of El Gamal encryption challenge tuple. The remainder of the proof consists of purely structural transformations on the expressions in order to arrive at an equivalence between two expressions of the right form. We suggest that, in general, proofs in PPC can be separated into a large sequence of structural transformations required to achieve the right shape of the protocol, couple with a few transformations whose soundness are grounded in mathematical facts about the special nature of the problem. These special facts can be represented with special hypotheses (like $\mathcal{DDHA}$) and special inference rules. Taken with the structural rules of Figure 1 this would allow us to derive El Gamal's semantic security from the DDHA in an entirely mechanical manner.

Although we give a direct equational proof, we could also have used a backward proof search (exploiting the mechanizable nature of proofs in PPC) to work backward from the protocol to the conditions that the cryptographic primitives must satisfy.

Theorem 24. *If El Gamal encryption using the group family $\mathbb{G}$ is semantically secure, then the Decision Diffie-Hellman assumption holds for $\mathbb{G}$. Furthermore, there is an equational proof of the equivalence $\mathcal{DDHA}$ from the equivalence $\mathcal{L}\text{-}\mathcal{EG} \cong \mathcal{R}\text{-}\mathcal{EG}$ asserting El Gamal semantic security.*

Assuming $\mathcal{L}\text{-}\mathcal{EG} \cong \mathcal{R}\text{-}\mathcal{EG}$, we use the rule CON to obtain

$$\nu_{\texttt{pub}}(\nu_{\texttt{msg}}(\nu_{\texttt{challenge}}(\mathcal{L}\text{-}\mathcal{EG} \mid \texttt{in}\,[\texttt{pub}, \langle p, g, g^a\rangle]\,.\texttt{out}\,[\texttt{msg}, \langle 1, g^r\rangle \mid r \in_R [1, \operatorname{ord} G_p]]\,.$$
$$\texttt{in}\left[\texttt{challenge}, \langle\langle p, g, g^a\rangle, \langle m_0, m_1\rangle, \langle g^b, g^c\rangle\rangle\right].\texttt{out}\left[\texttt{ddh}, \langle p, g, g^a, g^b, g^c\rangle\right])))$$
$$\cong$$
$$\nu_{\texttt{pub}}(\nu_{\texttt{msg}}(\nu_{\texttt{challenge}}(\mathcal{R}\text{-}\mathcal{EG} \mid \texttt{in}\,[\texttt{pub}, \langle p, g, g^a\rangle]\,.\texttt{out}\,[\texttt{msg}, \langle 1, g^r\rangle \mid r \in_R [1, \operatorname{ord} G_p]]\,.$$
$$\texttt{in}\left[\texttt{challenge}, \langle\langle p, g, g^a\rangle, \langle m_0, m_1\rangle, \langle g^b, g^c\rangle\rangle\right].\texttt{out}\left[\texttt{ddh}, \langle p, g, g^a, g^b, g^c\rangle\right])))$$

Since both the right-hand side and the left-hand side are scheduler-insensitive processes, we can use the proof rule NU2 to obtain the equivalence $\mathcal{DDHA}$. In general, this technique is useful in going from long expressions to shorter ones.

6 Conclusion and Future Directions

In this paper, we present a set of proof rules for asymptotic observational equivalence, prove them sound using a form of bisimulation, and apply the proof system to simple cryptographic protocols. We show, using only our proof rules, that the semantic security of El Gamal encryption may be derived from the Decision Diffie-Hellman (DDH) assumption, and vice versa. Although the definition of asymptotic observational equivalence is stated in essentially the same way as our first paper on this approach [18], the semantic relation is actually different here because we have refined the operational semantics. In particular, our operational semantics now allows a broader class of probabilistic schedulers (needed to choose between concurrent actions) and we execute private ("silent") actions simultaneously in parallel before public communication. These changes give us more equivalences between processes. In comparison with a recent preliminary report [22], here we have introduced a completely new probabilistic bisimulation and the congruence proof to work with the new semantics and to take advantage of some attractive ideas advanced in [28]. Other prior papers on our process calculus [21, 19] do not discuss probabilistic bisimulation or the proof rules for our calculus.

The equational proof system presented in Section 4.2 combines relatively straightforward congruence and probabilistic parallel composition rules with several rules that equate processes with different syntactic forms. Some rules that will seem completely obvious to those familiar with nondeterministic process calculus are actually the result of careful work on the operational semantics. For example, the associativity of probabilistic parallel composition failed in several semantics that initially seemed plausible, and unobservability of communication on private channels, which is essential for reasoning about idealized security protocols, motivated the current semantics in which private communications are scheduled together in parallel in advance of public communication.

We now appear to have an adequate basis to proceed in two important directions. The first is to apply equational specification and reasoning to a number of interesting examples, such as commitment and agreement protocols. The other is to develop additional proof rules as needed to carry out these examples. It is naturally expected that we will extend the reach of the proof system and simplify some of the rules in the process. Although we do not yet know when this will arise, we expect that in order to handle some examples of interest, it will be necessary to refine the form of bisimulation to support additional equational reasoning. It may also be possible to develop model-checking procedures along the lines of these already explored for probabilistic temporal logics *e.g.*, [17]. In fact, we hope to be able to develop automated reasoning procedures for use in a network security setting using techniques developed in our study of the properties of our process calculus.

Acknowledgements: Thanks to M. Abadi, R. Amadio, D. Boneh, R. Canetti, A. Datta, C. Dwork, R. van Glabbeek, M. Goldsmith, M. Hofmann, R. Jagadeesan, A. Jeffrey, S. Kannan, B. Kapron, P. Lincoln, A. Martin, R. Milner, I. Mironov, M. Mitchell, M. Naor, P. Panangaden, D. Sarenac, and P. Selinger for helpful discussions and advice on relevant literature.

References

1. Abadi, M., and Fournet, C. Mobile values, new names, and secure communication. In *28th ACM Symposium on Principles of Programming Languages* (2001), pp. 104–115.
2. Abadi, M., and Gordon, A. D. A calculus for cryptographic protocols: the spi calculus. *Information and Computation 143* (1999), 1–70. Expanded version available as SRC Research Report 149 (January 1998).
3. Atallah, M. J., Ed. *Algorithms and Theory of Computation Handbook*. CRC Press LLC, 1999, ch. 24, pp. 19–28.
4. Bellantoni, S. *Predicative Recursion and Computational Complexity*. PhD thesis, University of Toronto, 1992.
5. Boneh, D. The decision Diffie-Hellman problem. *Proceedings of the Third Algorithmic Number Theory Symposium 1423* (1998), 48–63.
6. Burrows, M., Abadi, M., and Needham, R. A logic of authentication. *Proceedings of the Royal Society, Series A 426*, 1871 (1989), 233–271. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.
7. Canetti, R. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symp. on the Foundations of Computer Science* (2001), IEEE. Full version available at `http://eprint.iacr.org/2000/067/`.
8. Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Transactions on Information Theory 22* (November 1976), 644–654.
9. Dolev, D., and Yao, A. C.-C. On the security of public-key protocols. In *Proc. 22nd Annual IEEE Symp. Foundations of Computer Science* (1981), pp. 350–357.
10. Durgin, N. A., Mitchell, J. C., and Pavlovic, D. A compositional logic for protocol correctness. In *14th IEEE Computer Security Foundations Workshop* (Cape Breton, Nova Scotia, Canada, June 2001).
11. El Gamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory 31* (1985), 469–472.

12. Goldreich, O. *The Foundations of Cryptography*, vol. 1. Cambridge University Press, June 2001.
13. Goldreich, O. *The Foundations of Cryptography*, vol. 2. June 2003. Manuscript under preparation; latest version available at `http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html`.
14. Goldwasser, S., and Bellare, M. *Lecture Notes on Cryptography.* 2003. Lecture notes for a class taught by the authors at MIT (1996–2001); available online at `http://www.cs.nyu.edu/courses/fall01/G22.3033-003/`.
15. Goldwasser, S., and Micali, S. Probabilistic encryption. *Journal of Computer and System Sciences 28*, 2 (1984), 270–299. Previous version in *STOC 1982*.
16. Hofmann, M. *Type Systems for Polynomial-Time Computation*. Habilitation Thesis, Darmstadt; see `www.dcs.ed.ac.uk/home/mxh/papers.html`, 1999.
17. Huth, M., and Kwiatkowska, M. Z. Quantitative analysis and model checking. In *LICS '97* (1997), pp. 111–122.
18. Lincoln, P. D., Mitchell, J. C., Mitchell, M., and Scedrov, A. A probabilistic poly-time framework for protocol analysis. In *Proc. 5th ACM Conference on Computer and Communications Security* (San Francisco, California, 1998), M. K. Reiter, Ed., ACM Press, pp. 112–121.
19. Mateus, P., Mitchell, J. C., and Scedrov, A. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. In *14th International Conference on Concurrency Theory* (Marseille, France, 2003), R. M. Amadio and D. Lugiez, Eds., vol. 2761 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 327–349.
20. Milner, R. *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, 1989.
21. Mitchell, J. C., Mitchell, M., and Scedrov, A. A linguistic characterization of bounded oracle computation and probabilistic polynomial time. In *Proc. 39th Annual IEEE Symposium on the Foundations of Computer Science* (Palo Alto, California, 1998), IEEE, pp. 725–733.
22. Mitchell, J. C., Ramanathan, A., Scedrov, A., and Teague, V. A probabilistic polynomial-time calculus for the analysis of cryptographic protocols (preliminary report). In *17th Annual Conference on the Mathematical Foundations of Programming Semantics, Arhus, Denmark, May, 2001* (2001), S. Brookes and M. Mislove, Eds., vol. 45, Electronic notes in Theoretical Computer Science.
23. Needham, R., and Schroeder, M. Using encryption for authentication in large networks of computers. *Communications of the ACM 21*, 12 (1978), 993–999.
24. Paulson, L. C. Mechanized proofs for a recursive authentication protocol. In *10th IEEE Computer Security Foundations Workshop* (1997), pp. 84–95.
25. Pfitzmann, B., and Waidner, M. A model for asynchronous reactive systems and its application to secure message transmission. In *IEEE Symposium on Security and Privacy* (Washington, 2001), pp. 184–200.
26. Ramanathan, A., Mitchell, J. C., Scedrov, A., and Teague, V. A probabilistic polynomial-time calculus for the analysis of cryptographic protocols. ftp://ftp.cis.upenn.edu/pub/papers/scedrov/ppc-2004-long.{ps,pdf}, 2004.
27. Schneider, S. Security properties and CSP. In *IEEE Symposium on Security and Privacy* (Oakland, California, 1996).
28. van Glabbeek, R. J., Smolka, S. A., and Steffen, B. Reactive, generative, and stratified models of probabilistic processes. *International Journal on Information and Computation 121*, 1 (August 1995).
29. Yao, A. C.-C. Theory and applications of trapdoor functions. In *IEEE Foundations of Computer Science* (1982), pp. 80–91.