On Hiding Information from an Oracle*

Martín Abadi†
DEC Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301

Joan Feigenbaum‡
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Joe Kilian§
MIT
545 Technology Square
Cambridge, MA 02139

Abstract: We consider the problem of *computing with encrypted data*. Player A wishes to know the value $f(x)$ for some x but lacks the power to compute it. Player B has the power to compute f and is willing to send $f(y)$ to A if she sends him y, for any y. Informally, an *encryption scheme* for the problem f is a method by which A, using her inferior resources, can transform the *cleartext instance* x into an *encrypted instance* y, obtain $f(y)$ from B, and infer $f(x)$ from $f(y)$ in such a way that B cannot infer x from y. When such an encryption scheme exists, we say that f is *encryptable*.

The framework defined in this paper enables us to prove precise statements about what an encrypted instance hides and what it leaks, in an information-theoretic sense. Our definitions are cast in the language of probability theory and do not involve assumptions such as the intractability of factoring or the existence of one-way functions. We use our framework to describe encryption schemes for some well-known functions.

We also consider the following generalization of encryption schemes. Player A, who is limited to probabilistic polynomial time, wishes to guess the value $f(x)$ with probability at least $\frac{1}{2} + 1/|x|^c$ of being correct, for some constant c. Player B can compute any function and generate arbitrary probability distributions. Players A and B can interact for a polynomial number of rounds by sending polynomial-sized messages. We prove a strong

*An extended abstract of this paper appeared in the Proceedings of the Nineteenth Annual ACM Symposium on the Theory of Computing ([AFK]).

†Part of the work of the first author was done at the Stanford Computer Science Department and supported by an IBM Graduate Fellowship.

‡Part of the work of the second author was done at the Stanford Computer Science Department and supported by a Xerox Corporation Fellowship and a grant from the AT&T Bell Laboratories Graduate Research Program for Women.

§The third author is supported by a Hertz Foundation Fellowship and NSF grant #DCR MCS8509905.

negative result: there is no such generalized encryption scheme for SAT that leaks at most $|x|$ (unless the polynomial hierarchy collapses at the third level).

1. Introduction

Suppose your private computer lacks the power needed to compute a function of importance to you. You have access to a computing center that does have the resources to compute this function. You would like to take advantage of these public resources but not to expose your confidential data. Is it possible to encrypt the data before sending it to the computing center and still to make sense out of the answer returned?

We consider the general problem of *computing with encrypted data*. Player A wishes to know the value $f(x)$ for some x but lacks the power to compute it. Player B has the power to compute f and is willing to send $f(y)$ to A if she sends him y, for any y. Informally, the function f is *encryptable* if there are two efficiently computable functions E and D such that E maps the *cleartext instance* x to an *encrypted instance* y, D maps $f(y)$ to $f(x)$, and nothing about x can be inferred from y, even with unbounded computing power. Note that player B has enough power to compute the hard function f and therefore presumably to crack a cryptosystem based on intractability assumptions. Furthermore, B may not guarantee the privacy of the encrypted data: for example, nothing sent to a public computing center is secure. In short, we do not know who will see the encrypted data or how he may try to tamper with it, but we know that he will be thwarted as long as x is hidden information-theoretically.

The requirement that the encrypted instance y reveal *nothing* about the cleartext instance x may be unrealistic and unnecessary. For example, A may only be concerned about concealing the high-order bits of x. In this case, a pair of functions E and D that allow her to obtain $f(x)$ and conceal the high-order bits of x would be acceptable, even if B could infer something else about x from y, such as the approximate size of x or the least significant bit of x. We develop the machinery needed to prove that E *hides* at least some function $H(x)$ or that E *leaks* at most some function $L(x)$.

In this paper, we give examples of natural encryptable functions and establish a strong negative result about the encryptability of NP-Hard functions. There are four motivations for this work. First, there is the immediate, practical application of the computing center scenario. Second, encryption schemes can be used as building blocks in cryptographic protocols. Third, it is interesting that we can establish precise negative results, because these are difficult to come by in cryptography. Fourth, encryptability turns out to be related to fundamental concepts in complexity theory, such as nonuniformity and random-self-reducibility.

A typical result connecting encryptability to complexity is: the functions for which there exist efficient encryption schemes that leak nothing are exactly the functions computable in expected polynomial time. Although not difficult, this theorem is important, because it supports the intuition that functions that are harder to compute are harder to encrypt. More precisely, if f is hard to compute, then A cannot hope to hide everything about x.

In order to obtain general results about the encryptability of a wider class of functions, we examine what happens when $L(x)$ is nontrivial. For example, we investigate which functions can be encrypted if A is willing to leak the *size* of x. The first observation to make is that any naïve approach to a general result about these functions is doomed to fail. Consider the function $f(x)$ that is 1 if the Turing Machine encoded by $|x|$ halts on all inputs and 0 otherwise. Then f is not recursive, but A can certainly consult B while hiding everything about x except $|x|$: she can choose a random y for which $|y| = |x|$ and ask for $f(y)$.

This example shows that it is not obvious how to establish a precise theorem about what functions can be encrypted leaking at most $|x|$. There is no simple connection between how hard f is to encrypt and how hard it is to compute.

However, we can show a precise relationship between how hard f is to encrypt and how hard f is to compute with the aid of a *polynomial advice function*, as developed by Karp and Lipton in their study of nonuniform complexity classes ([KL]). This relationship leads to a negative result that holds for *generalized encryption schemes* (GES's), which are defined precisely in the next section:

Main Theorem: No NP-Hard function has an efficient GES that leaks at most $L(x)$, for any L such that $|L(x)| = O(\log |x|)$ and L can be computed in nondeterministic polynomial time with the aid of a polynomial advice function, unless the polynomial hierarchy collapses at the third level. For instance, SAT cannot be encrypted leaking at most $|x|$.

To prove this theorem, we show that, for any such L, a function that is encryptable leaking at most $L(x)$ is itself computable in nondeterministic polynomial time with the aid of a polynomial advice function. We then apply a theorem of Yap that states that, if both yes-instances and no-instances of SAT are recognizable in nondeterministic polynomial time with polynomial advice, then the polynomial hierarchy collapses at the third level ([Yap]).

The technique we use to connect encryptability to nonuniform complexity can also be used to analyze random-self-reducibility ([AL, TW]). Intuitively, a random-self-reduction is a randomized polynomial-time reduction from a set to itself that maps an element of size n to each other element of size n with equal probability. We define random-self-reducibility precisely in Section 5. Our framework enables us to prove some new results about it. For example, if a set is random-self-reducible, then it can be recognized in NP with the help of a polynomial advice function.

Computing with encrypted data was discussed in full generality for the first time in [Fei]. Restricted versions were considered in [BM, RAD]. The framework that we establish here is superior to the one in [Fei], which did not include formal definitions of leaking and hiding. The definition of encryption in [Fei] was too weak; it was satisfied by schemes that leaked some crucial information about the cleartext instance x. Here, we impose stricter requirements on encryption schemes, and we obtain more meaningful and intuitive results.

Yao's *discreet functions* ([Yao86b]) are essentially functions with generalized encryption schemes that leak at most the size of the cleartext instance x. Yao shows that there exist

functions in DSPACE($\Omega(2^{n^{1+\epsilon}})$) that cannot be encrypted in this sense. In contrast, we show that *no* NP-Hard function can be encrypted in this sense, unless the polynomial hierarchy collapses.

Presentations of our work have elicited questions about its relationships to the results in [Yao82, Yao86a] and to zero-knowledge proof systems (e.g., [GHY89, GMR]). In short, despite surface resemblances, there are fundamental differences between our work and that reported in [GHY89, GMR, Yao82, Yao86a]. We address these differences in Appendix A below.

In the next section, we define encryptability. In Section 3, we present nontrivial examples of encryption schemes and an application to secure circuit evaluation. Section 4 contains our main results about encryptability and complexity. We define and explore random-self-reducibility in Section 5. We state some open problems in Section 6. Appendix B contains the proof of a complexity-theoretic lemma that was stated without proof in [AFK]; although the lemma is not needed for the results given here, we include it because of its potential independent interest.

2. Basic definitions

Throughout this paper, f is the function that A is interested in but lacks the resources to compute. The cleartext instance x is in $Dom(f)$, and a *key* k is drawn from a suitable set $\mathcal{K}$. The encryption function $E : Dom(f) \times \mathcal{K} \rightarrow Dom(f)$ and the decryption function $D : Dom(f) \times \mathcal{K} \times Range(f) \rightarrow Range(f)$ are both partial, deterministic functions. When A wants to compute the value $f(x)$ with the help of B, she chooses a key k, computes $y = E(x, k)$, sends y to B, gets back $f(y)$, and computes $f(x) = D(x, k, f(y))$.

Let X, K, and Y be random variables defined on $Dom(f)$, the key space $\mathcal{K}$, and $Range(E)$, respectively. The *a priori* probability that A wants to know the value $f(x)$ is $P(X = x)$. A uses a sampling procedure $k()$ to draw keys from $\mathcal{K}$ according to the distribution of K. The distribution of Y is induced by the joint distribution of X and K, namely, $P(Y = y) = \sum_{E(x,k)=y} P(X = x, K = k)$.

We assume that B knows the encryption scheme and the joint distribution of X and K. The functions L and H, both defined on $Dom(f)$, are such that B can compute $L(x)$ if he knows y but cannot compute $H(x)$. Just as $Y = E(X, K)$ is a random variable induced by X and K, so $L(X)$ and $H(X)$ are random variables induced by X.

The following definitions express formally what properties E must have if A is to be able to hide at least H while leaking at most L.

Definition: An encryption function E for f *leaks* at most L if, for all *a priori* distributions on X, for all $z \in$ Range(L), the random variables X and Y are independent given $L(X) = z$.

Definition: The encryption function E for f *hides* at least H if, for all *a priori* distributions on X, the random variables $H(X)$ and Y are independent.

In other words, if E encrypts f leaking L, then B learns nothing new about x from the

encrypted instance y if he already knows that $L(x) = z$. Similarly, if E hides H, then A reveals no information about $H(x)$ when she sends y to B.

Our main theorem holds for generalized encryption schemes (GES's), which we now define. As before, a GES for f entails deterministic encryption and decryption algorithms E and D, and a randomized key-sampling algorithm $k()$. In a GES, however, A's queries to the oracle B need not be elements of $Dom(f)$, and B's responses need not be elements of $Range(f)$; they can be drawn from arbitrary sets, which we denote by $\mathcal{A}$ and $\mathcal{B}$.

Definition: A *generalized encryption scheme* (GES) is a two-party protocol with the following properties.

(I) There are m rounds of communication. Denote A's i^{th} message by a_i and B's i^{th} message by b_i.

(II) On cleartext input x, A computes an encryption key $k = k(x)$ before she initiates communication with B. This one key is used by A in each round and in decryption.

(III) The encryption algorithm E takes inputs of the form $(x, k, \overline{b})$, where $x \in Dom(f)$, $k \in \mathcal{K}$, and $\overline{b}$ is a list of elements of $\mathcal{B}$, and produces outputs in $\mathcal{A}$. In round i of the protocol, A computes $a_i = E(x, k, \overline{b})$ and sends it to B; the list $\overline{b}$ must be of length $i - 1$ and consist of B's responses $b_1, \ldots, b_{i-1}$ to A's previous queries. Thus, in round 1, the list $\overline{b}$ is empty, and the query a_1 is a function of x and k.

(IV) B draws his responses $b_1, \ldots, b_m$ from any distribution on $\mathcal{B}$ that satisfies property (V) below.

(V) The decryption algorithm D takes inputs of the form $(x, k, \overline{b})$, where $\overline{b} = b_1, \ldots, b_m$ is a list of m elements of $\mathcal{B}$. If k and $\overline{b}$ are the results of an execution of the protocol, then $D(x, k, \overline{b}) = f(x)$ with probability at least $1/2 + 1/|x|^c$, for some constant c.

To extend the definitions of leaking and hiding so that they apply to GES's, simply replace the random variable Y with a vector of random variables $\langle Y_1, \ldots, Y_m \rangle$, where m is the number of rounds.

Of course, we are particularly interested in encryption schemes that are practical as well as effective. Thus, in this paper, we restrict attention to schemes in which E and D terminate in polynomial time, $k()$ terminates in expected polynomial time, we can check in polynomial time whether a particular key k is valid for a cleartext instance x, and $|k(x)|$, m, and $|b_1|$ through $|b_m|$ are polynomial in $|x|$.[1] It is possible to develop an analogous

[1] Throughout this paper, we mean the following when we say that a function f is computable in expected polynomial time. There is an algorithm ALG_f that always returns the correct value $f(x)$ on any input $x \in Dom(f)$. The algorithm ALG_f uses a source of randomness, and its expected running time, which is a function of the input x and the distribution on these random choices, is bounded by a polynomial in $|x|$, for all inputs x, including those that are not in $Dom(f)$. Note that we do *not* mean that the expected running time is computed with respect to a distribution on the inputs. For example, if f is the characteristic function for a set S, and f is computable in expected polynomial time, then $S \in$ ZPP; in this case, $Dom(f)$ is all of $\{0,1\}^*$. If ALG is an algorithm that computes a sample point of a distribution instead of a single-valued function (e.g., the key-sampling algorithm $k()$ of an encryption scheme), then the same requirements hold: ALG terminates in expected polynomial time on all inputs and gives a correct answer on all meaningful inputs.

theory in which everything is "scaled up," i.e., the functions E, D, and $k()$ are of higher complexity, keys are longer, and there is a larger number of rounds of interaction with longer messages.

On meaningless inputs, such as strings x that are not in $Dom(f)$ or lists $\overline{b}$ of answers that a correct player B would never have produced in response to a sequence of queries about x, the functions E, D, and $k()$ are required to terminate as quickly as they do on meaningful inputs, but they are not required to produce meaningful output. In particular, they are not required to recognize that an input x is not in $Dom(f)$ or to detect that a particular message b_i could not have been produced by a correct oracle B. Indeed, they cannot be required to do so, because we give encryption schemes in Section 3 for functions f whose domains are not known to be recognizable in polynomial time, and player B, by definition, has sufficient computational resources to produce responses that A cannot produce or even verify.

In the case in which f is a predicate, we discuss briefly encryption schemes that may leak extra information about no-instances. We relate these schemes to the theory of random-self-reducibility.

3. Examples and applications

Here we give encryption schemes for three well-known number-theoretic functions. The natural reductions that we use for encryption have been used before in the study of discrete logarithms and quadratic residues (e.g., [AL]).

The Discrete Logarithm Problem (DLP):

The DLP function f takes as arguments a prime p, a generator g for Z_p^*, and an integer u such that $(u, p) = 1$. The value $f(u, g, p)$ is the unique exponent $e \in [1, p-1]$ for which $g^e \equiv u \bmod p$. The key space $\mathcal{K}$ used in the encryption scheme is Z. The answer to the encrypted instance is denoted by e'.

$k(\langle u,\ g, p\rangle)$
{
  Return(An element of $[1, p-1]$ chosen uniformly at random);
}
$E(\langle u,\ g,\ p\rangle, k)$
{
  Return($\langle u \cdot g^k \bmod p,\ g,\ p\rangle$);
}
$D(\langle u,\ g,\ p\rangle, k, e')$
{
  Return($e' - k \bmod p - 1$);
}

The Quadratic Residuosity Problem (QRP):

The QRP function f takes two arguments, an integer n that is the product of two distinct primes p and q, with $p \equiv q \equiv 3 \bmod 4$, and an integer u such that (u, n) is 1 and the Jacobi Symbol $(\frac{u}{n})$ is $+1$. (We could use a wider class of n's – for example, the so-called Blum integers – but the increased generality is not needed to illustrate encryptability.) The value $f(u, n)$ is 1 if there is an integer a such that $a^2 \equiv u \bmod n$, and $f(u, n)$ is 0 otherwise. Note that the Jacobi symbol $(\frac{z}{n})$ can be computed in polynomial time using the Reciprocity Law and that, if n has the specified form, then $f(-1, n)$ is 0. We use $Z_n^*[+1]$ to denote the integers in $[1, n-1]$ that are relatively prime to n and have Jacobi symbol $+1$. The key space $\mathcal{K}$ is $Z \times \{0, 1\}$, and the answer to the encrypted instance is denoted by e'.

$k(\langle u,\ n\rangle)$
{
    Do
        Choose z uniformly at random from $[1, n-1]$
    While $(z, n) > 1$ **or** $(\frac{z}{n}) = -1$;
    Choose ϵ uniformly at random from $\{0, 1\}$;
    Return$(\langle z,\ \epsilon\rangle)$;
}
$E(\langle u,\ n\rangle, \langle z, \epsilon\rangle)$
{
    If $\epsilon = 0$
        Then Return$(\langle uz^2 \bmod n,\ n\rangle)$;
        Else Return$(\langle -uz^2 \bmod n,\ n\rangle)$;
}
$D(\langle u,\ n\rangle, \langle z, \epsilon\rangle, e')$
{
    If $\epsilon = 0$
        Then Return(e');
        Else Return$(1 - e')$;
}

The Primitive Root Problem (PRP):

The PRP function f takes as arguments a prime p and an integer u that is relatively prime to p. The value $f(u, p)$ is 1 if u generates the cyclic group Z_p^* (i.e., if u is a primitive root mod p), and 0 otherwise. The key space $\mathcal{K}$ is Z, and once again the answer to the encrypted instance is denoted by e'.

$k(\langle u,\ p\rangle)$
{
    Do
        Choose an odd k uniformly at random from $[1, p-1]$
    While $(k, p-1) > 1$;
    Return(k);
}
$E(\langle u,\ p\rangle, k)$
{

Return($\langle u^k \bmod p,\ p\rangle$);
}
$D(\langle u,\ p\rangle, k, e')$
{
Return(e');
} /* $f(u^k \bmod p, p) = f(u,p)$ if $(k, p-1) = 1$ */

In all of these encryption schemes, the functions E, D, and $k()$ have the required properties, i.e., E and D terminate in polynomial time, $k()$ terminates in expected polynomial time and produces keys of length polynomial in $|x|$, and the fact that a key k is valid for a cleartext instance x can be checked in polynomial time. To see that the function $k()$ in the PRP scheme runs in expected polynomial time, note that $p-1$ has fewer than $\log p$ distinct prime factors, whereas the total number of distinct primes less than p is proportional to $p/\log p$; hence the number of valid keys for each pair $\langle u,\ p-1\rangle$ can be bounded below trivially by $\Omega(p/\log p)$, and we can expect to find one by trial and error in $O(\log p)$ trials.

Theorem 1: The encryption scheme given for the DLP leaks at most g and p. The scheme for the QRP leaks at most n. The scheme for the PRP leaks at most p and the order of u mod p.

Proof:

The crucial idea in all three proofs is that, as long as k is chosen uniformly, the distribution of Y given $L(X)$ is also uniform, regardless of the *a priori* distribution of X. The proofs of the first two statements are nearly identical, and so we only give the first.

Consider the random variables X and Y given $L(X) = \langle g,\ p\rangle$. We wish to show that they are independent. Let $x = \langle u,\ g,\ p\rangle$ and $y = \langle v,\ g',\ p'\rangle$, where $\langle g',\ p'\rangle \neq \langle g,\ p\rangle$. For any such y,

$$P(X = x, Y = y \mid L(X) = \langle g,\ p\rangle)$$

and

$$P(X = x \mid L(X) = \langle g,\ p\rangle) \cdot P(Y = y \mid L(X) = \langle g,\ p\rangle)$$

are the same, because they are both 0. Thus, we may assume that $x = \langle u,\ g,\ p\rangle$ and $y = \langle v,\ g,\ p\rangle$ and deal with unconditional probabilities to prove independence.

For each such x and y, there is a unique key k such that $E(x,k) = y$. This key is $f(vu^{-1} \bmod p,\ g,\ p)$, and we denote it k_{xy}. Note that k is chosen independently of u. For

all such y,

$$\begin{aligned} P(Y = y) &= \sum_{x=\langle u,\ g,\ p\rangle} P(X = x, K = k_{xy}) \\ &= \sum_{x} P(X = x) \cdot P(K = k_{xy}) \\ &= \sum_{x} P(X = x) \cdot \frac{1}{p-1} \\ &= \frac{1}{p-1} \cdot \sum_{x} P(X = x) \\ &= \frac{1}{p-1}. \end{aligned}$$

We complete the proof of independence as follows.

$$\begin{aligned} P(X = x, Y = y) &= P(X = x, K = k_{xy}) \\ &= P(X = x) \cdot P(K = k_{xy}) \\ &= P(X = x) \cdot \frac{1}{p-1} \\ &= P(X = x) \cdot P(Y = y). \end{aligned}$$

The encryption function E given for PRP leaks more than those given for DLP and QRP. Let $order_p(u)$ be the smallest exponent e such that $u^e \equiv 1 \bmod p$ and let $O_p(u) = \{z \in Z_p^*;\ order_p(z) = order_p(u)\}$. In the scheme given, if $\langle v,\ p\rangle = E(\langle u,\ p\rangle,\ k)$, then $order_p(v) = order_p(u)$, and thus E leaks $order_p(u)$.

As in the proof for the DLP scheme, we may assume that $x = \langle u,\ p\rangle$ and $y = \langle v,\ p\rangle$, where $order_p(u) = order_p(v)$, for otherwise

$$P(X = x, Y = y \mid L(X) = \langle p,\ order_p(u)\rangle)$$

and the product

$$P(X = x \mid L(X) = \langle p,\ order_p(u)\rangle) \cdot P(Y = y \mid L(X) = \langle p,\ order_p(u)\rangle)$$

are the same, because they are both 0. For any such x, the key k may assume $\phi(p-1)$ distinct values, exactly $\phi(p-1) \cdot |O_p(u)|^{-1}$ of which satisfy $u^k \equiv v \bmod p$. Let $\mathcal{K}_{xy}$ denote this set of keys. Note that, once again, k is chosen independently of u; the probability that a key in $\mathcal{K}_{xy}$ is chosen is $|\mathcal{K}_{xy}| \cdot (\phi(p-1))^{-1}$, that is, $|O_p(u)|^{-1}$.

$$\begin{aligned} P(Y = y) &= \sum_{x} P(X = x, K \in \mathcal{K}_{xy}) \\ &= \sum_{x} P(X = x) \cdot P(K \in \mathcal{K}_{xy}) \\ &= \sum_{x} P(X = x) \cdot |O_p(u)|^{-1} \\ &= |O_p(u)|^{-1} \cdot \sum_{x} P(X = x) \\ &= |O_p(u)|^{-1}. \end{aligned}$$

The proof of independence follows immediately:

$$\begin{aligned} P(X = x, Y = y) &= P(X = x, K \in \mathcal{K}_{xy}) \\ &= P(X = x) \cdot P(K \in \mathcal{K}_{xy}) \\ &= P(X = x) \cdot |O_p(u)|^{-1} \\ &= P(X = x) \cdot P(Y = y). \end{aligned}$$

∎

We can use the fact that our scheme for encrypting PRP instances leaks at most p and $order_p(u)$ to devise a scheme for DLP that leaks at most p. In the revised scheme, A chooses c_1, uniformly from $\{1, \ldots, p-1\}$, and c_2, uniformly from the integers less than p and relatively prime to $p-1$. The encrypted instance that she sends to B is $\langle u',\ g',\ p \rangle$, where $g' \equiv g^{c_2} \bmod p$ and $u' \equiv u \cdot g^{c_1} \bmod p$. If $e' = f(\langle u',\ g',\ p \rangle)$, then $f(\langle u,\ g,\ p \rangle) = c_2 e' - c_1$.

These three examples are meant to illustrate our ideas and not to be an exhaustive list of encryptable functions. Others, such as inverse-RSA, can be encrypted using similar schemes.

Before proceeding to our main results, which relate encryptability to computational complexity, we mention an application of the results in this section. We propose a protocol for *secure circuit evaluation* that is related to the work in [CDvdG, GHY87, GMW87, Yao82, Yao86a].

Suppose that we make the following changes to our model. Instead of assuming that B has oracular power, say that he has knowledge of a secret algorithm — more concretely, that he has a circuit that computes f. In this new model, we seek a protocol by which B can guide A through the circuit, ultimately revealing $f(x)$ to A, while A conceals x from B and B conceals the circuit from A.

In [AF], Abadi and Feigenbaum use our encryption function for the QRP as a building block in a secure-circuit-evaluation protocol that is much simpler than the ones that have appeared previously. In this protocol, the circuit for f is hidden unconditionally from A, while the data x remain confidential only under the Quadratic Residuosity Assumption (QRA). It is also shown in [AF] that some complexity-theoretic assumption, although not necessarily the QRA, is needed to prove the security of a circuit evaluation protocol.

4. Main results: encryptability and complexity

We turn now to some basic results about the connection between the difficulty of encrypting instances of f and the complexity of computing f.

Theorem 2: If f is computable in expected polynomial time, then f is encryptable hiding x. Conversely, if there is an encryption scheme for f that hides $|x|$, then f is computable in expected polynomial time.

Proof: Suppose that f is computable in expected polynomial time. We exhibit an encryption scheme for f that hides x. Let x_0 be a fixed element of $Dom(f)$.

```
k(x)
{
    Return(f(x));
}
E(x, k)
{
    Return(x_0);
}
D(x, k, e')
{
    Return(k);
}
```

In other words, if we make the (natural) assumption that A can compute everything that is computable in expected polynomial time, then she can just compute the function herself and not reveal anything to B.

Now suppose that f is encryptable hiding $|x|$, with encryption function E, decryption function D, and key space $\mathcal{K}$. Let x_0 be a fixed element of $Dom(f)$. Because E runs in polynomial time, there is a fixed polynomial p_0 such that, for any $k \in \mathcal{K}$, if $E(x_0, k)$ is defined, it is bounded in length by $p_0(|x_0|)$. Let $Y_0 = \{y_1, \ldots, y_T\}$ be the set of instances of f that have size at most $p_0(|x_0|)$. Note that T is a constant. The encryption function E cannot map x_0 to an instance outside Y_0.

Let x be an arbitrary cleartext instance. Because E hides $|x|$, it cannot take x to an instance outside Y_0. Thus, the range of E is finite, and we can precompute the values $\epsilon_y = f(y)$ for $y \in Range(E)$. We can now state an expected polynomial-time algorithm for f.

```
f(x)
{
    Choose a key k; /* This is the expected polynomial-time step. */
    y := E(x, k);
    Lookup ε_y;
    Return(D(x, k, ε_y));
}    ∎
```

Using a similar proof, we could characterize the functions with GES's that leak nothing, but this simpler version of the theorem illustrates its main point: if A can hide the size of an instance, then she can hide everything about the instance; however, if this is the case, then f must be easy to compute. One might hope for a generalization of Theorem 2 to say that functions that are harder to compute are also harder to encrypt. This hope was dashed by the Turing Machine example of Section 1.

That example generalizes as follows: if $f(x) = g(|x|)$, for some g, then f is encryptable leaking $|x|$. By considering functions g of complexity $\Theta(2^{2^n})$, $\Theta(2^{2^{2^n}})$, etc., we get a family of increasingly difficult f's, all encryptable leaking $|x|$. At first glance, this family of functions seems pathological. The results below show that this intuition is correct.

We now temporarily restrict attention to boolean functions f for which $Dom(f) = \{0,1\}^*$; we relax the first restriction at the end of this section. Suppose that f is the characteristic function for the set S. When we refer to an encryption scheme for S, we mean an encryption scheme for f; when we say that f is in a certain complexity class, we mean that S is in that class.

Assuming that the polynomial hierarchy (PH, [Sto]) does not collapse at the third level, we show that SAT is not encryptable leaking $|x|$.

These results rely on a connection between encryptability and nonuniform complexity. Specifically, we make use of the theory of polynomial advice functions developed by Karp and Lipton ([KL]). For convenience, we repeat the necessary definitions.

If $S \subseteq \{0,1\}^*$ and h is a function $\mathcal{N} \rightarrow \{0,1\}^*$, then $S : h = \{x \text{ s.t. } h(|x|) \cdot x \in S\}$, where the symbol $\cdot$ denotes concatenation of strings. If V is a class of sets and F a class of functions, then $V/F = \{S : h \text{ s.t. } S \in V \text{ and } h \in F\}$. Intuitively, V/F is the class of sets that can be recognized given the power needed to recognize sets in V plus an amount of advice bounded in length by functions in F. Examples of well-studied nonuniform classes are P/*log* and P/*poly*, which is equivalent to the class of sets accepted by families of small circuits ([Pip]).

Before proceeding to the proof of our main theorem, we can explain the Turing Machine example of Section 1 and its generalizations described above. Any function of x whose value depends only on $|x|$ is clearly encryptable leaking $|x|$. It is also clearly in P/*poly*. Karp and Lipton showed that no NP-Hard function is in P/*poly*, assuming that the PH does not collapse at the second level. Hence, under the same assumption, no NP-Hard function is trivially encryptable leaking $|x|$ by virtue of having values determined only by $|x|$. This still leaves open the possibility that an NP-Hard function could have a nontrivial encryption scheme that leaks $|x|$. In the proof of our main theorem, we eliminate this possibility, under a slightly stronger assumption than that of Karp and Lipton, i.e., that the PH does not collapse at the third level.

Theorem 3: If SAT has a GES that leaks at most $|x|$, then the PH collapses at the third level.

It suffices to prove:

Lemma 1: If f has a GES that leaks at most $|x|$, then f is in the nonuniform complexity class NP/*poly* $\cap$ CoNP/*poly*.

Lemma 2: If SAT is in CoNP/*poly* then NP is contained in CoNP/*poly*.

Theorem 3 then follows directly from a special case of [Yap, Theorem 2]; namely, if NP is contained in CoNP/*poly*, the PH collapses at the third level.

In [AFK], we announced that, if SAT has a GES that leaks at most $|x|$, then the PH collapses at the second level. If the conclusion of Lemma 1 were that f is in the nonuniform complexity class (NP $\cap$ CoNP)/*poly*, then this stronger theorem than the one given here would follow from Lemma 3 of [AFK]. However, as we state here, the proper conclusion of Lemma 1 is that f is in NP/*poly* $\cap$ CoNP/*poly*, which is not know to be equal to (NP $\cap$ CoNP)/*poly*. Hence, the strongest statement we can make is that, if SAT has a GES that

leaks at most $|x|$, then the PH collapses at the third level. Although it is no longer needed, we give the proof of our former Lemma 3 because of its potential independent interest; see Appendix B.

Proof of Lemma 1:

The proof relies on three basic tools: universal hashing ([CW]), polynomial advice functions, and counting. Because the entire proof is quite long, we first consider a simple case that illustrates its basic spirit. Suppose that E, D, and $k()$ comprise an encryption scheme for f that leaks at most $|x|$; there is only one round of communication, B returns $f(y)$, for the encrypted instance y, and A always gets the correct value $f(x)$ when she runs the decryption algorithm. Intuitively, the proof goes as follows. For each n, let $y_n = E(0^n, k_n)$, where 0^n is used as a generic, fixed instance of size n and k_n is a key valid for 0^n. We can obtain y_n and the value $f(y_n)$ from a polynomial advice function. Once we have y_n and $f(y_n)$, we can find $f(x)$ for any x of length n in nondeterministic polynomial time, using this algorithm:

$f(x)$ /* Assume you know y_n and $f(y_n)$, where $n = |x|$. */
{
  Choose a key k such that $E(x, k) = y_n$;
  Return$(D(x, k, f(y_n)))$;
}.

The crucial point is that if there is a key k_n such that $E(0^n, k_n) = y_n$, then there must be a key k such that $E(x, k) = y_n$, because E hides everything but the size of x.

Now suppose that f has a GES that leaks no more than $|x|$, which we denote by n for the rest of the proof. Call this GES G and its key-sampling algorithm $k()$. Our goal is to show that this weaker hypothesis still guarantees the existence of a polynomial advice function with which we can compute f nondeterministically in polynomial time. The fact that a GES allows polynomially many rounds of communication and allows B to return responses drawn from arbitrary distributions poses no barrier to the construction of an advice function. The difficulty arises because A may get a wrong answer when she runs the decryption algorithm. We need some additional terminology and notation to address this difficulty.

Define a *transcript* t to be a sequence $(a_1, b_1), \ldots, (a_m, b_m)$ of question/answer pairs exchanged during an execution of an m-round GES. Player A's guess for the value $f(x)$ will be based on x, the random key k, and the transcript t that results from A's conversation with B. (Recall that A uses only one encryption key in a GES and that she chooses it before sending her first message a_1 to B.) We denote this guess by $A(x, k, t)$. For any x, transcripts occur according to a probability distribution. Because G leaks only n, this distribution must be the same for all x of size n, and hence we denote by T_n the random variable that assumes this distribution.

We say that transcript t is *feasible* for cleartext instance x and key k if there is a nonzero probability that the dialogue between A and B is the one denoted by t. We use $\kappa(x, t)$ to denote the keys k such that t is feasible for x and k. Note that we can determine in polynomial time whether k is in $\kappa(x, t)$, because A can be simulated relative to the

sequence of B's responses in t.

We call a key k *good for* (x, t) if k is in $\kappa(x, t)$ and $A(x, k, t) = f(x)$; otherwise, we say that k is *bad for* (x, t). We denote the sets of good and bad keys for (x, t) by $\kappa_g(x, t)$ and $\kappa_b(x, t)$, respectively.

In our proof of the simple case of Lemma 1, the advice can be viewed as a generic, one-round transcript. In the more general case, we seek an advice function consisting of a polynomial-sized *set* of transcripts that allows us to compute $f(x)$ in nondeterministic polynomial time. As in the simple case, the nondeterministic step consists of guessing a key in $\kappa(x, t)$, where x is the cleartext instance and t is the transcript in the advice function; for GES's, we cannot get by with just one transcript, because there may be no t for which all of the sets $\kappa_g(x, t)$ have an element in common. Furthermore, GES's are allowed to compute wrong values for $f(x)$ a certain fraction of the time, but, in order to show that f is in NP/*poly* $\cap$ CoNP/*poly*, we must construct an advice function that allows us to compute $f(x)$ correctly all of the time. This is accomplished by including, for each transcript that is part of the advice, a universal hash function that allows us to distinguish good keys from bad.

Consider the GES G for f that leaks at most n. In order to demonstrate the usefulness of a transcript t in an advice function for f, we would like expressions for quantities such as the probability that G computes the wrong value for $f(x)$, given that $X = x$ and $T_n = t$. The key-sampling algorithm $k()$ in G is required to produce a valid key in expected polynomial time. We can view $k()$ as a deterministic algorithm that has access to an infinite tape of unbiased random bits. It is useful to consider what happens when we limit the number of random bits available to the GES to some fixed polynomial number, say $r(n)$. Each polynomial r gives rise to a different *truncated GES* G'_r corresponding to G. Keys in G'_r are just unbiased coin-toss sequences. Let $k'_r()$ be the key-sampling algorithm that, on input x, returns a key that is a sequence of $r(|x|)$ unbiased random bits. Thus, for each possible cleartext instance x, $k'_r()$ produces each legal key with equal probability. G'_r works as follows: for cleartext instance x, $k'_r()$ generates a key s of $r(|x|)$ fair coin tosses. G'_r then evokes $k()$, the key-sampling algorithm of the original scheme G, using s as the tape of random bits. Once $k()$ produces a key, G'_r continues to simulate G, computing A's guess for $f(x)$. Thus, in the cases in which G'_r does produce a transcript of a conversation between A and B, it is the same one that G would have produced. Furthermore, G'_r leaks at most $|x|$ if G leaks at most $|x|$.

Of course, there may be runs on which $k()$ uses more than $r(n)$ random bits, and, on these runs, G'_r produces no transcript at all. When this happens, we say that G'_r *hangs*. More precisely, we use the following definitions. Let $\kappa'_r(x, t)$, $\kappa'_{g,r}(x, t)$ and $\kappa'_{b,r}(x, t)$ denote the various key sets for G'_r.

Definition: $hang_r(x, t) = P(k()$ uses more than $r(n)$ coin tosses $\mid X = x \ \wedge \ T_n = t)$.

Definition: $Hang_r(x) = E(hang_r(x, T_n))$.

In other words, $Hang_r(x)$ is the probability that G'_r hangs on cleartext instance x, if we limit the number of available random bits to $r(|x|)$. The expectation in the definition is computed over the distribution $P(T_n = t)$.

We now quantify the effects on the usefulness of a transcript of allowing the GES to make errors and truncating the random tape.

Definition: $\mathit{fail}(x,t) = P(G \text{ computes } f(x) \text{ incorrectly} \mid X = x \ \wedge \ T_n = t)$.

Definition: $\mathit{Fail}(x) = E(\mathit{fail}(x, T_n))$.

Definition: $\mathit{fallibility}_r(x,t) = P(G \text{ computes } f(x) \text{ incorrectly} \mid X = x \ \wedge \ T_n = t \ \wedge \ k()$ uses at most $r(n)$ random bits). Note that this is also equal to $P(G'_r$ computes $f(x)$ incorrectly $\mid X = x \ \wedge \ T_n = t)$.

The failure rate $\mathit{fallibility}_r(x,t)$ of G'_r is greatest if G always computes $f(x)$ correctly when $k()$ uses more than $r(n)$ random bits, given $X = x$ and $T_n = t$. Using this observation and the definition of conditional probability, we obtain the upper bound

$$\mathit{fallibility}_r(x,t) \leq \frac{\mathit{fail}(x,t)}{1 - \mathit{hang}_r(x,t)}. \quad (1)$$

Definition: Transcript t is *reliable for* x, with respect to the polynomial r, if $\mathit{fallibility}_r(x,t) < 2^{-n}$ and $\mathit{hang}_r(x,t) < .1$.

Here 2^{-n} and .1 may clearly be replaced by other fixed quantities that are "exponentially small" and "near zero," respectively. We seek reliable transcripts for our advice function. Inequality (1) shows that, in order to prove that t is reliable for x, with respect to r, it suffices to show that

$$\mathit{fail}(x,t) < .9 \cdot 2^{-n} \quad \text{and} \quad \mathit{hang}_r(x,t) < .1. \quad (2)$$

Note that truncated GES's can indeed be used to derive the kinds of expressions we seek. They have the convenient property that the probability that t occurs for cleartext instance x is directly proportional to the size of $\kappa'_r(x,t)$. Similarly, the probability that G'_r computes $f(x)$ incorrectly given that t occurs is simply the ratio $|\kappa'_{b,r}(x,t)|/|\kappa'_r(x,t)|$. Thus we have the following relationships. (Recall that G'_r leaks at most $|x|$.)

$$\frac{P(T_n = t \mid X = x_1)}{P(T_n = t \mid X = x_2)} = \frac{|\kappa'_r(x_1,t)|}{|\kappa'_r(x_2,t)|} = \frac{1 - \mathit{hang}_r(x_1,t)}{1 - \mathit{hang}_r(x_2,t)}, \quad (3)$$

where $|x_1| = |x_2|$, and

$$\mathit{fallibility}_r(x,t) = \frac{|\kappa'_{b,r}(x,t)|}{|\kappa'_r(x,t)|}. \quad (4)$$

Finally, we show that, for any fixed positive constant c_1, if G is a GES for f, then there exist another GES $\overline{G}$ for f and a polynomial r for which

$$\mathit{Fail}(x) < 2^{-c_1 n} \quad \text{and} \quad \mathit{Hang}_r(x) < n^{-c_1}. \quad (5)$$

Recall that, by definition of a GES, there is a positive constant c such that $\mathit{Fail}(x) < 1/2 - n^{-c}$ (in G). Certainly, it suffices to prove (5) for all $c_1 > c$. Also by definition, there is a polynomial *poly* such that the expected number of random bits used by $k()$ for any x is at most $\mathit{poly}(n)$. Let $s(n) = n^{c_1} \cdot \mathit{poly}(n)$. In the resulting truncated GES, $\mathit{Hang}_s(x)$ is the

probability that $S(x) > s(n)$, where $S(x)$ is a random variable representing the number of random bits used by $k()$ on cleartext instance x. The probability that $S(x) > s(n)$ is bounded above by $E(S(x))/s(n)$ ([HPS, page 101]). The expected value $E(S(x))$ is just $poly(n)$, and so this upper bound on $Hang_s(x)$ is just n^{-c_1}. Therefore, if G is allowed $s(n)$ random bits, the probability that it either hangs or returns the wrong answer is less than $1/2 - n^{-c} + n^{-c_1}$, which is polynomially bounded below $1/2$, because $c_1 > c$. Thus, there is another polynomial, say $u(n)$, such that, if we perform $u(n)$ runs of G'_s on any input x of length n and take as our guess at $f(x)$ the answer returned on a majority of those runs that do not hang, the probability that this guess is wrong is less than $2^{-c_1 n}$.

Let $r(n) = s(n) \cdot u(n)$. Here is a truncated GES $\overline{G}'_r$ for f that satisfies (5). On input x of length n, the key-sampling algorithm $\overline{k}'_r()$ for $\overline{G}'_r$ performs $u(n)$ simulations of $k'_s(x)$, where $k()$ is the key-sampling for G. Let k_1 through $k_{u(n)}$ be the keys obtained; if G'_s hangs on the i^{th} simulation, then $\overline{k}'_r()$ uses a special "blank" symbol for k_i. For each i such that k_i is not blank, $\overline{G}'_r$ simulates the interaction between A and B that occurs in G and obtains a list l_i of B's responses. After all interaction is over, $\overline{G}'_r$ performs (at most) $u(n)$ simulations of G's decryption algorithm on inputs (x, k_i, l_i), gets a set of guesses for $f(x)$, and takes the majority. The protocol $\overline{G}$ obtained by using an infinite tape of random bits and simulating G rather than G'_r obviously satisfies the definition of a GES. It satisfies $Fail(x) < 2^{-c_1 n}$ by the argument in the preceding paragraph. It also satisfies $Hang_r(x) < n^{-c_1}$, because the probability that $\overline{G}'_r$ hangs is the probability that *all* of the simulations G'_s hang, and each simulation hangs with probability less than n^{-c_1}, by choice of s. Thus, $\overline{G}$ is a GES for f that satisfies (5).

A GES that satisfies (5) is said to be *well-behaved with respect to* r. In Lemmas 1.1 and 1.2 below, there are implicit constants c_1 for which we will need the fact that, if f has one GES, then it has another that is well-behaved with respect to r, for an appropriate polynomial r. In order to avoid obscuring the main points of the lemmas by making these quantities c_1 and r explicit, we simply "assume without loss of generality that G is well-behaved." By that, we always mean that we "assume without loss of generality that G is well-behaved with respect to r, where r is derived from the appropriate constant c_1." We also drop the r from the symbols $hang_r$, $Hang_r$, $fallibility_r$, G'_r, $k'_r()$, κ', $\kappa'_{g,r}$, and $\kappa'_{b,r}$, and from the term "reliable with respect to r." The missing r is always the one derived from the implicit c_1.

We are now prepared to prove the existence of useful sets of reliable transcripts.

Lemma 1.1: Suppose that f has a GES G that leaks at most n. Assume, without loss of generality, that G is well-behaved. Then, there is a positive constant c_2 such that, for all sufficiently large n, there is a set S_n of n^{c_2} transcripts the majority of which are reliable for all x of size n.

Proof: We show first that, for any x of size n, if we pick a transcript t at random, according to the distribution on T_n, then the probability that it is reliable for x is bounded above $1/2$. Because $Hang(x) = E(hang(x, T_n)) < .001$ for sufficiently large n, the probability that $hang(x, t) \geq .1$ cannot be greater than .01. Likewise, since $Fail(x) = E(fail(x, T_n)) < 2^{-2n}$ for sufficiently large n, the probability that $fail(x, t) \geq .9 \cdot 2^{-n}$ must also be less than .01.

(Both of these facts follow from the inequality we used in the discussion of well-behaved GES's.) Thus, the probability that either $hang(x,t) \geq .1$ or $fail(x,t) \geq .9 \cdot 2^{-n}$ is no greater than .02. By (2), above, this means that the probability that a random transcript is reliable for x is at least .98, which is certainly bounded above 1/2.

Consider a set S of k transcripts selected independently according to the distribution on T_n. We can apply the Chernoff bound for the tail of a binomial distribution to bound above by $2^{O(-k)}$ the probability that at most half of these transcripts are reliable for x ([C]). By setting k equal to n^{c_2}, for a fixed constant c_2, we can strictly bound this probability above by 2^{-n}. There are only 2^n possible values of x, and thus the probability that at most half of the transcripts in a randomly selected S are reliable for all x is strictly bounded above by $2^n \cdot 2^{-n} = 1$. Hence, at least one set S_n that satisfies the conditions of the lemma must exist. ∎

The rest of the proof of Lemma 1 consists of showing how to use universal hash functions to guarantee that certain sets contain only good keys and how to use them as advice in a nondeterministic polynomial-time algorithm to compute $f(x)$.

Let $H_{p,q}$ be a set of p-universal hash functions that map G' keys to $\{1, 2, \ldots, q\}$. For concreteness, we take $H_{p,q}$ to be the following set of functions. Let c_3 be a constant such that $q^{c_3} > 2^{r(n)}$, the number of G' keys for cleartext instances of size n. Consider the set of polynomials of degree $p-1$ with coefficients in the finite field $\mathrm{GF}(q^{c_3})$; elements of this field are in turn represented as polynomials over Z_q. Each degree-$(p-1)$ polynomial *poly* over $\mathrm{GF}(q^{c_3})$ corresponds to a hash function h in $H_{p,q}$. To compute $h(k)$, where k is a G' key, we first plug k into *poly*; this gives us an element of $\mathrm{GF}(q^{c_3})$, represented uniquely as a polynomial *poly'* over Z_q. We take the constant term of *poly'* to be $h(k)$. Note that $|Range(h)| = q$ and that each such h can be specified using an amount of space that is polynomial in p and $|q|$. The property of $H_{p,q}$ that we use is this: for any set $\{k_1, k_2, \ldots, k_p\}$ of distinct keys, if $h \in H_{p,q}$ is chosen uniformly at random, then the probability that

$$h(k_1) = h(k_2) = \cdots = h(k_p)$$

is $q^{-(p-1)}$. We call a set $\{k_1, \ldots, k_p\}$ for which the values $h(k_i)$ are all the same an (h,p)-*committee*.

In the following lemma, t is a transcript, and I_t is the set of cleartext instances of size n for which t is reliable. As in Lemma 1.1, we assume without loss of generality that G is well-behaved.

Lemma 1.2: There is a constant d such that, for all t, there exist an integer p and a prime q, where $p \leq n^d$ and $|q| \leq n^d$, and a function h in $H_{p,q}$ for which these conditions hold for all x in I_t:

(A) $\kappa_g{}'(x,t)$ contains an (h,p)-committee, and

(B) $\kappa_b{}'(x,t)$ does not contain an (h,p)-committee.

Proof: Condition (A) holds for any h in $H_{p,q}$ provided that $|\kappa_g{}'(x,t)| > pq$. This follows from the pigeonhole principle: if h maps more than pq elements to q values, there must be at least p elements that are mapped to the same place. We proceed to show that we can

find p and q of the appropriate size so that a function h chosen at random from $H_{p,q}$ also satisfies (B) with positive probability.

Let

$$U = \max_{x \in I_t} |\kappa'(x,t)|$$

and

$$V = \max_{x \in I_t} |\kappa'_b(x,t)|.$$

Because $hang(x,t) < .1$, equation (3) implies that

$$\min_{x \in I_t} |\kappa'(x,t)| \geq .9 \cdot U.$$

There are at most 2^n cleartext instances x in I_t; for each one of them, there are at most $\binom{V}{p}$ ways for form a committee of bad G' keys. Hence, there are at most

$$2^n \cdot \binom{V}{p} < 2^n \cdot \left(\frac{eV}{p}\right)^p$$

potential committees of bad keys. (The inequality follows from Stirling's approximation for the factorial function ([K, page 46]).) The probability that a randomly selected element of $H_{p,q}$ maps all the elements of one of these potential committees to the same destination is $q^{-(p-1)}$. Thus the probability that a random h has any committees made up entirely of bad keys is strictly bounded above by

$$2^n \cdot q \cdot \left(\frac{eV}{pq}\right)^p.$$

We must find a constant d and integers p and q, satisfying the conditions of the lemma and the conditions imposed thus far in the proof, for which this probability is less than 1.

The size of the G' keys is bounded by n^{d_1}, for some constant d_1 that depends on the GES, but not on n. Thus, $V < 2^{n^{d_1}}$. Let q be a prime such that $eV < q < 2eV$; there must be a prime in this interval by a theorem of Chebyshev ([NZ, page 185]). Thus $|q| < n^d$, where $d = 2d_1$. We must choose p such that

$$2^n \cdot q \cdot \left(\frac{eV}{pq}\right)^p < 2^n \cdot (2eV) \cdot p^{-p} < 1.$$

Our bound on V implies that this is equivalent to

$$2^{n+1+\log_2 e + n^{d_1} - p \log p} < 1,$$

and thus it suffices to choose $p = n^d$.

The last thing to check is that $|\kappa'_g(x,t)| > pq$. To obtain this lower bound, we may assume that x has the maximum number of bad keys, i.e., that

$$|\kappa'_g(x,t)| \geq \min_{x \in I_t} |\kappa'(x,t)| - \max_{x \in I_t} |\kappa'_b(x,t)| \geq .9 \cdot U - V.$$

Under the same assumption, equation (4) and the definition of reliable give the bound

$$2^n \cdot V \leq U.$$

Thus it suffices to check that pq is strictly less than $(.9 \cdot 2^n - 1) \cdot V$. Our construction gives the strict upper bound $pq < (n^d \cdot 2e) \cdot V$, which is clearly stronger for large n. ∎

Let t be a transcript in the set S_n of Lemma 1.1, C be an (h, p)-committee, where h, p, and q are the values given by Lemma 1.2, and x be any instance of size n. By Lemma 1.2, we know that either all of the keys in C are good for (x, t), in which case we say that C is a *good committee for* x, or some of the keys in C are good for (x, t) and some are bad. It cannot be the case that all of the keys in C are bad. Thus, if $C \subseteq \kappa'(x, t)$ and the values $A(x, k, t)$ are the same for all k in C, then C is a good committee for x. In this case, we can use any k in C to compute $f(x) = A(x, k, t)$.

We now complete the proof of Lemma 1. We show how to use

$$n \mapsto \{\langle t_i, p_i, q_i, h_i \rangle, 1 \leq i \leq l\},$$

where $l = |S_n|$, as advice to find $f(x)$ in nondeterministic polynomial time. Guess a subset M of S_n, where $|M| \geq \lceil |S|/2 \rceil$; the transcripts in M are the ones that are reliable for x. For each t_i in M, guess an (h_i, p_i)-committee of G' keys $C = \{k_{i1}, \ldots, k_{ip_i}\}$ that is good for x. Verify that C forms an (h_i, p_i)-committee by checking that the values $h_i(k_{ij})$, for $1 \leq j \leq p_i$, are all the same. Verify that C is a good committee for x by checking that $C \subseteq \kappa'(x, t_i)$ and that the p_i values $A(x, k_{ij}, t_i)$ are all the same. Finally, verify that the answers $A(x, k_{i1}, t_i)$ are all the same, for t_i in M. We can conclude that this value is $f(x)$.

Note that this nonuniform procedure for computing $f(x)$ works both when x is a yes-instance and when it is a no-instance. Thus, if f is the characteristic function for S, we have shown that S is in NP/*poly* and that $\overline{S}$ is in NP/*poly*. The latter is equivalent to the statement that S is in CoNP/*poly* ([Yap, Lemma 6]). ∎

Proof of Lemma 2: The crux of this proof is that, if the satisfiability of any formula of length n can be determined in CoNP with advice $h(n)$, then the membership of x in any NP set T can be determined in CoNP with advice $\{h(i), 1 \leq i \leq l\}$, where l is polynomial in $|x|$.

More formally: the hypothesis is that SAT $= S : h = \{x\colon h(|x|) \cdot x \in S\}$, for some polynomial advice function h and some $S \in$ CoNP. Let T be a set in NP and let r be a polynomial-time, many-to-one reduction from T to SAT. That is, $x \in T$ if and only if $r(x) \in$ SAT. Because r is polynomial-time computable, there is a fixed polynomial p for which $|r(x)| \leq p(|x|)$.

Thus

$$T = \{x\colon h(|r(x)|) \cdot r(x) \in S\},$$

where

$$S = \{y\colon \forall c\ G(c, y)\},$$

for some polynomial-time predicate G. The following function h', whose value at n is the concatenation of the values $h(i)$, $1 \leq i \leq n$, is a polynomial advice function for T:

$$h'(n) = h(1)\$h(2)\$\cdots\$h(p(n)).$$

(The character \$ is a distinguished delimiter.)

Consider

$$S' = \{y\colon (y = \alpha_1\$\alpha_2\$\cdots\$\alpha_{p(|x|)} \cdot x) \ \wedge \ (\alpha_{|r(x)|} \cdot r(x) \in S)\}.$$

We have

$$S' = \{y\colon \forall c\ [(y = \alpha_1\$\alpha_2\$\cdots\$\alpha_{p(|x|)} \cdot x) \ \wedge \ G(c, \alpha_{|r(x)|} \cdot r(x))]\} \in \text{CoNP}.$$

Thus

$$\begin{aligned} T &= \{x\colon h(|r(x)|) \cdot r(x) \in S\} \\ &= \{x\colon h'(|x|) \cdot x \in S'\} = S' : h' \end{aligned}$$

is in CoNP/*poly*, as was to be shown. ∎

We now consider the possibility of a GES for SAT that leaks at most some function L other than $|x|$. Following, e.g., [BLS], we use NPSV to denote the set of partial, single-valued functions computable by nondeterministic, polynomial-time transducers. A function f is in the nonuniform class NPSV/*poly* if there is a polynomial advice function h and a function g in NPSV such that $f(x) = g(h(|x|) \cdot x)$, for all x.

Corollary 1: In the hypotheses of Lemma 1 and Theorem 3, the function $|x|$ can be replaced by any function $L(x)$ such that $Dom(L) = \{0,1\}^*$, $L \in$ NPSV/*poly*, and $|L(x)| = O(\log |x|)$.

Proof: The cornerstone of the argument in Lemma 1 is that, if f has an encryption scheme that leaks at most $|x|$, then all cleartext instances of size $|x|$ are indistinguishable from B's point of view. If f has a scheme that leaks at most $L(x)$, where $|L(x)| = O(\log |x|)$, then the function L can assume at most $|x|^c$ values, for some constant c, on cleartext instances of size $|x|$; thus the cleartext instances of size $|x|$ can be partitioned into $|x|^c$ classes, where the instances in each class are indistinguishable from B's point of view. By hypothesis, we can use polynomial advice to determine $L(x)$, and thus which of these classes contains x, in nondeterministic polynomial time. We can use advice of the form $\{\langle t_i,\ p_i,\ q_i,\ h_i\rangle, 1 \leq i \leq l\}$ for each of the classes to determine $f(x)$; since there are only polynomially many classes, the total amount of advice for instances of size $|x|$ is polynomial in $|x|$, and f is still in NP/*poly* ∩ CoNP/*poly*. ∎

Corollary 2: If $\overline{\text{SAT}}$ has a GES that leaks at most $L(x)$, such that $Dom(L) = \{0,1\}^*$, $L \in$ NPSV/*poly*, and $|L(x)| = O(\log |x|)$, then the PH collapses at the third level.

Proof: By Corollary 1, this hypothesis implies that $\overline{\text{SAT}}$ is in NP/*poly* ∩ CoNP/*poly*. If $\overline{\text{SAT}}$ is in NP/*poly*, then all of CoNP is contained in NP/*poly*, by a straightforward proof like that of Lemma 2. If CoNP ⊆ NP/*poly*, then the PH collapses at the third level, also by [Yap, Theorem 2]. ∎

We also consider *one-sided encryption schemes*, that is, schemes that behave as in Section 2 when the cleartext instance x is a yes-instance but may leak more than $L(x)$ when x is a no-instance.

Corollary 3: If $\overline{\text{SAT}}$ is one-sided encryptable leaking $L(x)$, where $Dom(L) = \{0,1\}^*$, $L \in$ NPSV/*poly* and $|L(x)| = O(\log |x|)$, then the PH collapses at the third level.

Proof: Suppose that $\overline{\text{SAT}}$ has a one-sided GES that leaks at most $L(x)$. We proceed, as in Lemma 1, using transcripts, hash functions, and nondeterministically chosen keys, to construct a nonuniform algorithm for recognizing $\overline{\text{SAT}}$. The algorithm uses only polynomial-length advice strings because $|L(x)| = O(\log |x|)$. Since this is a *one-sided* encryption scheme, it is guaranteed to leak at most $L(x)$ only on yes-instances (i.e., on unsatisfiable formulas); hence the nonuniform algorithm is guaranteed to work only on yes-instances (unlike the one in Lemma 1). This is sufficient to prove that $\overline{\text{SAT}} \in$ NP/*poly*, which in turn implies that PH $= \Sigma_3^{\text{P}} = \Pi_3^{\text{P}}$. ∎

In the hypotheses of Lemma 1, Theorem 3, and Corollaries 2 and 3, SAT and $\overline{\text{SAT}}$ can be replaced by any total functions (not necessarily boolean) that are NP-Hard and CoNP-Hard, respectively.[2] The proof of Corollary 3 would not go through if $\overline{\text{SAT}}$ were replaced with SAT; this is because the hypothesis that SAT is one-sided encryptable leaking $O(\log |x|)$ bits implies only that NP $\subseteq$ NP/*poly*, which is trivial.

5. Connection to random-self-reducibility.

A random-self-reduction is essentially an expected polynomial-time reduction from a set to itself that maps an element of size n to each other element of size n with equal probability. This notion has arisen in the study of cryptography and in the study of zero-knowledge proofs ([AL, TW]). Random-self-reducible sets are as hard on average as they are in the worst case. Angluin and Lichtenstein pointed out that this is a possible reason that these sets give rise to good candidates for one-way functions. Here, we define the notion precisely and show how it relates to encryptability.

Let S be a set. It is convenient to encode elements of S as pairs $\langle v,\ w\rangle$. A random-self-reduction t is a mapping from S to S and $\overline{S}$ to $\overline{S}$ that makes use of an appropriate set $\mathcal{K}$ of keys. On input x, t chooses a key k and then computes $t(x,k)$, another element of S. The key chosen is of length polynomial in $|x|$, the choice of k takes expected time polynomial in $|x|$, and the computation of $t(x,k)$ takes worst-case polynomial time. Let $\mathcal{K}_x$ be the set of keys that may be chosen on input x. In order to be a random-self-reduction, t must have the following additional properties.

(1) For each x and each $k \in \mathcal{K}_x$, $|t(x,k)| = |x|$.

(2) For each $x = \langle v,\ w\rangle$ and each $k \in \mathcal{K}_x$, if $t(\langle v,\ w\rangle, k) = \langle v',\ w'\rangle$, then $v' = v$.

[2] An example of an NP-Hard function that is not boolean is #SAT, where #SAT(x) is the number of satisfying assignments for the formula x. See [GJ, Chapter 5] for a formal definition of NP-Hardness of general functions.

(3) For each $x = \langle v,\ w\rangle$, t outputs each $\langle v,\ w'\rangle$ in S of length $|x|$ with equal probability.

A random-self-reduction is a one-sided encryption function for the characteristic function f for S that leaks $\langle v,\ |x|,\ f(x)\rangle$. The decryption algorithm is trivial.

We call t a p-random-self-reduction if p is the fraction of the input that t fixes – in other words, if $p = \frac{|v|}{|\langle v,\ w\rangle|}$. In general, p will be a function of $|\langle v,\ w\rangle|$. The smaller p is, the more randomization is achieved; so if S is 0-random-self-reducible, there is no fixed part v, and any element of size n can be mapped uniformly to any other.

The well-known examples of random-self-reducible sets are the number-theoretically defined sets that are ubiquitous in cryptographic applications. Indeed the encryption functions in Section 3 are random-self-reductions. To make this observation rigorous, we must express the QRP, say, as a set-recognition problem, which we do as follows.

Let S_{QRP} be the set of pairs $\langle u,\ n\rangle$ in which $n = pq$, where p and q are distinct primes congruent to 3 mod 4 and u is a quadratic residue mod n. S_{QRP} is clearly in NP, because the following algorithm determines membership of $\langle u,\ n\rangle$: guess the factorization of n, verify that both factors are prime, using Pratt's algorithm ([Pra]), guess a square root a of u, and verify that $a^2 \equiv u \bmod n$. Similarly, $\overline{S_{\mathrm{QRP}}}$ is in NP: given a pair $\langle u,\ n\rangle$, we can show in nondeterministic polynomial time either that the real prime factorization of n has the wrong form, that u is not relatively prime to n, or that u is of the form $(-1)\cdot a^2$ for some a. The functions $k()$ and E in the encryption scheme given for QRP in Section 3 together form a $1/2$-random-self-reduction for S_{QRP}. The sets associated with the problems DLP and PRP from Section 3 are also $\Omega(1)$-random-self-reducible.

It is natural to ask whether there is an NP-Complete set that is c-random-self-reducible for some constant c. This question makes sense only if the set is encoded succinctly, as the set S_{QRP} is in the discussion above. If we do not insist on a succinct encoding, then we can start with a standard encoding of SAT, pad every instance of length n with n bits of garbage, and achieve a $1/2$-random-self-reduction by fixing the meaningful part of the instance and randomizing the garbage. The folklore that suggests that one cannot base a one-way function on an NP-Complete set also suggests that NP-Complete sets are not c-random-self-reducible, because of the connection between one-way functions and random-self-reducibility drawn in [AL].

Along these lines, we can show that CoNP-Hard sets are not $O(\log n/n)$-random-self-reducible, unless the PH collapses at the third level; this follows from Corollary 3, because an $O(\log n/n)$-random-self-reduction is just a special case of a one-sided encryption function that leaks $O(\log n)$ bits.

In the same vein, it is easy to show directly that if $\overline{\mathrm{SAT}}$ is 0-random-self-reducible, then NP = CoNP. Suppose that t is a 0-random-self-reduction for $\overline{\mathrm{SAT}}$. We can use it to build an NP algorithm for $\overline{\mathrm{SAT}}$. On input x, the algorithm first generates a string y of the form $\mathrm{FALSE} \wedge \mathrm{FALSE} \wedge \cdots \wedge \mathrm{FALSE}$, where $|y| = |x|$ (y is just a known $\overline{\mathrm{SAT}}$ instance of length $|x|$). Then the algorithm nondeterministically guesses a k and verifies that $t(x,k) = y$; if so, it accepts x. An appropriate k must exist, because t is a 0-random-self-reduction.

We can combine the results of [BHZ, For, TW] to prove a related result. Tompa and Woll have a stricter definition of random-self-reducibility that they use to prove that,

if S is random-self-reducible in their sense, then it has a *perfect* zero-knowledge proof system. If S has a perfect zero-knowledge proof system, then $\overline{S}$ is in AM[2] ([For]). If CoNP is contained in AM[2], then the PH collapses at the second level ([BHZ]). Thus, NP-Complete sets are also unlikely to be random-self-reducible, in the sense of Tompa and Woll. In a forthcoming paper, Nisan shows that NP-Hard sets are not $O(\log n/n)$-random-self-reducible (according to *our* definition), unless the PH collapses at the second level ([N]).

Finally, we would like to point out that there is a 1/2-random-self-reducible set in NP that is not known to be in CoNP. Let S be the set of pairs of isomorphic graphs $\langle G_1,\ G_2\rangle$. The mapping that leaves G_1 fixed and takes G_2 to the graph induced by a random permutation of $V(G_2)$ is a 1/2-random-self-reduction.

6. Open problems

Question 1: We have presented general results about the encryptability of NP-Hard functions and expected-polynomial-time computable functions, and we have presented examples of encryption schemes for functions that are believed to be neither NP-Hard nor expected-polynomial-time computable. However, we have no general results about such functions. Is FACTORING encryptable leaking at most $|x|$? Is FACTORING random-self-reducible?

Question 2: Suppose that SAT is encryptable leaking at most L, where $Dom(L) = \{0,1\}^*$, $L \in$ NPSV/*poly*, and $|L(x)| = O(\log\log|x|)$. Does the PH collapse below the third level?

Question 3: Suppose that SAT is encryptable leaking at most L, where $Dom(L) = \{0,1\}^*$, $L \in$ NPSV/*poly*, and $|L(x)| = |x|/2$. Does the PH collapse at the i^{th} level for some i? Note that this question is interesting only if instances are encoded succinctly.

Question 4: As we saw above, Graph Isomorphism is 1/2-random-self-reducible. Is it encryptable hiding at least the isomorphism classes of the input graphs?

Question 5: Can player A encrypt SAT leaking at most $|x|$ if she is allowed to query several oracles B_1, B_2, ..., B_n that cannot communicate with each other? Can she encrypt any functions that she cannot encrypt by querying only one oracle? (Recently answered in the affirmative: see reference [BF].)

7. Appendix A

In [Yao82, Yao86a], Yao announced a protocol by which A, who has a secret input i, and B, who has a secret input j, can compute the value $f(i,j)$, while neither reveals to the other anything about the secret inputs that is not implied by $f(i,j)$. In [Yao82, Yao86a], A needs the cooperation of B *because she does not know* j, rather than because she lacks the power to evaluate f. In Yao's model, the players A and B are equally powerful, and they have fairness as a goal, as well as validity and privacy. The proof that the protocol achieves these goals uses the Quadratic Residuosity Assumption (QRA), and hence the functions f that are considered are all efficiently computable.

The goal of a zero-knowledge proof system ([GHY89, GMR]) for the predicate f is to allow A to *prove* to B that $f(x) = \epsilon$, $\epsilon \in \{0,1\}$, without incidentally leaking more information about x than this single bit. There, A, the player who hides something, is at least as powerful as B, from whom she hides it. Often, A's advantage is based on the possession of trapdoor information. In contrast, the goal of an encryption function for f is to allow A to compute $f(x)$ with the help of B without revealing to B anything except $L(x)$. Here, A is the weaker player and must conceal something from someone with more resources. There is no element of proof in our framework as it stands.

For example, the goal of a zero-knowledge proof system for SAT would be for A to take a propositional formula F *that is known to both A and B* and convince B that she knows a model for F without telling him anything about what that model is. Such proof systems were exhibited independently in [BCC] and [GMW86]. The goal of an encryption scheme for SAT would be for A to transform F, *which is not known to B*, into another formula G such that A can deduce the satisfiability of F from that of G without revealing $L(F)$. We know that, if $\Sigma_3^{\mathrm{P}} \neq \Pi_3^{\mathrm{P}}$, then such an encryption scheme does not exist for any L such that $Dom(L) = \{0,1\}^*$, $L \in \mathrm{NPSV}/poly$ and $|L(F)| = O(\log |F|)$.

If f has both an encryption scheme and a zero-knowledge proof system, then the two can be combined in a protocol by which B can convince A that he has a proof that $f(x) = z$, say, without revealing any information about what the proof is or learning what x is. For example, there is such a protocol for the discrete logarithm function.

8. Appendix B

The following lemma was stated without proof in [AFK]. It extends a result in [KL]. Similar extensions can be found in, e.g., [BBS, Sch, Yap].

Lemma: If $\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{CoNP})/poly$, then the PH collapses at the second level, i.e., $\Sigma_2^P = \Pi_2^P$.

Proof: This is a purely technical extension of Karp and Lipton's proof that $\mathrm{NP} \subseteq \mathrm{P}/poly$ implies that $\Pi_2^{\mathrm{P}} = \Sigma_2^{\mathrm{P}}$ (Theorem 6.1 of [KL]). The two major steps in their proof, i.e., the inductive step and the final use of the recursive definition of the Π_3^{P}-Complete problem, can be made to work for sets $S \in \mathrm{NP} \cap \mathrm{CoNP}$ without assuming that $S \in \mathrm{P}$. For readers unfamiliar with the techniques of [KL], we give our strengthened version of the proof in

full, pointing out where they use the properties of P but could have made do with NP $\cap$ CoNP.

First we show by induction that

$$\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{CoNP})/poly \Longrightarrow \mathrm{PH} \subseteq (\mathrm{NP} \cap \mathrm{CoNP})/poly. \quad (6)$$

Let E_i be the set of encodings of true sentences of the form

$$Q_1 x_1 Q_2 x_2 \cdots Q_i x_i F(x_1, x_2, \ldots, x_i),$$

where $Q_1 = \exists$, the Q_j are alternately $\exists$ and $\forall$, x_j is a vector of l_j boolean variables, and F is a polynomial-time predicate. Let A_i be defined identically, except that the roles of $\exists$ and $\forall$ are reversed. It is known that E_i is Σ_i^{P}-complete and that A_i is Π_i^{P}-complete, under logspace reductions ([Sto]). Because $(\mathrm{NP} \cap \mathrm{CoNP})/poly$ is closed under complementation ([Yap]), $A_i \in (\mathrm{NP} \cap \mathrm{CoNP})/poly \Leftrightarrow E_i \in (\mathrm{NP} \cap \mathrm{CoNP})/poly$. (Here, Karp and Lipton use closure under complementation of $\mathrm{P}/poly$.) To prove (6), we show that $E_i \in (\mathrm{NP} \cap \mathrm{CoNP})/poly$, for all i.

By hypothesis, $E_1 \in (\mathrm{NP} \cap \mathrm{CoNP})/poly$. We proceed by induction on i. Assume that $E_{i-1} \in (\mathrm{NP} \cap \mathrm{CoNP})/poly$ (and hence that $A_{i-1} \in (\mathrm{NP} \cap \mathrm{CoNP})/poly$). Thus, there is a set $S \in \mathrm{NP} \cap \mathrm{CoNP}$, a constant k, and an advice function h such that $|h(n)| \leq n^k$ and $x \in A_{i-1} \Leftrightarrow h(|x|) \cdot x \in S$.

If y is the encoding of a sentence in E_i or A_i, and a is a vector of l_1 boolean variables, then we use y_a to denote the encoding of the sentence obtained from y by deleting the quantifier Q_1 and substituting a for x_1 in $F(x_1, x_2, \ldots, x_i)$. We can do the encoding and the substitution in such a way that $|y_a| = |y|$ for all y and a.

Let $T = \{wy\colon \exists a \text{ such that } wy_a \in S\}$. Because $S \in \mathrm{NP}$, there is a polynomial-time predicate F such that

$$S = \{x\colon \exists b\ F(b, x)\}.$$

Thus

$$T = \{wy\colon \exists a \exists b\ F(b, wy_a)\} \in \mathrm{NP}.$$

Here, Karp and Lipton used their assumption that $S \in \mathrm{P}$.

By inductive hypothesis,

$$\begin{aligned} y \in E_i &\Leftrightarrow \exists a \text{ such that } y_a \in A_{i-1} \\ &\Leftrightarrow \exists a \text{ such that } h(|y_a|) \cdot y_a \in S. \end{aligned}$$

But $|y_a| = |y|$. So

$$\exists a \text{ such that } h(|y_a|) \cdot y_a \in S \Leftrightarrow\ h(|y|) \cdot y \in T.$$

By the hypothesis that $\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{CoNP})/poly$, we have $T \in (\mathrm{NP} \cap \mathrm{CoNP})/poly$, and thus there is a set $S' \in \mathrm{NP} \cap \mathrm{CoNP}$, an integer k', and an advice function h' such that

$$|h'(n)| \leq n^{k'}$$

and

$$x \in T \Leftrightarrow h'(|x|) \cdot x \in S'.$$

Thus,

$$y \in E_i \Leftrightarrow \ h(|y|) \cdot y \in T \Leftrightarrow \ h'(|h(|y|) \cdot y|) \cdot h(|y|) \cdot y \in S'.$$

Now $h'(|h(|y|) \cdot y|) \cdot h(|y|)$ is a polynomial advice function whose value depends only on $|y|$, and $S' \in \mathrm{NP} \cap \mathrm{CoNP}$. This establishes that $E_i \in (\mathrm{NP} \cap \mathrm{CoNP})/poly$ and completes the proof of (6).

Next, we complete the proof of the lemma by showing the following:

$$\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{CoNP})/poly \Longrightarrow \Pi_3^{\mathrm{P}} \subseteq \Sigma_2^{\mathrm{P}}. \quad (7)$$

It suffices to show that the hypothesis implies that A_3 is in Σ_2^{P}. The proof is based on the following recursive definition of A_3. Let y be a sentence of the form

$$Q_1 x_1 Q_2 x_2 \cdots Q_n x_n F(x_1, x_2, \ldots, x_n),$$

where the string of quantifiers $Q_1 Q_2 \cdots Q_n$ is in $\forall^* \exists^* \forall^*$ (note that these boolean variables x_1, ..., x_n are not vectors as above). Let

$$y' = Q_2 x_2 \cdots Q_n x_n F(0, x_2, \ldots, x_n)$$

and

$$y'' = Q_2 x_2 \cdots Q_n x_n F(1, x_2, \ldots, x_n).$$

Then the characteristic function C_{A_3} for A_3 satisfies

$$C_{A_3}(y) = [\text{if } Q_1 = \forall \text{ then } C_{A_3}(y') \wedge C_{A_3}(y'') \text{ else } C_{A_3}(y') \vee C_{A_3}(y'')].$$

The function C_{A_3} is uniquely determined by its values on the set of quantifier-free sentences and by this recursive definition, which has the form

$$C_{A_3}(y) = R(y, C_{A_3}(y'), C_{A_3}(y'')).$$

By (6), $A_3 \in (\mathrm{NP} \cap \mathrm{CoNP})/poly$. Thus $A_3 = S : h$, where $S \in \mathrm{NP} \cap \mathrm{CoNP}$ and $|h(n)| \leq n^k$. For each $w \in \{0,1\}^*$, define $f_w : \{0,1\}^* \longrightarrow \{0,1\}$ by $f_w(x) = 1 \Leftrightarrow wx \in S$. Then, for each y that contains at least one quantifier, membership of y in A_3 is expressed by the formula:

$$\exists w \forall z \ [f_w(y) = 1 \wedge f_w(z) = R(z, f_w(z'), f_w(z''))]. \quad (8)$$

Here w ranges over all strings of length at most $|y|^k$, and z ranges over all strings of length $|y|$. We interpret (8) as follows: y is in A_3 if and only if there is some w such that $f_w(y) = 1$. This particular f_w is really the characteristic function for A_3 if and only if, for all z, $f_w(z) = R(z, f_w(z'), f_w(z''))$. Karp and Lipton were able to finish the proof by saying that

$$V = \{\{y, z\} \colon f_w(y) = 1 \wedge f_w(z) = R(z, f_w(z'), f_w(z''))\}$$

is clearly in P, because S is in P. Here $S \in \mathrm{NP} \cap \mathrm{CoNP}$, and we wish to show that V is in CoNP. Once we have shown that, we will have (8) in the form

$$\exists w \forall z \forall b \; G(w, z, b, y), \quad (9)$$

for some polynomial-time predicate G, and we will be able to merge the two universal quantifiers in (9) to have A_3 in the form $\{y\colon \exists w \forall c \; G(w, c, y)\}$. This will mean that $A_3 \in \Sigma_2^{\mathrm{P}}$, which in turn will imply (7) and complete the proof of the lemma.

The remainder of the proof is highly technical and the formulas may be somewhat intimidating. Essentially, the fact we need is that if two formulas can both be checked by NP machines and by CoNP machines, then their equivalence can also be checked by an NP machine and by a CoNP machine.

Because $S \in \mathrm{CoNP}$, $\{y\colon f_w(y) = 1\}$ has the form $\{y\colon \forall c_1 \; G_1(c_1, wy)\}$, for some polynomial-time predicate G_1. Next consider

$$U = \{z\colon f_w(z) = R(z, f_w(z'), f_w(z''))\}.$$

If we expand the definitions of f and R, we can write this in the form

$$\begin{aligned} U = \{z\colon & [wz \in S \wedge [\text{if } Q_1(z) = \forall \text{ then } [wz' \in S \wedge wz'' \in S] \text{ else } [wz' \in S \vee wz'' \in S]] \vee \\ & [wz \notin S \wedge [\text{if } Q_1(z) = \forall \text{ then } [wz' \notin S \wedge wz'' \notin S] \text{ else } [wz' \notin S \vee wz'' \notin S]]]\}. \end{aligned}$$

Because $S \in \mathrm{NP}$, $\overline{S} \in \mathrm{CoNP}$, and hence $\overline{S}$ has the form $\{y\colon \forall c_2 \; G_2(c_2, y)\}$ for some polynomial-time predicate G_2. Thus, V is of the form (9) and is hence in CoNP. The universally quantified variable b is the vector $c_1 c_2$ and the polynomial-time predicate G is

$$\begin{aligned} G(w, z, b, y) = {} & G_1(c_1, y) \wedge \\ & [[G_1(c_1, wz) \;\wedge\; [\text{if } Q_1(z) = \forall \text{ then } [G_1(c_1, wz') \wedge G_1(c_1, wz'')] \\ & \qquad \text{else } [G_1(c_1, wz') \vee G_1(c_1, wz'')]]] \vee \\ & [G_2(c_2, wz) \;\wedge\; [\text{if } Q_1(z) = \forall \text{ then } [G_2(c_2, wz') \wedge G_2(c_2, wz'')] \\ & \qquad \text{else } [G_2(c_2, wz') \vee G_2(c_2, wz'')]]]. \end{aligned}$$

The fact that $A_3 \in \mathrm{CoNP}$ implies that $\Sigma_3^{\mathrm{P}} \subseteq \Pi_2^{\mathrm{P}}$. This in turn implies that $\Sigma_2^{\mathrm{P}} \subseteq \Pi_2^{\mathrm{P}}$, and, by symmetry, that $\Sigma_2^{\mathrm{P}} = \Pi_2^{\mathrm{P}}$. ∎

9. Acknowledgements

We are deeply indebted to an anonymous referee, who pointed out numerous ways in which a previous version of this paper could be corrected and improved. We are also grateful to Manuel Blum, David Eppstein, Stuart Haber, Dexter Kozen, Steve Vavasis, Avi Wigderson, Moti Yung, and Stathis Zachos for fruitful discussions of these ideas. We thank Juris Hartmanis for pointing out Open Questions 2 and 3 and Ron Rivest for suggesting Open Question 5.

10. References

[**AF**] Abadi, Martín and Joan Feigenbaum. "A Simple Protocol for Secure Circuit Evaluation," *Proceedings of the* 5th *STACS*, Springer Verlag, 1988, 264–272.

[**AFK**] Abadi, Martín, Joan Feigenbaum, and Joe Kilian. "On Hiding Information from an Oracle," *Proceedings of the* 19th *STOC*, ACM, 1987, 195–203.

[**AL**] Angluin, Dana and David Lichtenstein. "Provable Security of Cryptosystems: a Survey," YALEU/DCS/TR-288, 1983.

[**BCC**] Brassard, Gilles, David Chaum and Claude Crépeau. "Minimum Disclosure Proofs of Knowledge," *JCSS*, 37, 1988, 156–189.

[**BF**] Beaver, Donald and Joan Feigenbaum. "Encrypted Queries to Multiple Oracles," AT&T Bell Laboratories Technical Memorandum, August 14, 1989.

[**BHZ**] Boppana, Ravi, Johan Hastad, and Stathis Zachos. "Does CoNP Have Short Interactive Proofs?," *IPL*, 25, 1987, 127–132.

[**BLS**] Book, Ronald, Timothy Long, and Alan Selman. "Quantitative Relativizations of Complexity Classes," *SIAM J. on Comput.*, 13, 1984, 461–487.

[**BM**] Blakley, G. R. and Catherine Meadows. "A Database Encryption Scheme which Allows the Computation of Statistics Using Encrypted Data," *Proceedings of the Symposium on Security and Privacy*, IEEE, 1985, 116-122.

[**C**] Chernoff, H. "A Measure of Asymptotic Efficiency for Tests of Hypothesis Based on the Sum of the Observations," *Ann. of Math. Stat.*, 23, 1952, 493–507.

[**CDvdG**] Chaum, David, Ivan Damgard, and Jeroen van de Graaf. "Multiparty Computations Ensuring Secrecy of Each Party's Input and Correctness of the Output," *Proceedings of CRYPTO '87*, Springer-Verlag, 1988, 87–119.

[**CW**] Carter, Larry and Mark Wegman. "Universal Classes of Hash Functions," *JCSS*, 18, 1979, 143–154.

[**Fei**] Feigenbaum, Joan. "Encrypting Problem Instances, or, ..., Can You Take Advantage of Someone Without Having to Trust Him?," *Proceedings of CRYPTO '85*, Springer-Verlag, 1986, 477–488.

[**For**] Fortnow, Lance. "The Complexity of Perfect Zero-Knowledge," *Proceedings of the* 19th *STOC*, ACM, 1987, 204–209.

[**GHY89**] Galil, Zvi, Stuart Haber, and Moti Yung. "Minimum-Knowledge Interactive Proofs for Decision Problems," *SIAM J. on Comput.*, 18, 1989, 711–739.

[**GHY87**] Galil, Zvi, Stuart Haber, and Moti Yung. "Cryptographic Computation: Secure Fault-Tolerant Protocols in the Public-Key Model," *Proceedings of CRYPTO '87*, Springer-Verlag, 1988, 135–155.

[**GJ**] Garey, Michael and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[**GMR**] Goldwasser, Shafi, Silvio Micali, and Charles Rackoff. "The Knowledge Complex-

ity of Interactive Proof-Systems," *SIAM J. on Comput.*, 18, 1989, 186–208.

[GMW86] Goldreich, Oded, Silvio Micali, and Avi Wigderson. "Proofs that Yield Nothing but the Validity of the Assertion, and a Methodology of Cryptographic Protocol Design," *Proceedings of the* 27th *FOCS*, IEEE, 1986, 174–187.

[GMW87] Goldreich, Oded, Silvio Micali, and Avi Wigderson. "How to Play ANY Mental Game," *Proceedings of the* 19th *STOC*, ACM, 1987, 218–229.

[HPS] Hoel, Paul, Sidney Port, and Charles Stone. *Introduction to Probability Theory*, Houghton Mifflin, Boston, 1971.

[K] Knuth, Donald. *The Art of Computer Programming*, Vol. 1, Addison Wesley, Reading, 1973.

[KL] Karp, Richard and Richard Lipton. "Turing Machines that Take Advice," *L'Enseignement Mathématique* IIe *Série*, XXVIII, 3–4, 1982, 191–209.

[N] Nisan, Noam. In preparation.

[NZ] Niven, Ivan and Herbert Zuckerman. *An Introduction to the Theory of Numbers*, Wiley, New York, 1972.

[Pip] Pippenger, Nicholas. "On Simultaneous Resource Bounds," *Proceedings of the* 20th *FOCS*, IEEE, 1979, 307–311.

[Pra] Pratt, Vaughan. "Every Prime Has a Succinct Certificate," *SIAM J. on Comput.*, 4, 1975, 214–220.

[RAD] Rivest, Ron, Len Adleman, and Michael Dertouzos. "On Databanks and Privacy Homomorphisms," in *Foundations of Secure Computation*, R. A. DeMillo et. al., editors, Academic Press, Inc., New York, 1978, 168–177.

[Sto] Stockmeyer, Larry. "The Polynomial-Time Hierarchy," *TCS*, 3, 1977, 1-22.

[TW] Tompa, Martin and Heather Woll. "Random-Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information," *Proceedings of the* 28th *FOCS*, IEEE, 1987, 472–482.

[Yao82] Yao, Andrew C. "Protocols for Secure Computations," *Proceedings of the* 23rd *FOCS*, IEEE, 1982, 160–164.

[Yao86a] Yao, Andrew C. "How to Generate and Exchange Secrets," *Proceedings of the* 27th *FOCS*, IEEE, 1986, 162–167.

[Yao86b] Yao, Andrew C. "Intrinsically Indiscreet Problems," Workshop on Communication and Computing, October 22, 1986, MIT.

[Yap] Yap, Chee. "Some Consequences of Non-Uniform Conditions on Uniform Classes," *TCS*, 26, 1983, 287–300.