Logics for reasoning about cryptographic constructions

Russell Impagliazzo* Bruce M. Kapron†

Abstract

We present two logical systems for reasoning about cryptographic constructions which are *sound* with respect to standard cryptographic definitions of security. Soundness of the first system is proved using techniques from nonstandard models of arithmetic. Soundness of the second system is proved by an interpretation into the first system. We also present examples of how these systems may be used to formally prove the correctness of some elementary cryptographic constructions.

1 Introduction

We present two formal deduction systems that can prove cryptographic constructs secure via the standard of complexity-theoretic cryptography: by giving a reduction between the security of the construct and that of the underlying primitives. Proofs in both systems avoid unnecessary quantifiers, so they may eventually be helpful in machine-assisted verification of protocol correctness. The second is simpler, but narrower in application, being usable only to reason about computational indistinguishability. We prove soundness of the more specific system by interpretting it in the general system. In fact, the main application of the general system may be as a "meta-logic" for proving soundness of logics of security.

Motivation. Correctness in security is even more important than for general systems, since attackers will actively seek out and exploit any defect. Unfortunately, it seems also to be even more difficult to design correct cryptographic protocols than to design correct programs. This is because one needs to think of the behavior of the protocols not just as they are intended to be used, but under any feasible strategy for an attacker. This leads to the very difficult problem of characterizing which attacks are possible with feasible amounts of resources. In particular, we need to specify clear assumptions about the attacker's computational limitations with respect to the underlying cryptographic functions. Finally, it is often not easy to specify precisely what security properties the protocol should have, in other words what constitutes a successful attack.

A central goal of cryptography is to construct functions and protocols solving relatively complex tasks, such as authentication or pseudo-random generation, from simpler building blocks known as cryptographic primitives. The security of the constructed object depends on both the security of the underlying primitives and the way in which these primitives are used in a construction. A flaw in either the primitive or the construction can make the constructed function vulnerable to attack.

In the absence of strong lower bounds on complexity, we cannot absolutely prove that any such constructed functions are secure. However, we would like to certify that the construction itself does not introduce weaknesses that were not present in the primitives used.

There have been several approaches to validating that a cryptographic construction does not introduce weaknesses. The standard in the complexity-theory based cryptographic community is the reduction.

The methodology of complexity-theoretic cryptography involves:

*Department of Computer Science and Engineering, University of California, San Diego, La Jolla CA 92093, russell@cs.ucsd.edu.

†Computer Science Department, University of Victoria, Victoria, BC, CANADA V8W 3P6, bmkapron@cs.uvic.ca.

1. A standard attacker model, where the attacker is limited in computational time, and wishes to achieve success with non-negligible probability.

2. A clear taxonomy of cryptographic primitives, such as one-way function, pseudo-random generator, pseudo-random block cipher, cryptographic hash function, and secure bit commitment and signature functions. Each type of primitive is clearly defined by what form of attack would be considered to break the primitive.

3. Similarly, a taxonomy of attacker models, and of security properties of protocols, specifying what kinds of interaction an attacker has with the underlying protocol and what constitutes a break of the protocol.

4. A proof of security via a reduction which modifies a successful attack on the protocol into a successful attack on the primitive.

This has become the standard in the cryptographic community for what it means to prove a protocol secure.

This method has the advantages of clearly identifying both the security properties of the construct and the assumptions made about the primitive. In addition to providing a conceptual foundation for cryptography, reductions can provide precise, quantitative tools for analyzing specific protocols ([4]).

However, the mathematical sophistication which is its strength is also a limiting factor in implementation. First, seemingly trivial modifications in the protocol can cause it to become totally insecure. Secondly, the cryptographic protocol itself may be only one component of a much more complex communications protocol that needs to be verified as a whole. Finally, the security properties guaranteed need to be transparant to system builders who wish to incorporate the new tool in their work. It is important to provide tools for specification and verification of secure protocols that are understandable to programmers and can be incorporated into existing methodologies of protocol verification. For these reasons, much research has gone into developing formal methods for the security analysis of protocols.

On the other hand, most work on the formal verification of security protocols tends to deal with security more abstractly ([8, 13, 1]). Instead of asking, "If the construct is insecure, is the primitive insecure?" these verification methods ask, "If the primitive is secure in an ideal way, is the construct secure?". Application of a primitive is assumed to produce a piece of atomic data without internal structure, which limits the power of an adversary. On the other hand, so called Dolev-Yao assumptions [13] grant adversaries more power by allowing nondeterministic choice between possible attacks. As a result of this idealized approach, it is quite possible that a protocol proved correct in the formal system is insecure when instantiated using a secure primitive. However, these logics have been quite successful in detecting flaws in protocols, and as a basis for automated tools to assist in protocol analysis ([21]). This is largely due to their having simple, intuitive rules that avoid quantifiers and explicit reasoning about probabilities.

Recently, a number of papers have proposed ways to bridge this gap between complexity-theory and logics for security. The papers [16] and [18] give a syntactic characterization of a kind of cryptographic security via equivalence in a probabilistic process calculus. Abadi and Rogaway [2] present a formal system for reasoning indistinguishability of expressions built using a pseudo-random block cipher. They show that this logic is sound in the standard sense: any algorithm for distinguishing two expressions proved indistinguishable can be converted into an attack on the block cipher. (See also [17] for a corresponding completeness result.) *Universal composability* (e.g., [10]) and related concepts (e.g., [3]) aim to produce cryptographic primitives that are "equivalent to the ideal primitive" for a variety of applications. It would be interesting to explore whether protocols proved secure by formal methods are secure when instantiated with sufficiently strong primitives or in the random oracle model.

In this paper, we take a more direct route to merge these two areas. We present two logics for security that are simplified formalizations of the language and methods of modern cryptography. Of course, all pa-

pers in complexity theory are formalizable in a sufficiently strong logical system, such as Peano arithmetic. The challenge is coming up with a formal system that is sound in the above sense, powerful enough to handle a significant fraction of constructions, and simple enough to be a useful tool for verifying that security is maintained as protocols are modified and combined in implementations. The difficulties involved in formulating such a system include: the formulation of security definitions, the use of probabilities in definitions of security, reasoning about random choices and distributional problems, quantifying the computational power of adversaries, and intuitive but dangerous mis-applications of induction. An illustrative example for this last difficulty is given in Section 2.

It is the aim of this paper to show that it is possible to formulate logical systems that handle these issues implicitly. We combine ideas from cryptography, implicit complexity, and proof complexity. By having a term algebra (such as in Cook's PV [12]) represent feasible functions, quantification over functions implicitly defines the scope of feasible adversaries. A counting quantifier is added to allow simple probabistic reasoning. Only "small" numbers (polynomial length) can be proved to exist, so one can implicitly define negligible vs. non- negligible probabilities and limit the scope of induction arguments. By eliminating asymptotics and explicit resource bounds, we make proofs largely quantifier-free, consisting of equations and inequalities. The logic is sound, in the standard sense: any security proof for a construction of one cryptographic object from another provides a reduction from an algorithm that breaks the constructed object to one that breaks the underlying primitive.

Soundness is proved using non-standard models of arithmetic, but knowledge of such models is unnecessary for users – all that is required is an intuition regarding "relatively large" values. A similar situation occurs in the setting of nonstandard analysis. It is possible to give an elementary undergraduate presentation of calculus using infinitesmals (see, e.g., [15]). Proving soundness of the axiomatization of infinitesmals which is used requires the introduction of nonstandard models. However, it is not essential to understand the soundness proofs in order to use these systems.

The first logic we introduce is rather general and powerful, but still a little cumbersome. We show how it can be specialized to provide simpler logics for limited types of reasoning. In particular, we give a simple, sound logic for reasoning about computational indistinguishability. The soundness proof for this system follows directly from that of our more general system, by interpretting the specific system in the general one. Indeed, we think the main application of the general system may be as a "meta-logic", to prove the soundness of more specialized logics for security.

Our work is a preliminary, step towards provably sound, practical formal methods for security analysis. We believe that the primary purpose of formal methods in cryptography is not to make it easier for cryptographers to produce valid proofs, but to clarify the difference between valid and invalid proofs for non-cryptographers. By restricting the kind of proofs allowed, we eliminate many forms of common mistake that, to the non-expert, appear indistinguishable from a valid argument. On the other hand, we feel our logic is both rich enough that it is possible, and simple enough that it is not overly burdensome, to formalize most proofs in our logic. These proofs could be used by implementers to distinguish harmless and harmful modifications and combinations of protocols.

In this paper, we focus primarily on constructions of cryptographic functions, rather than protocols. However, we give one example (in Appendix C) which demonstrates how reasoning about simple protocols is supported in our framework.

Notation. Let $\{0,1\}^*$ denote the set of all binary strings, and $\{0,1\}^n$ strings of length n. For $a \in \{0,1\}^n$ and $b \in \{0,1\}^m$, $a \circ b \in \{0,1\}^{m+n}$ is the concatenation of a and b. For $a, b \in \{0,1\}^n$, $a \odot b$ denotes mod 2 inner product of a and b (i.e., when a and b are viewed as bit vectors.) For $a \in \{0,1\}^n$, and $1 \leq i < j \leq n$, a_i denotes the ith bit of a and $a_{\{i\ldots j\}}$ denotes $a_i \circ \cdots \circ a_j$. If A is any set, $\#A$ will denote the number of elements in A.

2 A fallacious induction argument

We begin with a fallacious induction proof to demonstrate the potential pitfalls of induction applied to cryptographic constructions. By an injudicious application of induction we will prove the correctness of a construction of a pseudo-random generator based on the iteration of Goldreich-Levin style construction of a hard-core predicate. However, we will also show that the function which is the result of the construction is not one-way, yielding a contradiction.

A function f is *one-way* if it is easy to compute f but any computationally bounded adversary has no more than a negligible chance of finding a preimage for $f(X)$ when X is chosen uniformly at random from $\{0,1\}^n$, for sufficiently large n. A 0-1 valued function b is a hard-core predicate for a function f if it is easy to compute b, but any computationally limited adversary has a no better than negligible chance of doing better than guessing $b(X)$ when given $f(X)$, if X is chosen uniformly at random from $\{0,1\}^n$ for sufficiently large n. A function $f : \{0,1\}^n \to \{0,1\}^m$ is a *pseudo-random generator* if any computationally limited adversary has no more than a negligible chance of distinguishing $f(X)$ from Y chosen uniformly at random from $\{0,1\}^m$, whenever X is chosen uniformly at random from $\{0,1\}^n$. The *stretch of* f is $m-n$.

The *Goldreich-Levin theorem* [14] states that if f is any one-way function, then there is a one-way function g defined from f and a 0-1 valued function b such that b is a hard-core predicate for g. In particular, $g(x \circ r) = f(x) \circ r$ and $b(x \circ r) = x \odot r$. Note that we are assuming x and r have the same length. An immediate application of the Goldreich-Levin theorem is a method for constructing a pseudo-random generator with stretch 1 from a one-way permutation. If f is a permutation then so is g, so that $g(x \circ r)$ will be uniformly distributed and b will be a hard-core predicate for g. Let $f'(x \circ r) = g(x \circ r) \circ b(x \circ r)$, so that f' is pseudo-random (with stretch 1). This f' is also one-way, and so we can repeat this process to obtain another hard-core predicate.

Can we iterate this construction to obtain a pseudo-random generator with stretch n? We will now give a fallacious argument that this is the case. Suppose that $f : \{0,1\}^n \to \{0,1\}^n$ is a one-way permutation. We define a sequence of functions $f_0, \ldots, f_n$, where $f_i : \{0,1\}^{n+n^2} \to \{0,1\}^{n+n^2+i}$ as follows: $f_0(x \circ \vec{r}) = f(x) \circ \vec{r}$, and for $0 < i \leq n$, $f_i(x \circ \vec{r}) = f_{i-1}(x) \circ (x \odot r_i)$, where $\vec{r}$ denotes $r_1 \circ \cdots \circ r_n$. Now we have that for each i, $0 < i \leq n$, if f_{i-1} is pseudo-random, then so is f_i, by an argument similar to that given above, with a slightly modified version of the Goldreich-Levin theorem. But if f is a one-way permutation then f_0 is pseudo-random. Thus we can fallaciously conclude by "induction" that f_n is pseudo-random.

Is f_n really pseudo-random? If it were, it would also be one-way. Now $f(x \circ \vec{r})$ has the form $f(x) \circ \vec{r} \circ \vec{b}$, where $\vec{b} = b_1 \circ \cdots \circ b_n$ and $b_i = x \odot r_i$. This gives rise to the following system of equations over GF(2):

$$\sum_{j=1}^{n} x^j r_i^j = b_i \qquad (1 \leq i \leq n)$$

So we can efficiently recover x from $f(x \circ \vec{r})$ for any x and $\vec{r}$ by solving this system of equations, contrary to the purported one-wayness of f_n.

Where did our "induction" argument go wrong? One way of looking at the problem is that it hides the fact that at stage i, an adversary's advantage, or chance of success, depends on i. This advantage could double at each stage, and still appear to be negligible given the assumption that the advantage at the previous stage is negligible. However, after n iterations the advantage will be increased by a factor of 2^n and will no longer be negligible. This is analogous to the familiar situation in which composing a feasible function with itself a feasible number of times results in a function which is not feasible (e.g. exponentiation by repeated squaring.) Clearly, any system for reasoning formally about such constructions will require a careful treatment of induction.

3 A general system for cryptographic constructions

We begin by giving a high-level presentation of what types of reasoning can be done in the general formal system. In the following, we use "polynomial" as shorthand for "relatively short" or "relatively fast". Our system will need to deal with the following types of objects.

1. There is a *security parameter* presented as two constants: $\mathbf{n}$, a "moderate integer", is the security parameter in binary; and $\mathbf{s}$, a "string", is the same security parameter in unary, $\mathbf{n} = |\mathbf{s}|$. This security parameter is assumed to be sufficiently large that any particular asymptotically true statement is true of $\mathbf{n}$.

2. *Strings* are objects that could be inputs, outputs, or random tapes for a probabilistic algorithm that runs in time polynomial in the security parameter. $\mathbf{s}$ is a string, the variables $x, y, z...$ represent randomly chosen strings, and any efficient function applied to a string yields a string. Using a standard embedding such as dyadic notation, we can also view strings as integers. Thus the usual arithmetic operations (except exponentiation) and comparison relations make sense for strings. Strings are implicitly assumed to be of polynomial length; and random variables must be defined to be of some "moderate integer" length when quantified over.

3. *Moderate integers* are those polynomially bounded in the security parameter. An integer is moderate if and only if it is the length of some string, so we don't need to have new variables of this type. We use $p(\mathbf{n})$ for p a polynomial or $|t|$ for t a string term to denote a moderate integer. Now, for example, we can express that q is negligible function by: for every z and every x with $|x| = \mathbf{n}$, $q(x) \leq \frac{1}{|z|}$, eliminating the quantifiers needed for the usual asymptotic defnition.

4. *Feasible functions* are polynomial-time functions that take strings to strings. They include certain basic functions, such as a pairing function and arithmetic operations. New feasible functions can be defined from others by composition or certain forms of recursion. It is well-known (see [11] that in this way we are able to define any particular polynomial-time function. There are also function symbols of each arity, $f, g, ...$, which allow us to quantify over functions. However, quantification over strings is not allowed outside a quantification over functions, so functions cannot depend on a specific input. We use $A, A_1, A_2...$ to represent possible adversaries for cryptographic functions, but formally, these are just the same as other function variables.

5. *Counting integers* are integers that represent sizes of sets of strings. If $\varphi(x_1, ..x_k)$ is a quantifier-free formula in string variables $x_1, ...x_k$ (possibly involving function variables) and $t_1, ..t_k$ are moderate integer terms, we can form the counting integer term $\#(|x_1| = t_1 \wedge ... \wedge |x_k| = t_k)\varphi(x_1, ...x_k)$ to represent the number of strings that satsify φ of the given sizes. Arithmetic operations make sense for counting integers, and their lengths are moderate integers. However, not being readily computable, they are not allowed to be inputs to function symbols or treated as strings. Thus, we can reason about $C(\mathbf{n}) = \#(|x| = |\mathbf{n}|)(0 \leq x \leq n - 1 \wedge gcd(x, \mathbf{n}) = 1)$, but we cannot use $C(\mathbf{n})$ as the input to an adversary (unless we can show how to compute it another way).

6. We can use counting integers implicitly to reason about *probabilities*, rational numbers in $[0, 1]$. $\Pr_{|x|=t}\varphi(x) = \frac{\#(|x|=t)\varphi(x)}{\#(|x|=t)(x=x)}$. Since both numerator and denominator are counting integers, and we can simulate arithmetic operations on rationals by operations on their numerators and denominators, we can also reason about probabilities.

Accepting the existence of a fixed parameter $\mathbf{n}$ that satisfies all asymptotically true arithmetic statements, and a distinction between integers that are "polynomially bounded" in $\mathbf{n}$ and those that are not, requires

some suspension of disbelief. Our feeling is that this will mainly bother those that are mathematically but not logically sophisticated. The mathematically unsophisticated will accept it unquestioningly, and the logically sophisticated will realize that any particular proof is only using a fixed set of asymptotic facts and polynomial bounds, and so is sound for sufficiently large n. Those in the middle range might take some comfort from the fact that we have proved soundness for our system, and be willing to suspend their disbelief based on this proof.

The language: We will work in a language which extends that commonly used in bounded arithmetic (see, e.g., [9]). We will have a collection of first-order variables $x, y, z, \ldots, i, j, k \ldots$. Note that first-order terms will be interpreted variously as strings over $\{0,1\}$ and as numbers. We will therefore assume that numbers are represented in dyadic notation, which gives a 1-1 correspondence between numbers and strings over $\{0,1\}$ (i.e. $0 \equiv \epsilon, 1 \equiv 0, 2 \equiv 1, 3 \equiv 00\ldots$). We will try to follow the following convention: when the intended interpretation of a variable is a string, we will use $x, y, z, \ldots$, and when the intended interpretation is a number we will use $i, j, k, \ldots$. We will also have for each $k > 0$, *function variables of arity* k $f^k, g^k, h^k, \ldots, f^k_1, f^k_2, \ldots$. These variables are intended to represent poly-time functions. Note that while we have function variables, we are really working in a multi-sorted first-order theory. If f^k is a function variable and $t_1, \ldots, t_k$ are terms, then $f^k(t_1, \ldots, t_k)$ is a term. Note that we will typically omit superscripts indicating arity for the sake of readability.

As usual, we have relations symbols $=, \leq$ and a collection of basic function symbols and axioms which capture the intended interpretation of these symbols and the relationships between them. We will remain somewhat informal about this, but we do assume the following function symbols with given interpretations (to be concrete – albeit extravagant – we could take as axioms all true open formulas involving basic function symbols.) $0, S, +, \cdot$ are standard arithmetic constants and operations ($S(x) = x + 1$); $\circ$ denotes string concatenation; $|x|$ denotes the number of bits in the *dyadic* notation of x; $x_{\{i\ldots j\}}$ denotes bits i through j of x ($1 \leq i < j \leq |x|$), $x_{\{i\}}$ denotes bit i of x; $x \otimes y$ denotes $2^{|x|\cdot|y|}$ (this *smash* function is often denoted $\#$ but we need this symbol for other purposes.)

We will also have constant symbols $\mathbf{s}$ and $\mathbf{n}$ as discussed above.

Finally, we will have a *counting "quantifier"* $\#$. For any quantifier-free formula φ, and sequences $\vec{x}, \vec{y}$ of variables, $\#(|x_1| = |y_1| \wedge \ldots |x_k| = |y_k|)\varphi$ will be a term. For readability, we sometimes write this as $\#(|\vec{x}| = |\vec{y}|)\varphi$ or

$$\#\begin{pmatrix} |x_1|=|y_1| \\ \cdots \\ |x_k|=|y_k| \end{pmatrix}\varphi$$

The intended interpretation of such a term is

$$\#\left\{\langle x_1, \ldots, x_k\rangle \mid |x_i| = |y_i|, 1 \leq i \leq k \text{ and } \varphi(x_1, \ldots, x_k)\right\}.$$

Note that the occurrences of $\vec{x}$ in $\#(|\vec{x}| = |\vec{t}|)\varphi$ are bound.

Note that by introducing $\#$, we have the possibility of terms which represent functions of (potentially) more than poly-time complexity. In order to limit this as much as possible, we will identify a subclass of terms, the *basic terms*, as being those which do not involve the use of $\#$. Thus a basic term is built up from the basic function symbols and function variables. We will impose the following restriction on terms: we allow terms of the form $f(t)$ where f is a function variable only if t is a basic term. On the other hand, we do allow terms to be formed from non-basic terms using the basic function symbols.

We introduce some abbreviations related to $\#$. First of all, if C is a counting term of the form $\#(|\vec{x}| = |\vec{y}|)\varphi$, then by $\#(|x| = |y|)C$ we denote the term $\#(|x| = |y| \wedge |\vec{x}| = |\vec{y}|)\varphi$. We will also use terms of the form $\#(0 \leq i \leq j)\varphi$ which is used to count the number of *values* $i \in \{0, \ldots, j\}$ for which φ holds. This will an abbreviation for $\#(|t| = |2j|)\varphi[b(t)/j]$. Here b is (poly-time) function such that $b(t)$ returns

the value corresponding to the (possible zero-padded) *binary* string given by t. Note that for a formula φ with free variable i and a term t, $\varphi[t/i]$ denotes φ with all occurrences of i replaced by t. The intended interpretation of $\#(0 \leq i \leq j)\varphi$ is the number of values of i between 0 and j which satisfy φ.

Formulas are defined in the standard way, with the extra clause: if φ is a formula and f is a function variable which has a free occurrence in φ, then $\forall f \varphi$ and $\exists f \varphi$ are formulas.

Axiom system: We are now ready to informally describe the system T. As mentioned above, we will have a collection of axioms which capture the intended interpretation of the basic function symbols in the language. We will not spell out these axioms here, but we assume that there is a collection of such axioms, which we will call *BASIC*. Beyond *BASIC*, we first have axioms which state that there is a string of "moderate integer" length, namely $|\mathbf{s}| > \overline{k}$ for every $k \in \mathbb{N}$, where $\overline{k} = S(S(\dots S(0)\dots)$ (k times), and an axiom $\mathbf{n} = |\mathbf{s}|$. Also, for each basic function symbol in our language, we will include an axiom stating that there is a poly-time function defined by that function symbol, e.g., $\exists f \forall x \forall y (f(x,y) = x \circ y)$. We will also need axioms which state that the poly-time functions are closed under certain closure conditions. For examples, we will have an axiom of the form $\forall g \forall h \exists f \forall x (f(x) = g(x, h(x)))$. We will also have an axiom which states that poly-time functions are closed under limited recursion on notation, that is, if we iterate a poly-time function a polynomial number of times, the result will also be a poly-time function, provided that we can show that the result has polynomially-bounded growth.

We will require a collection of axioms which give the properties of $\#$. These will include axioms which describe how $\#$ interacts with logical connectives, for example, an inclusion-exclusion principle with respect to $\wedge$ and $\vee$:

$$\#(|x| = |y|)(\varphi \vee \psi) + \#(|x| = |y|)(\varphi \wedge \psi) = \#(|x| = |y|)\varphi + \#(|x| = |y|)\psi$$

We will also have axioms which describe how $\#$ interacts with basic function symbols, e.g.,

$$\#(|x| = |y| \wedge |x'| = |y'|)\varphi(x, x') = \#(|x| = |y \circ y'|)\varphi(x_{\{1\dots|y|\}}, x_{\{|y|+1\dots|y\circ y'|\}})$$

A more detailed presentation of axioms for counting terms and poly time functions is given in Appendix A

Finally, we require an induction scheme. We will limit induction to open (i.e. quantifier-free) formulas. Induction will be on the *length* of the dyadic representation of the induction variable. For each polynomial p and open formula φ, we will have the following induction axiom:

$$(\varphi(0) \wedge (\forall x < p(|y|)(\varphi(x) \rightarrow \varphi(x+1)))) \rightarrow \forall x \leq p(|y|)\varphi(x) \qquad (\mathrm{LIND}^o(p))$$

By a *bounded* formula we mean a formula in which quantifiers occur only in the form $\forall x(|x| \leq |y| \rightarrow \varphi)$ or $\exists x(|x| \leq |y| \wedge \varphi)$. The following is the fundamental result about T:

Theorem 3.1 (Soundness Theorem) *Suppose that φ and ψ are bounded formulas such that*

$$T, \forall g \forall \vec{z} \varphi(\vec{f}, g, \vec{z}, \mathbf{s}) \vdash \forall g \forall \vec{z} \psi(\vec{f}', g, \vec{z}, \mathbf{s}) \qquad (*)$$

where $\vec{f}$ and $\vec{f}'$ are (possibly non-disjoint) sequences of function variables. We then have the following for all sequences $\vec{f}, \vec{f}'$ of poly-time functions. If (in $\mathbb{N}$), for every poly-time function g and polynomials $\vec{p}$, there is an n_0 for that for all $x, \vec{z}$ with $|x| \geq n_0$, and $|z_i| = p_i(|x|)$, $\varphi(\vec{f}, g, \vec{z}, x)$, then (in $\mathbb{N}$), for every poly-time function g and polynomials $\vec{p}$, there is an n_0 so that for all $x, \vec{z}$ with $|x| \geq n_0$, and $|z_i| = p_i(|x|)$, $\psi(\vec{f}', g, \vec{z}, x)$.

Proof: A proof of the Soundness Theorem is given in Appendix B.

Significance of the Soundness Theorem. We want to use the soundness theorem to prove the correctness of a construction based on some cryptographic primitive. Suppose that f is a function symbol representing this primitive, and that f' is defined using f. The formula φ will have f and f' occurring as free variables, and will formalize f's security as well as giving the definition of f' from f (i.e. φ will include some set of equations which give the definition of f' from f). The security assumption on f will state that any poly-bounded adversary has no more than a negligible chance of breaking f. Thus we have a formula of the form $\forall g \ldots$, where g represents an arbitrary poly-time adversary (in the sequel, we will typically denote such an adversary function by A). The formula ψ will have f' as a free variable and will formalize the security of f', again by using a formula of the form $\forall g \ldots$. The asymptotic assumption about φ is then equivalent to a standard complexity-based assumption about the security of f. By (*), we can formally derive that f' is secure, under the formal assumption that f is secure. The Soundness Theorem, which gives us an asymptotic conclusion about ψ, allows us to conclude that f' is cryptographically secure. In the next section we will give an example which spells out this approach in more detail.

Example: verifying a pseudo-random stretcher. We will now use the soundness theorem to prove the correctness of a construction for "stretching" the output of a pseudo-random generator. The reader should note that the primary purpose of this example is to demonstrate the nuts-and-bolts of proofs in the system T. It is not our contention that such proofs are particularly perspicuous, or amenable to automation. In fact, our ultimate goal is to develop systems which allow more direct reasoning about cryptographic constructions (such a system is presented in Section 4). In this setting, the system T can be viewed as an intermediate step. The techniques presented here can be extended to give general soundness results for the system of Section 4.

To begin, we must show how the definition of PRG can be formalized in our framework. Recall that for any positive polynomial p, a poly-time function $g : \{0,1\}^* \to \{0,1\}^*$ is a *pseudo-random generator with stretch* $p(n)$ if for all x, $|g(x)| = |x| + p(|x|)$ and if for every poly-time function $A : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, and all polynomials r and q, there is an n_0 such that for all $n \geq n_0$,

$$\Pr_{\substack{R \in_{\mathcal{U}} \{0,1\}^{r(n)} \\ X \in_{\mathcal{U}} \{0,1\}^{n}}} [A(R, g(X)) = 1] - \Pr_{\substack{R \in_{\mathcal{U}} \{0,1\}^{r(n)} \\ Y \in_{\mathcal{U}} \{0,1\}^{n+p(n)}}} [A(R, Y) = 1] \leq \frac{1}{q(n)} \qquad (\dagger)$$

Statements having the form of (†) can be formalized in the language of T as follows:

$$\forall z_1 z_2 \left(|z_1| \cdot \left(2^{p(\mathbf{n})} \cdot \# \left({}^{|r|=|z_2|}_{|x|=\mathbf{n}} \right) (A(r, g(x)) = 1) - \# \left({}^{|r|=|z_2|}_{|y|=\mathbf{n}+p(\mathbf{n})} \right) (A(r, y) = 1) \right) \leq 2^{|z_2|} 2^{\mathbf{n}+p(\mathbf{n})} \right).$$

In the sequel, we will sometimes write formulas using standard probability notation for the sake of readability (as in (†)). Such formulas may be formalized in the language of T, as described here. At this point, as we are focussing on the soundness of T, we do not want to introduce a more complex syntax involving symbols such as Pr, as this would only complicate the soundness proof. When applying the system, of course, we want a richer syntax, which can easily be built up on top of T. We will use this richer syntax in our examples of proofs in T.

A few more words on methodology. In this setting, we start by *assuming* the existence of some *primitive* f, which is a poly-time function displaying some security property. The security property is given in the assumption $\forall A \forall r \forall z \varphi$. The fact that f is poly-time allows us to introduce a function symbol representing f. Based on this primitive, we construct a *scheme* which, under the assumptions about f, is a poly-time function satisfying some security property. The constructed object f' is defined in terms of f and the recursion schemes in the system, such as limited recursion. These schemes all preserve polynomial-time. In reasoning about f', we will assume that T is extended with defining equations for f'. In general, in order to

reduce the level of formalization, we will introduce defining equations for any function that we assume as a poly-time function. It is not hard to see how to completely formalize this approach within T.

Suppose that f is a PRG with stretch 1. We will write $f(x)$ as $b(x) \circ r(x)$ where $|b(x)| = 1$ and $|r(x)| = |x|$. Define f', by $f'(x,i) = b'(x,i) \circ r'(x,i)$ where

$$\begin{aligned} r'(x,0) &= x & b'(x,0) &= 0 \\ r'(x,i+1) &= r(r'(x,i)) & b'(x,i+1) &= b'(x,i) \circ b(r'(x,i)) \\ |r'(x,i)| &\leq |x| & |b'(x,i)| &\leq i \end{aligned}$$

Note that this is the construction of Goldreich and Micali presented in [7]. The bounds on the growth of r' and b' are not required for the correctness of the construction, but to ensure that they are feasible functions.

We will now show how, based on our soundness theorem, we can use f' to obtain a PRG with stretch n from a PRG f with stretch 1. (We could in fact obtain a PRG with stretch $p(n)$ for any polynomial p. We are limiting ourselves to stretch n just to simplify the presentation.)

Theorem 3.2 *Let $\varphi(f, A, r, z)$ and $\psi(f', A, r, z)$ be the following formulas*

$$\Pr_{\substack{R \in_{\mathcal{U}} \{0,1\}^{|r|} \\ X \in_{\mathcal{U}} \{0,1\}^{\mathbf{n}}}}[A(R, f(X)) = 1)] - \Pr_{\substack{R \in_{\mathcal{U}} \{0,1\}^{|r|} \\ Y \in_{\mathcal{U}} \{0,1\}^{\mathbf{n}+1}}}[A(R,Y) = 1] \leq \frac{1}{|z|}$$

$$\Pr_{\substack{R \in_{\mathcal{U}} \{0,1\}^{|r|} \\ X \in_{\mathcal{U}} \{0,1\}^{\mathbf{n}}}}[A(R, f'(X, \mathbf{n})) = 1] - \Pr_{\substack{R \in_{\mathcal{U}} \{0,1\}^{|r|} \\ Y \in_{\mathcal{U}} \{0,1\}^{\mathbf{2n}}}}[A(R,Y) = 1] \leq \frac{1}{|z|}$$

Then $T', \forall A \forall r \forall z \varphi \vdash \forall A \forall r \forall z \psi$.

Here T' denotes T extended by the definition of f' given above.

From this result and the Soundness theorem for T, it follows immediately that

Corollary 3.3 *If f is a PRG with stretch 1, then $\lambda x. f'(x, |x|)$ is a PRG with stretch n.*

The proof of theorem 3.2 will be based on a standard "hybrid argument". We require the following two lemmas which may be proved using the $\mathrm{LIND}^o(p)$ induction axiom.

The first lemma will be used in order to argue that if the average "distance" between two adjacent hybrid distributions is "large", then the distance between two adjacent hybrid distributions chosen at random will also be large.

Lemma 3.4 (Telescoping Sum Lemma) *For any two counting terms C, C', assuming that $\forall m \leq p(\mathbf{n}) \forall 0 \leq i < m (a \cdot C = b \cdot C'[i+1/i])$, we can prove in T that*

$$\forall m < p(\mathbf{n}) (a \cdot C[m/i] - b \cdot C'[0/i] = a \cdot \#(0 \leq i < m+1) C - b \cdot \#(0 \leq i < m+1) C')$$

The second lemma is more technical in nature; it basically allows us to view one step in the recursive construction of f' as a "context" in which the primitive f is applied to an arbitrary input and then operated upon. In a hybrid argument, this allows us to go from an adversary which breaks the construction to one which breaks the the primitive.

Lemma 3.5 (Context Lemma) $T \vdash \forall x \forall i < \mathbf{n} f'(x, i+1) = b(x) \circ f'(r(x), i)$

Proof of Theorem 3.2: In what follows, if A is a 0-1 valued function, we will write $A(\vec{x})$ and an abbreviation for the formula $A(\vec{x}) = 1$. Suppose that $\forall A \forall r \forall z \varphi$ holds, but that $\forall A \forall r \forall z \psi$ fails, as witnessed by A_0, r_0 and z_0. Let δ denote the term $2^{\mathbf{n}} \cdot \#(|r| = |r_0| \wedge |x| = \mathbf{n}) A_0(r, f'(x, \mathbf{n})) - \#(|r| = |r_0| \wedge |x| = 2\mathbf{n}) A_0(r, x)$. By assumption $|z_0| \cdot \delta > 2^{|r_0|} 2^{2\mathbf{n}}$. Define the poly-time hybrid function h as follows:

$$h(y, i, u) = \begin{cases} y_{\{1 \ldots \mathbf{n} - i\}} \circ u_1 \circ f'(u_{\{2 \ldots \mathbf{n}+1\}}, i) & \text{if } i \leq n; \\ 0 & \text{otherwise} \end{cases}$$

and let A' be a poly-time function defined by $A'(r, y, i, u) = A_0(r, h(y, i, u))$ We can now use the Context Lemma to obtain the following two equalities (**):

$$2^{\mathbf{n}} \cdot \#(|r| = |r_0| \wedge |x| = \mathbf{n}) A_0(r, f'(x, \mathbf{n})) = \#(|r| = |r_0| \wedge |x| = \mathbf{n} \wedge |y| = \mathbf{n}) A'(r, y, \mathbf{n} - 1, f(x))$$
$$2 \cdot \#(|r| = |r_0| \wedge |x| = 2\mathbf{n}) A_0(r, x) = \#(|r| = |r_0| \wedge |x| = \mathbf{n} + 1 \wedge |y| = \mathbf{n}) A'(r, y, 0, x)$$

This allows us to express δ in terms of the distance between the two extreme hybrids. In particular, if we let δ' denote

$$2 \cdot \#(|r| = |r_0| \wedge |x| = n \wedge |y| = n) A'(r, y, \mathbf{n} - 1, f(x)) - \#(|r| = |r_0| \wedge |x| = \mathbf{n} + 1 \wedge |y| = \mathbf{n}) A'(r, y, 0, x),$$

then from the equalities (**) we obtain $\delta' = 2\delta$, so that $|z_0| \cdot \delta' > 2^{|r_0|} 2^{2\mathbf{n}+1}$ (†). Now by the Context Lemma we have, for $0 \leq i < \mathbf{n} - 1$, that

$$2 \cdot \#(|r| = |r_0| \wedge |x| = \mathbf{n} \wedge |y| = \mathbf{n}) A'(r, y, i, f(x)) = \#(|r| = |r_0| \wedge |x| = \mathbf{n} + 1 \wedge |y| = \mathbf{n}) A'(r, y, i + 1, x).$$

And so by the Telescoping Sum Lemma we have

$$\delta' = 2 \cdot \#(0 \leq i < \mathbf{n} \wedge |r| = |r_0| \wedge |x| = \mathbf{n} \wedge |y| = \mathbf{n}) A'(r, y, i, f(x)) - \\ \#(0 \leq i < n \wedge |r| = |r_0| \wedge |x| = \mathbf{n} + 1 \wedge |y| = \mathbf{n}) A'(r, y, i, x). \quad (\dagger\dagger)$$

We are now in a position to give A_1, r_1 and z_1 which witness the failure of the assumption $\forall A \forall r \forall z \varphi$. Define the poly-time function A_1 as follows:

$$A_1(s, u) = \\ \begin{cases} A'(r, y, b(t), u) & \text{if } s = r \circ y \circ t \text{ with } |r| = |r_0|, |y| = \mathbf{n} \text{ and } |t| = |2\mathbf{n}|, \text{ and } 0 \leq b(t) < \mathbf{n}; \\ 0 & \text{otherwise.} \end{cases}$$

Recall that b is a poly-time function such that $b(t)$ is the value whose *binary* notation is t. Choosing r_1 such that $|r_1| = |r_0| + \mathbf{n} + |2\mathbf{n}|$, it follows by (††) that $\delta' = 2 \cdot \#\left(\begin{smallmatrix} |r| = |r_1| \\ |x| = \mathbf{n} \end{smallmatrix}\right) A_1(r, f(x)) - \#\left(\begin{smallmatrix} |r| = |r_1| \\ |x| = \mathbf{n}+1 \end{smallmatrix}\right) A_1(r, x)$. Finally, choosing z_1 such that $|z_1| = |z| \cdot 2^{|2\mathbf{n}|}$, it follows by (†) that $|z_1| \cdot \delta' > 2^{|r_1|} 2^{\mathbf{n}+1}$. But then A_1, r_1, z_1 provide a contradiction to the assumption $\forall A \forall r \forall z \varphi$. ■

The framework that we have introduced so far is not general enough to model arbitrary notions of security for protocols. For example, to deal with adversaries with some form of chosen-message attack would require the introduction of functionals. However, we do have sufficient machinery to model protocols in which only one execution of the protocol is analyzed, or in which the adversary is non-adaptive. As an example, we show how to model and verify a protocol for flipping coins on a public line [5], which uses hidden bit commitment as a subroutine. In order to achieve hidden bit commitment, we use a construction of Naor [19], which uses a pseudo-random generator as a primitive. Details are given in Appendix C

4 A system for computational indistinguishability

A drawback of the system presented in the preceding section is that it still requires us to reason explicitly about probabilities. In this section we show that by restricting our attention to a particular security property, we are able to eliminate the need for such explicit reasoning. We will present a system for reasoning about *computational indistinguishability*, in which proofs are essentially equational. We will prove soundness for this system by interpreting it in the general system from the previous section and then appealing to the soundness proof that we have already given for this system. In fact, we see this kind of soundness proof as the most important application of the general system: it provides a formal framework for doing "meta-proofs" for systems providing more direct forms of reasoning. An outline of how the interpretation proceeds is given in Appendix D.

We will work in a language with terms which denote probabilistic poly-time (PPT) functions. More precisely, terms will denote PPT function *ensembles*. To this end we will have a collection of basic functions, and typical closure schemes. We will also include terms of the form $\mathsf{let}\ i \leftarrow \mathsf{rand}(p(\mathbf{n}))\ \mathsf{in}\ t$ and $\mathsf{let}\ x \leftarrow \mathsf{rs}(p(\mathbf{n}))\ \mathsf{in}\ t$, for any polynomial p. Intuitively, the meaning of the first kind of term is just the meaning of t where variable i is selected uniformly at random from the set $\{0, \ldots, p(\mathbf{n}) - 1\}$, while the meaning of the second kind of term is t where x is selected uniformly at random from strings of length $p(\mathbf{n})$. So rand is shorthand for "random index" while rs is shorthand for "random string". Note that we will only allow the formation of terms of the first (second) form when i is free in t (resp. x is free in t.) In cases where there is no ambiguity, we sometimes write, e.g., $t[\mathsf{rs}(p(\mathbf{n}))/x]$ instead of $\mathsf{let}\ x \leftarrow \mathsf{rs}(p(\mathbf{n}))\ \mathsf{in}\ t$, say when t has only one occurrence of x. Here $t[s/x]$ means term t will all *free* occurrences of x replaced by s. Free variables are those which are not in the scope of a `let`. We always assume that substitution is done in such a way as to avoid variable capture, by changing the name of bound variables if necessary. In general, we refer to $i \leftarrow \mathsf{rand}(\mathbf{n})$ and $x \leftarrow \mathsf{rs}(\mathbf{n})$ as *random bindings*. Suppose $b_1, \ldots, b_k$ is a sequence of bindings. We will sometimes abbreviate $\mathsf{let}\ b_1\ \mathsf{in}\ \ldots\ \mathsf{let}\ b_k\ \mathsf{in}\ t$ as $\mathsf{let}\ \begin{pmatrix} b_1 \\ \cdots \\ b_k \end{pmatrix}\ \mathsf{in}\ t$ Note that with this syntax, the ordering of bindings from top to bottom is significant because for $1 \leq i < j \leq k$, b_j may refer to the variable bound in b_i. We also note that due to the semantics of terms, we can assume without loss of generality that terms always have the form $\mathsf{let}\ b_1\ \mathsf{in}\ \ldots\ \mathsf{let}\ b_k\ \mathsf{in}\ t$ where t is just a term of T. This follows from the fact that our semantics treats as identical the terms $(\mathsf{let}\ b_1\ \mathsf{in}\ \ldots\ \mathsf{let}\ b_k\ \mathsf{in}\ t)[(\mathsf{let}\ b_{k+1}\ \mathsf{in}\ s)/y]$ and $\mathsf{let}\ b_1\ \mathsf{in}\ \ldots\ \mathsf{let}\ b_{k+1}\ \mathsf{in}\ t[s/y]$. A term in this form is *closed* just in case all the variables occurring in t are bound in one of the $b_1, \ldots, b_k$. Suppose that s and t are closed terms. We introduce formulas of the form $s \approx t$, whose intended meaning is that the distribution ensembles represented by s and t are computationally indistinguishable. Again, this can be formalized by an appropriate interpretation in T.

We would like to reason directly about $\approx$. To this end, we will introduce a number of rule schemas which capture intuitive properties of $\approx$. The soundness of these rules may be showing that their interpretations in T may be proved in T. Soundness then follows from the soundness of T with respect to cryptographic interpretations. Each of the rules given below have fairly simple proofs in T. Details are given in Appendix D.

The first rule relates universal equality provable in T to $\approx$. Suppose that s and t are terms of T, and that $Q_1, \ldots, Q_k$ are a sequence of quantifiers of the form $\forall |x| \leq p(\mathbf{n})$ or $\forall i < p(\mathbf{n})$. For each quantifier Q_i define a corresponding random binding of the form $x \leftarrow \mathsf{rs}(p(\mathbf{n}))$ (resp. $i \leftarrow \mathsf{rand}(p(\mathbf{n}))$). Finally, assume that all the free variables of s and t appear in some Q_i. We then have

$$\frac{T \vdash Q_1 Q_2 \ldots Q_k (s = t)}{\mathsf{let}\ b_1\ \mathsf{in}\ \ldots\ \mathsf{let}\ b_k\ \mathsf{in}\ s \approx \mathsf{let}\ b_1\ \mathsf{in}\ \ldots\ \mathsf{let}\ b_k\ \mathsf{in}\ t} \qquad \text{(UNIV)}$$

Note that the choice of T in the antecedent of this rule is somewhat arbitrary and was made here for the sake of concreteness. We could actually choose a system without much of the extra machinery introduced in T,

say a suitably formulated version of Cook's PV [12].

The next rule states that we may substitute indistinguishable terms into any probabilistic polynomial time context. Let v be any term in the extended language whose only free variable is x. We then have

$$\frac{s \approx t}{v[s/x] \approx v[t/x]} \tag{SUB}$$

The following rule enables us to merge, split and/or shorten random strings, the result being indistinguishable from a randomly chosen string of the appropriate length.

$$\frac{}{\mathsf{let}\ \begin{pmatrix} \vec{i} \leftarrow \mathsf{rand}(\vec{p}(\mathbf{n})) \\ x \leftarrow \mathsf{rs}\left(\sum_{j=1}^{k}(p_j(\mathbf{n}) \dot{-} i_j)\right) \end{pmatrix}\ \mathsf{in}\ x \approx \mathsf{let}\ \begin{pmatrix} \vec{i} \leftarrow \mathsf{rand}(\vec{p}(\mathbf{n})) \\ \vec{x} \leftarrow \mathsf{rs}(\vec{p}(\mathbf{n})) \end{pmatrix}\ \mathsf{in}\ \bigcirc_{j=1}^{k} x_{j\{1 \ldots p_j(\mathbf{n}) \dot{-} i_j\}}} \tag{EDIT}$$

where $\bigcirc_{j=1}^{k} t_i$ denotes $t_1 \circ \cdots \circ t_k$.

The following rule has the flavour of an induction rule, but it also captures the basic fact about computational indistinguishability required for hybrid arguments.

$$\frac{\mathsf{let}\ i \leftarrow \mathsf{rand}(p(\mathbf{n}))\ \mathsf{in}\ t \approx \mathsf{let}\ i \leftarrow \mathsf{rand}(p(\mathbf{n}))\ \mathsf{in}\ t[i+1/i]}{t[0/i] \approx t[p(\mathbf{n})/i]} \tag{H-IND}$$

Finally, we have rules which state that $\approx$ is reflexive, symmetric and transitive.

We will now repeat the proof from the previous section that the function f' can be used to define a pseudo-random generator with stretch $\mathbf{n}$. Suppose $f(x) = b(x) \circ r(x)$ is a pseudo-random generator with stretch 1. We can capture this with $\mathsf{let}\ x \leftarrow \mathsf{rs}(\mathbf{n})\ \mathsf{in}\ f(x) \approx \mathsf{rs}(\mathbf{n}+1)$. We then obtain by UNIV and SUB that $\mathsf{let}\ x \leftarrow \mathsf{rs}(\mathbf{n})\ \mathsf{in}\ b(x) \approx \mathsf{rs}(1)$ and $\mathsf{let}\ x \leftarrow \mathsf{rs}(\mathbf{n})\ \mathsf{in}\ r(x) \approx \mathsf{rs}(\mathbf{n})$.

Suppose that f' is defined as above. We want to show that f' is a pseudo-random generator with stretch $\mathbf{n}$. Recall that for any x and i, we have $f'(x, i+1) = b(x) \circ f'(r(x), i)$. By UNIV we then have

$$\mathsf{let}\ \begin{pmatrix} i \leftarrow \mathsf{rand}(\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ f'(x, i+1) \approx \mathsf{let}\ \begin{pmatrix} i \leftarrow \mathsf{rand}(\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ b(x) \circ f'(r(x), i) \tag{$\dagger$}$$

¿From the properties of b and r obtained above, along with SUB we obtain

$$\mathsf{let}\ \begin{pmatrix} i \leftarrow \mathsf{rand}(\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ b(x) \circ f'(r(x), i) \approx \mathsf{let}\ \begin{pmatrix} i \leftarrow \mathsf{rand}(\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ \mathsf{rs}(1) \circ f'(x, i) \tag{$\dagger\dagger$}$$

Define the hybrid function h by $h(z, x, i) = z_{\{1 \ldots \mathbf{n} \dot{-} i\}} \circ f'(x, i)$. We then obtain

$$\begin{aligned}
\mathsf{let}\ \begin{pmatrix} i \leftarrow \mathsf{rand}(\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ h(z, x, i+1) &\approx \mathsf{let}\ \begin{pmatrix} i \leftarrow \mathsf{rand}(\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ z_{\{1 \ldots \mathbf{n} \dot{-} (i+1)\}} \circ f'(x, i+1) && \text{(by UNIV)} \\
&\approx \mathsf{let}\ \begin{pmatrix} i \leftarrow \mathsf{rand}(\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ z_{\{1 \ldots \mathbf{n} \dot{-} (i+1)\}} \circ \mathsf{rs}(1) \circ f'(x, i) \\
& && \text{(by SUB, $\dagger$ and $\dagger\dagger$)} \\
&\approx \mathsf{let}\ \begin{pmatrix} i \leftarrow \mathsf{rand}(\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ z_{\{1 \ldots \mathbf{n} \dot{-} i\}} \circ f'(x, i) && \text{(by EDIT and SUB)} \\
&\approx \mathsf{let}\ \begin{pmatrix} i \leftarrow \mathsf{rand}(\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ h(z, x, i) && \text{(by UNIV)}
\end{aligned}$$

So it follows by H-IND that $\mathsf{let}\ \begin{pmatrix} x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ h(z, x, 0) \approx \mathsf{let}\ \begin{pmatrix} x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix}\ \mathsf{in}\ h(z, x, \mathbf{n})$. But for any z and x with $|z| \leq \mathbf{n}$ we have $h(z, x, 0) = z \circ x$ and $h(z, x, \mathbf{n}) = f'(x, \mathbf{n})$. By UNIV, EDIT and SUB we then obtain $\mathsf{let}\ x \leftarrow \mathsf{rs}(\mathbf{n})\ \mathsf{in}\ f'(x, \mathbf{n}) \approx \mathsf{rs}(\mathbf{n}+\mathbf{n})$, as required.

In Appendix C we give another example in this system, proving that next-bit unpredictability implies pseudorandomness.

5 Conclusions and further work

The simple examples we have given here demonstrate that it is possible to give "direct" proofs, not involving the use of asymptotic notions, of a standard cryptographic construction. In the case of the system with a primitive relation for computational indistinguishability, we were also able to avoid explicit reasoning about probabilities.

There are a number of ways in which we could strengthen the systems presented here for example, adding type-two functionals to the first-order system in order to allow reasoning about protocols and adaptive message attacks; developing proof-theoretic techniques which allow us to extract quantitative information about reductions from correctness proofs; looking at specialized formal systems for other primitive notions (e.g., "one-wayness"). Other directions for future work include examing the strength of the systems we have presented, e.g., by relating the power of the logic for computational indistinguishability to the Abadi-Rogaway logic, or by providing sufficient conditions for protocols to be provably sound.

One way of viewing the systems that we have presented is as a layer between low-level cryptography and higher-level systems for reasoning about security. It would be interesting to formalize this idea, say by providing a proof of the universal composability [10] of a primitive in one of our systems (or an extension theorof), and then using a more abstract system (essentially using information-theoretic reasoning) to reason about the security of protocols which use the primitive. In this way we could provide an end-to-end, cryptographically sound, formalization of security proofs.

We have not considered questions about the decidability of the systems we presented, nor have we seriously considered practical means for automated reasoning using them. While decidability in itself is not a particularly interesting question, automation is an area which demands further investigation.

References

[1] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 148(1):1–70, 1999.

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 2001. To appear.

[3] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. IACR preprint, available at `http://eprint.iacr.org/2003/015/`.

[4] M. Bellare. Practice-oriented provable-security. In E. Okamoto, G. Davida, and M. Mambo, editors, *Proceedings of the 1st International Workshop on Information Security (ISW 97)*, volume 1396 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[5] M. Blum. Coin flipping by telephone: a protocol for solving impossible problems. In *Proceedings of the 24th IEEE Spring Computer Conference, COMPCON*, pages 133–137, 1982.

[6] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[7] R. Boppana and R. Hirschfeld. Pseudo-random generators and complexity classes. In S. Micali, editor, *Advances in Computer Reserch*, volume 5, pages 1–26. JAI Press, 1989.

[8] M. Burrows, M. Abadi, and R Needham. A logic of authentication. *Proceedings of the Royal Society of London A*, 426:233–271, 1989.

[9] S.R. Buss. *Bounded Arithmetic*. Bibliopolis, Naples, 1986.

[10] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computing*, pages 136–145, 2001. Full version available at `http://eprint.iacr.org/2000/067`.

[11] A. Cobham. The intrinsic computational difficulty of functions. In A. Bar Hillel, editor, *Proceedings of the International Conference on Logic, Methodology and Philosophy*, pages 24–30. North Holland, 1965.

[12] S.A. Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the 7th Annual ACM Symposium on the Theory of Computing*, pages 83–97, 1975.

[13] D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, 1983.

[14] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[15] H.J. Keisler. *Elementary Calculus: An Approach Using Infinitesimals*. Prindle, Weber & Schmidt, Boston, MA, second edition, 1986.

[16] P.D. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[17] D. Micciancio and B. Warinschi. Completeness theorems for the abadi-rogaway logic of encrypted expressions. *Journal of Computer Security*, 2003.

[18] J.C. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols (preliminary report). *Electronic Notes in Theoretical Computer Science*, 47:1–31, 2001.

[19] M. Naor. Bit commitment using pseudo-randomness. *Jounral of Cryptology*, 4:151–158, 1991.

[20] R. Parikh. Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, 36(3):494–508, 1971.

[21] P.Y.A. Ryan and S.A. Schneider. *The Modelling and Analysis of Security Protocols: the CSP Approach*. Pearson Education Ltd., Harlow, England, 2001.

[22] A.J. Wilkie. Modèles non standard de l'arithmétique, et complexité algorithmique. In J.P. Ressayre and A.J. Wilkie, editors, *Modèles Non Standard en l'Arithmétique et Théorie des Ensembles*, pages 1–45. Publications Mathématiques de l'Université Paris VII, 1985.

[23] A.C. Yao. Theory and applications of trapdoor functions (extended abstract). In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

A Axioms for poly time functions and counting terms

We first consider axioms for defining poly time functions. We begin with axioms which state that every basic function symbol defines a poly-time function. So, for example, we will have an axiom $\exists f \forall x \forall y (f(x) = x + y)$. We also have axioms for various closure schemes.

For each k and i, $1 \leq i \leq k$, we have the axiom PROJ(k,i)

$$\exists f^k \forall \vec{x}(f(\vec{x}) = x_i)$$

Related to PROJ we have the axiom CONST(k), giving a function which ignores its k inputs and returns the constant 0:

$$\exists f^k \forall \vec{x}(f(\vec{x}) = 0)$$

For each k and l we have the axiom COMP(k,l):

$$\forall g^{k+1} \forall h^{k+l} \exists f^{k+l} \forall \vec{x} \forall \vec{y}(f(\vec{x}, \vec{y}) = g(\vec{x}, h(\vec{x}, \vec{y})))$$

which states that the poly-time functions are closed under composition.

Finally, for each k we have an axiom LRN(k) expressing closure under limited (i.e. size bounded) recursion on dyadic notation:

$$\begin{aligned}
&\forall g^k \forall h^{k+2} \forall b^{k+1} \exists f^{k+1} \forall \vec{x} \forall y(\\
&\quad f(\vec{x}, 0) = g(\vec{x}) \wedge \\
&\quad |h(\vec{x}, y, f(\vec{x}, y))| \leq |b(\vec{x}, y)| \rightarrow (f(\vec{x}, 2y+1) = h(\vec{x}, y, f(\vec{x}, y)) \wedge f(\vec{x}, 2y+2) = h(\vec{x}, y, f(\vec{x}, y))) \wedge \\
&\quad |h(\vec{x}, y, f(\vec{x}, y))| > |b(\vec{x}, y)| \rightarrow (f(\vec{x}, 2y+1) = b(\vec{x}, y)) \wedge f(\vec{x}, 2y+2) = b(\vec{x}, y)))
\end{aligned}$$

We also have axiom schemas for $\#$. First we need to have an upper bound on the value of any counting term

C-1 $\#(|\vec{x}| = |\vec{y}|)\varphi \leq \prod_{1 \leq i \leq k}(y_i \otimes 1)$

We also have axioms relating counting to logical connectives and quantifiers (e.g. inclusion-exclusion).

C-2 $\#(|\vec{x}| = |\vec{y}|)(\varphi \wedge \psi) \leq \#(|\vec{x}| = |\vec{y}|)\varphi$

C-3 $\#(|\vec{x}| = |\vec{y}|)\varphi \leq \#(|\vec{x}| = |\vec{y}|)(\varphi \vee \psi)$

C-4 $\#(|\vec{x}| = |\vec{y}|)(\varphi \vee \psi) + \#(|\vec{x}| = |\vec{y}|)(\varphi \wedge \psi) = \#(|\vec{x}| = |\vec{y}|)\varphi + \#(|\vec{x}| = |\vec{y}|)\psi$

C-5 $(\varphi \rightarrow \psi) \leftrightarrow \#(|\vec{x}| = |\vec{y}|)\varphi \leq \#(|\vec{x}| = |\vec{y}|)\psi$

C-6 $\forall \vec{x}(|\vec{x}| = |\vec{y}| \rightarrow \varphi) \leftrightarrow \#(|\vec{x}| = |\vec{y}|)\varphi = \prod_{1 \leq i \leq k} y_i \otimes 1$

C-7 $\exists \vec{x}(|\vec{x}| = |\vec{y}| \wedge \varphi) \leftrightarrow \#(|\vec{x}| = |\vec{y}|)\varphi > 0$

In general, counting is invariant under permutations of the domain. The following axiom gives a weak version of this invariance. It is sufficient for our current purposes.

C-8 $\forall v(|v| = |y| \rightarrow (\#(|x| = |y| \wedge |\vec{x}| = |\vec{y}|)\varphi = \#(|x| = |y| \wedge |\vec{x}| = |\vec{y}|)\varphi[x \oplus v/x]))$

The following axiom follows from the fact that for any function f, $|\mathrm{rng}(f)| \leq |\mathrm{dom}(f)|$. ($x$ must not occur free in t.)

C-9 $\#(|x| = |y| \wedge |\vec{x}| = |\vec{y}|)(x = t) \leq \#(|\vec{x}| = |\vec{y}|)\top$

We also need the following axioms.

C-10 $\#(|x| = |y| \wedge |\vec{x}| = |\vec{y}|)\varphi = x \otimes 1 \cdot \#(|\vec{x}| = |\vec{y}|)\varphi$, if x does not have a free occurrence in φ.

C-11

$$\#(|x'| = |y'| \wedge |x''| = |y''| \wedge |\vec{x}| = |\vec{y}|)\varphi(x', x'', \vec{x}) = \\ \#(|x| = |y' \circ y''| \wedge |\vec{x}| = |\vec{y}|)\varphi(x_{\{1\ldots|y'|\}}, x_{\{|y'|+1\ldots|y'\circ y''|\}}, \vec{x})$$

Finally for any constant k we have the following axiom:

C-12 $\#(|x| = k \wedge |\vec{x}| = |\vec{y}|)\varphi = \sum_{|x|=k} \#(|\vec{x}| = |\vec{y}|)\varphi$

Here $\sum_{|x|=k} t$ is just meant to be an abbreviation (we are not introducing a new function symbol.)

B Proof of the soundness theorem

The ideas behind this proof go back to work by Parihk [20] on subsystems of Peano arithmetic without exponentiation. We will be using some simple results of model theory which follow Wilkie's paper [22].

Definition B.1 Let M be a model of T. Let $I = \langle I, \mathcal{F}^I \rangle$ be a submodel of M with the following properties:

- $\mathcal{F}^I$ is a the smallest class of functions on I which contains the interpretation in M of the function symbols of T, restricted to I, and which is closed under projection, composition and LRN,
- I is closed under all functions in $\mathcal{F}^I$, and
- if $u \in I$ and $|v| \leq |u|$ then $v \in I$.

I is called an *initial segment* of M.

Proposition B.2 *Suppose I is an initial segment of a model M of T. Let $t(\vec{f}, \vec{x})$ be a term, where $\vec{x}$ consists of all free first-order variables and $\vec{f}$ all free function variables in t (note: we need to talk about free variables because of the counting terms.) For any tuple $\vec{u}$ of elements of I and $\vec{\alpha}$ of elements of $\mathcal{F}^I$, $t(\vec{\alpha}, \vec{u}) \in I$.*

Proof: Proceed by induction on t. For terms built up using function symbols and variables, this follows from the closure of I under $\mathcal{F}^I$. Suppose that t is $\#(|x| = |t'|)\varphi$. Since $|\#(|x| = |t|)\varphi| \leq |t' \otimes 1|$ and, by assumption, $t'(\vec{\alpha}, \vec{u}) \in I$, the result follows from the closure properties of I. The case for more complex counting terms is similar. ■

Proposition B.3 *Suppose I is an initial segment of a model M. Let $\varphi(\vec{f}, \vec{x})$ be a bounded formula with all free variables displayed. For any tuple $\vec{u}$ of elements of I and $\vec{\alpha}$ of elements of $\mathcal{F}^I$,*

$$I \models \varphi(\vec{\alpha}, \vec{u}) \textit{ iff } M \models \varphi(\vec{\alpha}, \vec{u})$$

Proof: The result is immediate for open formulas. The inductive case for Boolean combinations is also immediate. So suppose that the result holds for φ' and φ has the form $\forall |x| \leq |b| \varphi'$. If $M \models \varphi(\vec{\alpha}, \vec{u})$ then $I \models \varphi(\vec{\alpha}, \vec{u})$, as universal formulas are preserved by submodels. Now suppose $I \models \varphi(\vec{\alpha}, \vec{u})$, Consider any $v \in M$ with $|v| \leq |u|$. By the closure properties of I, $v \in I$ and so $I \models \varphi'(\vec{\alpha}, \vec{u}, v)$. By the induction hypothesis $M \models \varphi'(\vec{\alpha}, \vec{u}, v)$. So $M \models \varphi(\vec{\alpha}, \vec{u})$. The argument for bounded existential formulas is similar. ■

Theorem B.4 *If $M \models T + \Sigma$, where all formulas in Σ are bounded, and I is an initial segment of M, then $I \models T + \Sigma$.*

Proof: All axioms other than LIND$^o(p)$ are straightforward, using the closure properties of I and $\mathcal{F}^I$. Formulas in Σ follow from proposition B.3. Fix $u \in I$. Suppose that $I \models \varphi(0)$ and that for every $v < p(|u|)$

$$I \models (\varphi(v) \rightarrow \varphi(v+1)).$$

Since φ is bounded, it follows by proposition B.3 that $M \models \varphi(0)$ and that for every $v < p(|u|)$

$$M \models (\varphi(v) \rightarrow \varphi(v+1))$$

Since $M \models$ LIND$^o(p)$, it follows that for every $w \leq p(|u|)$, $M \models \varphi(w)$. Now since $u \in I$ and I is an initial segment, it follows that $w \in I$ whenever $w \leq p(|u|)$. So by proposition B.3, $I \models \varphi(w)$ for every $w \leq p(|u|)$. ■

We require one more fact about initial segments.

Proposition B.5 *Suppose that I is an initial segment of a model M of T and $f \in \mathcal{F}^I$ is a k-ary function. Then there is a polynomial p such that for all $u_1, \ldots, u_k \in I$, $|f(\vec{u})| \leq p(|\vec{u}|)$.*

Proof: We use induction of the derivation of f. For basic function symbols, we can prove in T that they are bounded by some polynomial in the size of their arguments. For a function defined by projection the result is immediate. For functions defined by LRN we use the induction hypothesis to obtain a polynomial bound for the bounding function. For functions defined by composition, we use the induction hypothesis and simple arithmetic manipulations which must hold in I. ■

We are now in a position to give a proof of the Soundness Theorem, which we state again here:

Theorem B.6 (Soundness Theorem) *Suppose that φ and ψ are bounded formulas such that*

$$T, \forall g \forall \vec{z} \varphi(\vec{f}, g, \vec{z}, \mathbf{s}) \vdash \forall g \forall \vec{z} \psi(\vec{f}', g, \vec{z}, \mathbf{s}) \qquad (*)$$

where $\vec{f}$ and $\vec{f}'$ are (possibly non-disjoint) sequences of function variables. We then have the following for all sequences $\vec{f}, \vec{f}'$ of poly-time functions. If (in $\mathbb{N}$), for every poly-time function g and polynomials $\vec{p}$, there is an n_0 for that for all $x, \vec{z}$ with $|x| \geq n_0$, and $|z_i| = p_i(|x|)$, $\varphi(\vec{f}, g, \vec{z}, x)$, then (in $\mathbb{N}$), for every poly-time function g and polynomials $\vec{p}$, there is an n_0 so that for all $x, \vec{z}$ with $|x| \geq n_0$, and $|z_i| = p_i(|x|)$, $\psi(\vec{f}', g, \vec{z}, x)$.

Proof: Suppose that φ and ψ are bounded formulas satisfying (*). Let $\vec{f}$ and $\vec{f}'$ be sequences of poly-time functions such that in $\mathbb{N}$, for every poly-time function g and polynomials $\vec{p}$, there is an n_0 for that for all $x, \vec{z}$ with $|x| \geq n_0$, and $|z_i| = p_i(|x|)$, $\varphi(\vec{f}, g, \vec{z}, x)$. For the sake of contradiction suppose also that it is not the case that for every poly-time function g and polynomial p, there is an n_0 so that for all $x, \vec{z}$ with $|x| \geq n_0$, and $|z_i| = p_i(|x|)$, $\psi(\vec{f}', g, \vec{z}, x)$.

Let g_0 be a poly time function and $\vec{q}$ a sequence of polynomials which witness the failure of the conclusion. Expand the language so that there is a function symbol for every poly-time function. Let T' be obtained by adding to T every formula of the form $\varphi(\vec{f}, g, 2^{p_1(\mathbf{n})}, \ldots, 2^{p_k(\mathbf{n})}, \mathbf{s})$ where g is a poly-time function and $\vec{p}$ is a sequence of polynomials, along with $\neg\psi(\vec{f}', g_0, 2^{q_1(\mathbf{n})}, \ldots, 2^{q_k(\mathbf{n})}, \mathbf{s})$. Every finite subset of T' is satisfiable because of the asymptotic assumptions about φ and ψ. So by compactness T' has a model M. Define a submodel I with domain consisting of all $u \in M$ such that $|u|$ is polynomial in $\mathbf{n}$. Let $\mathcal{F}^I$ be the smallest class containing the initial functions on I and closed under projection, composition and

bounded primitive recursion on lengths. By proposition B.5 I is an initial segment of M, and so by B.4, $I \models T' \supseteq T$. Furthermore by the construction of I, $I \models \forall g \forall \vec{z} \varphi(\vec{f}, g, \vec{z}, \mathbf{s})$, but $I \not\models \forall g \forall x \forall z \psi(\vec{f}', g, \vec{z}, \mathbf{s})$, giving a contradiction. ■

We note that this proof can be strengthened in order to obtain the existence of a reduction which takes an adversary breaking the construction to an adversary which breaks the primitive. Suppose that the assumption of the theorem hold and that A_2 is an adversary which breaks the construction. We add function symbols for every function which is poly-time definable from A_2, and axioms which state that none of functions break the original primitive. By compactness we obtain a contradiction. In other words we obtain the existence of required reduction. With this stronger soundness result, it follows that our techniques apply to an adversary model where adversaries are modeled by nonuniform families of circuits.

C Examples of proofs

Our first example gives a proof in the system for computational indistinguishability, that next-bit unpredictability implies pseudorandomness. Suppose that $|f(x)| = p(\mathbf{n})$, where $p(\mathbf{n}) > \mathbf{n}$. We say that f is ($p(\mathbf{n})$-)*next bit unpredictable* if

$$\mathsf{let} \begin{pmatrix} i \leftarrow \mathsf{rand}(p(\mathbf{n})) \\ y \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix} \mathsf{in}\ f(y)_{\{1 \ldots i+1\}} \approx \mathsf{let} \begin{pmatrix} i \leftarrow \mathsf{rand}(p(\mathbf{n})) \\ y \leftarrow \mathsf{rs}(\mathbf{n}) \end{pmatrix} \mathsf{in}\ f(y)_{\{1 \ldots i\}} \circ \mathsf{rs}(1)$$

Note that in order to define this notion in our system, we have deviated from the original definition of [6] which is not in presented in terms of computational indistinguishability. However, it is not hard to see that our definition is indeed equivalent to the more standard one.

We will now give a proof in our system of the result of [23], that if f is $2\mathbf{n}$-next bit unpredictable then f is a pseudo-random generator with stretch $\mathbf{n}$. Define the hybrid function h by $h(z, x, i) = f(x)_{\{1 \ldots i\}} \circ z_{\{1 \ldots 2\mathbf{n} \dot{-} i\}}$. We then have

$$\begin{aligned}
\mathsf{let} \begin{pmatrix} i \leftarrow \mathsf{rand}(2\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}2\mathbf{n} \end{pmatrix} \mathsf{in}\ h(x, i) &\approx \mathsf{let} \begin{pmatrix} i \leftarrow \mathsf{rand}(2\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}2\mathbf{n} \end{pmatrix} \mathsf{in}\ f(x)_{\{1 \ldots i\}} \circ z_{\{1 \ldots 2\mathbf{n} \dot{-} i\}} && \text{(by UNIV)} \\
&\approx \mathsf{let} \begin{pmatrix} i \leftarrow \mathsf{rand}(2\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(2\mathbf{n}) \end{pmatrix} \mathsf{in}\ f(x)_{\{1 \ldots i\}} \circ \mathsf{rs}(1) \circ z_{\{1 \ldots 2\mathbf{n} \dot{-} (i+1)\}} && \text{(by EDIT)} \\
&\approx \mathsf{let} \begin{pmatrix} i \leftarrow \mathsf{rand}(2\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(2\mathbf{n}) \end{pmatrix} \mathsf{in}\ f(x)_{\{1 \ldots i+1\}} \circ z_{\{2\mathbf{n} \dot{-} (i+1)\}} \\
& && \text{(by SUB, since } f \text{ is } p\text{-next bit unpredictable)} \\
&\approx \mathsf{let} \begin{pmatrix} i \leftarrow \mathsf{rand}(2\mathbf{n}) \\ x \leftarrow \mathsf{rs}(\mathbf{n}) \\ z \leftarrow \mathsf{rs}(2\mathbf{n}) \end{pmatrix} \mathsf{in}\ h(x, i+1) && \text{(by UNIV)}
\end{aligned}$$

Now since $h(z, x, 0) = z$ and for z with $|z| = 2\mathbf{n}$, $h(z, x, 2\mathbf{n}) = f(x)$, it follows by H-IND, EDIT and SUB that $\mathsf{let}\ x \leftarrow \mathsf{rs}(\mathbf{n})\ \mathsf{in}\ f(x) \approx rs(2\mathbf{n})$, as required.

As a second example, we describe how to use the first-order system to model and verify a protocol for flipping coins on a public line [5], which uses hidden bit commitment as a subroutine. In order to achieve hidden bit commitment, we use a construction of Naor [19], which uses a pseudo-random generator as a primitive.

The following in a high-level description of the protocol: Alice sends Bob a random $v \in_U 0, 1^{3n}$. Bob computes $g(s)$ where s is a random n bit string and g is a prg from n bits to $3n$ bits, and picks a random $b \in_U \{0, 1\}$. To commit to $b = 0$ he sends $g(s)$, to a 1, he sends $g(s) \oplus v$ Then (for coin-flipping) Alice

sends Bob a random $b' \in_U \{0,1\}^n$. Bob reveals s and b and outputs $b \oplus b'$. Alice outputs $b \oplus b'$ if the above checks out; otherwise she rejects.

We will use lower case for the correct protocol and upper-case for a hypothetical possibly cheating message. We can model the correct protocol as 3 functions for each player: $a_1(v, b') = v$ is Alice's first message, $a_2(v, b', B_1) = b'$ is Alice's second message and $a_{out}(v, b', B_1, B_2 = (B, S)) = B \oplus b'$ if $g(S) = B_1 \oplus v$ or $g(S) = B_1$ (depending on whether B is 1 or 0), Reject otherwise. Similarly, $b_1(b, s, A_1) = g(s)$ or $g(s) \oplus A_1$ depending on b, and $b_2(b, s, A_1, A_2) = s$, b, $b_{out}(b, s, A_1, A_2) = b \oplus A_2$.

An adversary for Bob (i.e., a cheating Alice) is a bit T and a pair of functions $A_2(r, b_1(A_1(r), b, s))$ and $A_1(r)$. The adversary succeeds if probability $b_{out}(b, s, A_1, A_2) = T > \frac{1}{2}+$ a non-negligible function. Similarly, a successful cheating Bob, an adversary for Alice, is a bit T and functions $B_1(r, A_1)$, $B_2(r, A_1, A_2)$ so that $a_{out}(v, b', B_1, B_2) = T$ with probability $> \frac{1}{2}+$ non-negligilbe amount.

We now outline how to prove in T that the protocol outlined above is correct, i.e. that there are no successful cheating adversaries. We first consider a cheating Alice. Starting with the formal definition of pseudo-random generator given above, along with the easy to establish fact (using counting axiom C-8) that for any function A, any r_0 and any v with $|v| = 3 \cdot \mathbf{n}$

$$\#\left(\begin{smallmatrix}|y|=3\cdot\mathbf{n}\\|r|=|r_0|\end{smallmatrix}\right) A(r, y \oplus v) = \#\left(\begin{smallmatrix}|y|=3\cdot\mathbf{n}\\|r|=|r_0|\end{smallmatrix}\right) A_2(r, y)$$

we can derive that for any function A, any r_0 and any v with $|v| = 3 \cdot \mathbf{n}$

$$\Pr_{\substack{|r|=|r_0|\\|s|=\mathbf{n}}}[A(r, g(s) \oplus v) = 0] - \Pr_{\substack{|r|=|r_0|\\|s|=\mathbf{n}}}[A(g(s)) = 0] \leq \frac{1}{|z|}$$

It then follows that for any adversaries A_1, A_2, and any r_0 and z

$$\Pr_{\substack{|r|=|r_0|\\|s|=\mathbf{n}}}[A_2(r, b_1(A_1(r), 1, s) = 0] - \Pr_{\substack{|r|=|r_0|\\|s|=\mathbf{n}}}[A_2(r, b_1(A_1(r), 0, s) = 0] \leq \frac{1}{|z|}$$

But then for any adversaries A_1, A_2, and any r_0, z and $T \in \{0,1\}$

$$\Pr_{\substack{|s|=\mathbf{n},|r|=|r_0|\\|b|=1}}[b \oplus A_2(r, b_1(A_1(r), b, s) = T] \leq \frac{1}{2} + \frac{1}{|z|}$$

as required.

For a cheating Bob, we want to show that for any bit $T \in \{0,1\}$ and any pair of functions $B_1(v, r)$ and $B_2(v, r, a_2(v, b', B_1(v, r)))$ a cheating Bob uses,

$$\Pr_{\substack{|v|=3\mathbf{n},|r|=|r_0|\\|b|=1}}[a_{out}(b, v, B_1(a_1(v, b), r), B_2(v, r, a_2(v, b, B_1(a_1(v, b), r))))] = T] \leq 1/2 + 1/|z|.$$

This is purely a counting argument; no complexity theory assumptions are necessary. Using the definitions of $a_1(v, b) = v$, $a_2(v, b, B_1) = b$ we can re-write the probability above as

$$\Pr_{\substack{|v|=3\mathbf{n},|r|=|r_0|\\|b|=1}}[a_{out}(b, v, B_1(v, r), B_2(v, r, b)) = T)].$$

Then substituting the definition of a_{out}, and letting $B_2(v,r,b) = B(v,r,b), S(v,r,b)$, we can write this as

$$\Pr_{\substack{|v|=3\mathbf{n},|r|=|r_0|\\|b|=1}}[g(S(v,r,b)) = B_1(v,r) \oplus v \cdot B(v,r,b) \wedge b \oplus B(v,r,b) = T]$$

$$= \frac{1}{2}(\operatorname*{Exp}_{|v|=3\mathbf{n},|r|=|r_0|}[\#(|b|=1)(g(S(v,r,b)) = B_1(v,r) \oplus v \cdot B(v,r,b) \wedge b \oplus B(v,r,b) = T)])$$

$$= \frac{1}{2}(1 + \Pr_{|v|=3\mathbf{n},|r|=|r_0|}[g(S(v,r,0)) = B_1(v,r) \oplus v \cdot B(v,r,0) \wedge B(v,r,0) = T$$

$$\wedge\, g(S(v,r,1)) = B_1(v,r) \oplus v \cdot B(v,r,1) \wedge B(v,r,1) = \neg T]) \qquad \text{(C-4,C-12)}$$

$$\leq \frac{1}{2}(1 + \Pr_{|v|=3\mathbf{n},|r|=|r_0|}[g(S(v,r,0) = B_1(v,r) \oplus v \cdot T \wedge g(S(v,r,1) = B_1(v,r) \oplus v \cdot (1-T)]) \quad \text{(C-5)}$$

$$\leq \frac{1}{2}(1 + \Pr_{|v|=3\mathbf{n},|r|=|r_0|}[g(S(v,r,0)) \oplus g(S(v,r,1)) = v]) \qquad \text{(C-5)}$$

$$\leq \frac{1}{2}\left(1 + \operatorname*{Exp}_{|v|=3\mathbf{n},|r|=|r_0|}\left[\#\binom{|s_1|=\mathbf{n}}{|s_2|=\mathbf{n}}(g(s_1) \oplus g(s_2) = v)\right]\right) \qquad \text{(C-5)}$$

$$= \frac{1}{2}\left(1 + \frac{1}{2^{3\mathbf{n}}} \cdot \#\binom{|v|=3\mathbf{n}}{|s_1|=\mathbf{n}}{|s_2|=\mathbf{n}}(g(s_1) \oplus g(s_2) = v)\right) \qquad \text{(C-9)}$$

$$= \frac{1}{2}\left(1 + \frac{1}{2^{3\mathbf{n}}}2^{2\mathbf{n}}(1)\right) \qquad \text{(C-9,C-6)}$$

$$= \frac{1}{2}\left(1 + \frac{1}{2^{\mathbf{n}}}\right) \leq \frac{1}{2}\left(1 + 1/|z|\right)$$

The above proof is not completely formalized, but each line follows from the cited counting axioms, or else from BASIC axioms and the definitions of Pr and Exp.

D Soundness via interpretation in the general system

To simplify our presentation, we assume that all probabilistic terms have the form $\mathsf{let}\ b_1\ \mathsf{in}\ \ldots \mathsf{let}\ b_k\ \mathsf{in}\ t$ where t is a term of T not involving counting quantifiers. In a more formal presentation we would need to consider general probabilistic terms and then prove any such term is equivalent to a term in this normal form.

We interpret

$$\mathsf{let}\ b_1\ \mathsf{in}\ \ldots \mathsf{let}\ b_k\ \mathsf{in}\ t \approx \mathsf{let}\ c_1\ \mathsf{in}\ \ldots \mathsf{let}\ c_l\ \mathsf{in}\ s$$

as the following formula of T:

$$\forall A \forall z(\Pr_{b_1,\ldots,b_k}[A(t) = 1] - \Pr_{c_1,\ldots,c_l}[A(s) = 1]) \leq \frac{1}{|z|}$$

Soundness of the system for computational indistiguishability is established by proving the interpretation of each of its rules in T. These proofs use the properties of counting terms in an essential way. We will outline how these proofs go for some of the rules. We will use examples rather than considering the rules in full generality in order to avoid the required induction on syntactic structure which might obscure the underlying ideas behind these proofs. Reflexivity, symmetry and transitivity are obvious.

For UNIV, we have the assumption $Q_1 \ldots Q_k(s = t)$. Suppose for concreteness that we have $\forall |x| = \mathbf{n} \forall 0 \leq i < \mathbf{n}(s = t)$. Note that for any φ

$$\#(|x| = \mathbf{n} \wedge 0 \leq i < \mathbf{n})\varphi = \#(|x| = \mathbf{n} \wedge 0 \leq i < \mathbf{n})(|x| = \mathbf{n} \wedge 0 \leq i < \mathbf{n} \wedge \varphi)$$

From this it follows that for any φ with free variable y,

$$\#(|x| = \mathbf{n} \wedge 0 \leq i < \mathbf{n})\varphi[s/y] = \#(|x| = \mathbf{n} \wedge 0 \leq i < \mathbf{n})\varphi[t/y],$$

The interpretation of the conclusion of the UNIV rule follows as a special case.

For SUB, we have from the interpretation of the premise that

$$\forall A \forall z(\Pr_{b_1,\ldots,b_k}[A(t) = 1] - \Pr_{c_1,\ldots,c_l}[A(s) = 1]) \leq \frac{1}{|z|}$$

For any A, we specialize this formula to poly-time term $\lambda x.A(v(x))$, obtaining

$$\forall z(\Pr_{b_1,\ldots,b_k}[A(v(t)) = 1] - \Pr_{c_1,\ldots,c_l}[A(v(s)) = 1]) \leq \frac{1}{|z|}.$$

Since A is abitrary, we have

$$\forall A \forall z(\Pr_{b_1,\ldots,b_k}[A(v(t)) = 1] - \Pr_{c_1,\ldots,c_l}[A(v(s)) = 1]) \leq \frac{1}{|z|},$$

giving the interpretation of the conclusion of the rule.

EDIT follows fairly directly from the properties of $\#$. For H-IND, we use the Telescoping Sum Lemma 3.4. Let t be the probabilistic term in the premise the H-IND rule. For concreteness, suppose that t has the form $\mathsf{let}\ x \leftarrow \mathsf{rs}(\mathbf{n})\ \mathsf{in}\ \mathsf{let}\ 0 \leq j < \mathbf{n}\ \mathsf{in}\ t'$. Let C be the counting term $\#(|x| = n \wedge 0 \leq j < \mathbf{n})(A(t) = 1)$. From the Telescoping Sum Lemma we have

$$C[\mathbf{n}/i] - C[0/i] = \#(0 \leq i < \mathbf{n})C[i + 1/i] - \#(0 \leq i < \mathbf{n})C.$$

This is sufficient for establishing the interpretation of H-IND in T.