Limits on the Provable Consequences of One-way Functions

By

Steven Rudich

B.A. (Wesleyan University) 1984

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA at BERKELEY

Approved:

[illegible]	11/20/88
Chair	Date
[illegible]	11/21/88
Robert Solovay	Nov. 17, 1988

****

Limits on the Provable Consequences of One-way Functions.

Steven Rudich

Abstract

We present strong evidence that the implication, "if one-way permutations exist, then secure secret key agreement is possible", is not provable by standard techniques. Since both sides of this implication are widely believed true in real life, to show that the implication is false requires a new model. We consider a world where all parties have access to a black box for a randomly selected permutation. Being totally random, this permuation will be strongly one-way in a provable, information-theoretic way. We show that, if $P = NP$, no protocol for secret key agreement is secure in such a setting. Thus, to prove that a secret key agreement protocol which uses a one-way permutation as a black box is secure is as hard as proving $P \neq NP$. We also obtain, as a corollary, that there is an oracle relative to which the implication is false, i.e., there is a one-way permutation, yet secret-exchange is impossible. Thus, no technique which relativizes can prove that secret exchange can be based on any one-way permutation. Furthermore, we show that if a certain combinatorial conjecture is true, then there is similar evidence to show that a one-way permutation can't be contructed from a one-way function.

Our results present a general framework for proving statements of the form, "Cryptographic application X is not likely possible based solely on complexity assumption Y."

This research was supported by NSF grant CCR 88-13632 and by an IBM Doctoral Fellowship.

[illegible]

Committee Chairman: Manuel Blum

To NATHAN CONGDON

My first intellectual companion

Acknowledgements.

I am extremely grateful to Manuel Blum for everything. He has been a fantastic advisor. From the very beginning, he boosted my mathematical self-confidence by treating me as a peer. He also took the time to show me more than his perfected and polished ideas. He showed me the guts of research—false starts, fallacies, inchoate intuitions, and all. This exceptional honesty gave me a generous look at the inner working of a truly creative mind. Knowing Manuel has been, without any doubt, the high point of my intellectual life.

As far as this thesis is concerned, I owe Manuel more than a methodological debt. He was thinking about problems and issues treated here long before I came to graduate school. In particular, he asked if $P = NP$ implies that secret-key agreement is impossible with a random oracle, and conjectured that $P = NP$ implies $NP^R \bigcap Co-NP^R = P^R$. He pointed out that a random oracle provides an ideal one-way function, giving his conjectures a natural interpretation. He also detected the subtle possibility that we might simulate a random permutation oracle in such a way that it is invertible. As with all of Manuel's problems. I learned a great deal from these, not realizing the full depth of some of the issues for several years.

Russell Impagliazzo is my dear, dear friend and colleague. He never fails to have unique (often deeply unified) views on an issue. It is always wonderful to talk to him. Like Manuel, his intellectual honesty has always made it possible for me to learn from him.

This thesis is joint work with Russell. We will co-author all papers which result. We have discussed these questions with each other for about two years. The history of the secret-key agreement result includes about forty fallacious proofs; I could never have survived this cruel form of torture if I had been going it alone. A lesser co-author would have given up. Russell was always tenacious and insightful. If he were a sword, I would call him *Theorem Killer*.

I am very grateful for discussions with Amos Fiat, Mario Szegedy, and Gabor Tardos.

I want to thank Noam and Umesh for helping me check the details of the main result. I am also grateful to my proofreaders, David Feldman, Nathan Tawil, Charlie Rackoff, and Rachel Rue. Nathan's feedback clearly demonstrated the subtle relation between syntax and semantics.

My years at Berkeley were greatly enhanced by the faculty and my fellow students. Umesh Vazirani inspired me with his unique form of self-fulfilling optimism. Dick Karp is a flawless teacher; I now aspire to the standards of mathematical exposition he has set. I will be taking with me many mathematical manoeuvres which I learned by observing other students, Sampath Kannan, Moni Naor, Noam Nisan, and Ronitt Rubinfeld, to name a few.

Contents

1 Introduction.

Complexity theory is a field in its technical infancy. Problems greatly outnumber solutions. Even seemingly unsubtle questions, such as $PTIME = PSPACE$?, are likely to remain open into the next century. To sort out the relevant issues, one of the objects of study has become the set of open problems themselves.

Complexity theorists have developed a taxonomy of problem difficulty. NP-completeness theory was the first example of a method to classify by difficulty a seemingly diverse set of problems. Inspired by this example, theorists have isolated a core set of problems and complexity classes. Results take the form: Resolving problem X is as hard as resolving problem Y. Thus, researchers have maintained an understanding of an enlarging, diverse set of open problems.

In this thesis, we develop new techniques for obtaining results of the form: Proving that the existence of an ideal one-way function (or permutation) implies X, is as hard as proving $P \neq NP$. Consequently, we argue that certain statements are very hard to prove from the assumption that an ideal one-way function (or permutation) exists. The next two sections will describe the problems studied and the manner in which our techniques are applied.

1.1 The power of cryptographic assumptions.

A typical result in cryptography will be of the form: With assumption X, we can prove that a secure protocol for P is possible. Because the standard cryptographic assumptions are, at present, unproved, many results focus on weakening the assumptions known to imply that a given protocol is possible. As a consequence, we ask a new form of question: Which assumptions are too weak to yield a proof that a secure protocol for P is possible?

The protocol we will study is secure secret-key agreement. Secret-key agreement is a protocol where Alice and Bob, having no secret information in common, agree on a secret key over a public channel. Such a protocol is secure when no polynomial-time Eve listening to the conversation can determine part of the secret. Secure secret-key agreement is known to be possible under the assumption that trapdoor functions exist. Unsuccessful attempts have been made to base it on the weaker assumption that one-way functions exist.

We provide strong evidence that assuming a one-way permutation exists is unlikely to yield a proof that secure secret-key agreement is possible. We model the existence of a one-way permutation by assuming the availability of a random permutation oracle. A random permutation oracle is provably one-way in the strongest possible sense. We show that proving secure secret-key agreement possible in a world with a random permutation oracle would simultaneously prove $P \neq NP$. This implies the existence of an oracle relative to which one-way permutations exist, but secret-key agreement is impossible.

Combining our result with known implications in cryptography, we obtain forty-two natural corollaries:

No assumption from column A is likely to yield a result from column B.

A	B
One-way permutations exist	Secret-key agreement is possible
Signature schemes exist	Oblivious transfer is possible
Identification schemes exist	Trapdoor functions exist
Telephone coin flipping is possible	Multi-party computation is possible
Private-key cryptosystems exist	Public-key cryptosystems exist
Pseudo-random number generators exist	Voting schemes exist
Everything with an interactive protocol has a zero-knowledge protocol	

These are the first results of this type in cryptography.

1.2 One-way functions vs. one-way permutations.

Bennett and Gill[BG81] have raised the question: Can we construct a one-way permutation from a random oracle? (Actually, they did not raise this exact question, but Blum has pointed out that they meant to raise this question!) They showed that an affirmative answer would prove $NP^R \bigcap Co-NP^R \neq P^R$ with a random oracle. A cryptographic interpretation of this question would be: Given a one-way function can we build a one-way permutation? An affirmative answer would show that the large set of cryptographic protocols whose security is now based upon the existence of a one-way permutation could be based on a weaker assumption.

We wish to give a negative answer to this question. However, our results are based on an unproven combinatorial conjecture. Assuming this conjecture to be true, we show that constructing a one-way permutation from a one-way function would prove $P \neq NP$. Furthermore, the conjecture implies the existence of an oracle relative to which one-way functions exist, but one-way permutations do not.

2 Notation and definitions.

The notation and definitions, unless otherwise stated, are the standard ones encountered in a first-year complexity or cryptography course. The reader wishing to review these is directed to [AHU74, Sip85]. The probability theory used can be found in [Fel68].

Less familiar conventions:

We will abbreviate probabilistic polynomial-time Turing machine with the notation $PPTM$. The *computation of a $PPTM$ on a given input* will be a trace of the entire run of the machine given the input. (The computations are indexed by the possible random tapes.) If the machine is an oracle machine, this would include all the queries and answers received during the computation. (In this case, each computation would be determined by a random tape, and by a finite set of query-answer pairs.) We use the notation *poly* to refer to some polynomial function. Thus, we can use the freewheeling arithmetic $poly * poly = poly$. A *conversation* between two $PPTM$s is the history of writes to the cells of a common communication tape.

3 Dirichlet's pigeonhole principle.

We take this opportunity to remind the reader of a useful form of the pigeonhole principle:

Let M be a 0-1 matrix with a $1-\alpha$ proportion of 1s. For every $ab = \alpha$, a $1-a$ portion of the columns have at least a $1-b$ portion of 1s. (It suffices to note that the worst case is when the 0's are concentrated in an a by b rectangle.)

4 Cryptographic Preliminaries.

We start by giving the cryptographic definitions used in this thesis. Any informality is deliberate. Our results remain unaltered by any reasonable formalization of these definitions. Making an explicit choice would be an arbitrary restriction. The skeptical reader is welcome to fill in any missing formal details as he or she chooses.

4.1 Secret-Key Agreement.

A *secret-key agreement protocol* is a pair of $PPTM$s called Alice and Bob. Each machine has a set of private tapes: a random-bit tape, an input tape, two work tapes, and a secret tape. In addition, they have a common communication tape that both can read and write. A run of the protocol is as follows: Alice and Bob both start with the same integer l written in binary on their input tapes; Alice and Bob run, communicating via the common tape; Alice and Bob both write an l-length string on their secret tape. If this string is the same, Alice and Bob are said to *agree*. The entire history of the writes to the communication tape is called *the conversation*. $\alpha(l)$ will denote the probability that Alice and Bob agree on a secret of length l.

A $PPTM$ Eve *breaks* a secret-ket agreement protocol if Eve, given only the conversation, can guess the secret with probability $\alpha(l)/poly(l)$. A protocol is *secure* if no Eve can break it. One could imagine far more stringent notions of security. For example, we might require that Eve can't even get one bit of the secret. However, in our scenario, we will be breaking secret-key agreement in the strong sense defined above, thus including the weaker notions of breaking that an applied cryptographer would use. (For example, a cryptographer would be happy to learn one bit of the secret.)

4.2 One-way permutations.

A *one-way permutation* is a 1-1, onto, polynomial-time computable function from n-bit strings to n-bit strings, where the inverse permutation is not computable in polynomial-time. In fact, for cryptography we require that no *PPTM* can expect to invert the function on more than a $1/poly(n)$ fraction of the inputs of length n.

4.3 The state of the art.

For the reader familiar with the myriad of cryptographic protocols and assumptions. we cite some known connections to secret-key agreement protocols and one-way permutations:

The existence of a one-way permutation is known to imply that signature schemes exist[NY], identification schemes exist[FS86], telephone coin flipping is possible[Blu82], private-key cryptosystems exist[LR86], pseudo-random number generators exist[Yao82], and everything with an interactive protocol has a zero-knowledge interactive protocol[GMW87, IY87, BCC87].

Secure secret-key agreement is known to be possible under any of the following assumptions: Oblivious transfer is possible[Blu81, Rab81], trapdoor functions exist[DH76], multi-party computation is possible, public-key cryptosystems exist, and voting schemes exist[Ben87].

The reader is challenged to think of a natural cryptographic objective that is neither implied by one-way permutations, nor implies that secure secret-key agreement is possible. It seems that these assumptions partition known cryptographic objectives into two categories.

Thus, if one-way permutations do not suffice to show secure secret-key agreement, then no objective implied by one-way permutations suffices to show any objective which implies secure secret-key agreement.

5 Uniform Generation.

5.1 Polynomial-time relations.

A relation, R, is polynomial-time if we can decide xRy in time polynomial in $\|x\| + \|y\|$. In this thesis, we will only consider relations where the length of y is polynomially related to the length of x. *Is satisfied by* is an example of such a relation: x *is satisfied by* y iff x is a boolean formula and y is one of its satisfying assignments.

5.2 What is uniform generation?

Let R be the "is satisfied by" relation. We can ask two natural questions:

Existence Given x, does there exist a y such that xRy?

(Does a given formula has a satisfying assignment?)

Counting Given x, how many y exist such that xRy?

(How many satisfying assignments does a given formula have?)

The existence question, satisfiability, is NP-complete. The counting question, thought to be harder than satisfiability, is $\#P$-complete. Jerrum, Valiant, and Vazirani[JVV86] introduced a problem of intermediate complexity.

Uniform generation Given x, pick a y uniformly at random such that xRy.

(Given a formula, find a random satisfying assignment.)

More generally, let R be a polynomial-time relation. Let M be a $PPTM$ with a fixed (as opposed to expected) polynomial running time. We say M *uniformly generates* R if given x, M has at least a 50% chance of outputting a uniformly chosen y such that xRy; otherwise, M outputs "try again". If such a y does not exist, M will only output "try again". Notice that rerunning the algorithm when it fails to generate a random y will succeed in generating a random y in expected polynomial time.

5.3 $P = NP$ and uniform generation.

Theorem 5.1 (JVV) *For any polynomial-time relation, there exists a PPTM equipped with a $\sum_2^P$ oracle that uniformly generates it.*

Theorem 5.2 $P = NP \Longrightarrow$ *for any polynomial-time relation, there exists a PPTM that uniformly generates it.*

Proof: $P = NP \Rightarrow$ the polynomial-time hierarchy collapses[CKS81] $\Rightarrow$ a polynomial-time machine can simulate a $\sum_2^P$ oracle $\Rightarrow$we can use previous theorem to uniformly generate. ■

Let M be a $PPTM$. There are possibly many different computations of M consistent with a given input and output. (Of course, there may be none.) The following corollary shows that if $P = NP$, we can efficiently pick a random element from the finite set of these computations.

Corollary 5.1 $P = NP \Longrightarrow$ *it is possible to generate a random computation for a given PPTM, M, with given input, I, and given output, O, in expected polynomial time.*

Proof: Checking that the trace of a computation is consistent with M, I, and O is a polynomial-time relation. ■

Corollary 5.2 $P = NP \Longrightarrow$ *given a conversation, C, between two PPTMs M and N, we can uniformly generate a possible computation of M.*

Proof: Checking that C is consistent with a given computation of M is possible in polynomial-time. ■

5.4 An application to cryptography.

Public-key cryptography relies on the assumption that **P≠NP**. The formal version of this fact, $P = NP$ implies secret key agreement is not possible, is something one might see a rather technical proof of in a first-year course. We can use our results on uniform generation to give a particularly simple proof of the optimal result.

Theorem 5.3 $P = NP \Longrightarrow$ *Eve has an expected polynomial time algorithm to break any given secret key agreement protocol in the strongest possible sense: Eve will find the secret with exactly the same probability that Alice and Bob agree on one.*

Proof: Fix a computation and resulting secret for Bob. We will show that the probability that Alice agrees with Bob is the same as the probability that Eve agrees with Bob. By corollary 5.2, Eve can generate a random computation of Alice consistent with the conversation. Alice's particular computation is, by definition, a random computation of Alice consistent with the conversation. Thus, Eve and Alice produce secrets with exactly the same probability distribution. They must, therefore, have exactly the same probability of agreeing with Bob. In other words, from Bob's point of view, Alice and Eve think alike; he will fool Eve with exactly the same probability that he will fool Alice. ■

6 Random Oracles.

6.1 Definition.

Let r be a random real between 0 and 1, chosen with the uniform distribution; express r in binary notation. A *random oracle* is the set induced from r as follows: $\{x :$ the xth binary digit of r is a 1 $\}$.

With each random oracle R, we can associate a function f from n-bit strings to n-bit strings. $f(i)$ is defined by its length(i) binary digits; the jth digit is 1 iff $(2i+1)2^j \in R$. (Every natural is uniquely expressed as an odd times a power of 2.) Notice that as we vary over all possible R, we get all possible length-preserving functions, each one occurring with the same frequency. Furthermore, using R as an oracle, f is polynomial-time computable. Thus, a TM with a random oracle also has at its disposal an easy to compute length-preserving random function. The notions of a random oracle and a random function oracle will be used interchangeably. We will now show that this function is one-way in the strongest possible sense.

6.2 Random oracles and one-way functions.

First, for a given machine, we fix the input and the random-bit tape; we show the machine can only invert a very small fraction of functions as we vary over oracles.

Lemma 6.1 *Let T be an oracle PPTM with random oracle R and a fixed random-bit tape. Fix an input x of length n. Let f be the random function associated with R (as above). The probability that $f(T(x)) = x$, is less than $poly(n)/2^n$.*

over random oracles R

(Here, the random bit tape is fixed and R is allowed to vary.)

Proof: W.L.O.G., we can assume T queries the function oracle f. If T never asks a y such that $f(y) = x$, then the probability that $f(T(x)) = x$ is bounded by $1/2^n$. We argue that T never asks such a y with very high probability. Each time T asks a y the probability that $f(y) = x$ is equal to $1/2^n$; T asks only poly(n) many queries; T asks such a y with probability less than $poly(n)/2^n$. The probability that $f(T(x)) = x$ is therefore bounded by $(poly(n) + 1)/2^n$. ∎

Now, for a given machine, we fix the oracle; we show that the machine has a low expectation of inverting as we vary over inputs and random-bit tapes.

Lemma 6.2 *Let T be an oracle PPTM with random oracle R. Let f be the random function associated with R. There exists a $poly(n)$ such that for every length n, there is a $1 - 1/n^2$ measure of oracles (f's) for which the expectation that $f(T(x)) = x$ on a random input of length n, is less than $poly(n)/2^n$.*

Proof: Consider the matrix whose rows are labeled by inputs of length n and fixed random-bit tapes, and whose columns are labeled with all possible functions from n-bits to n-bits. Put a 1 in position $< i, f >$ iff T on input i with oracle f outputs y such that $f(y) = i$. By lemma 6.1, each row has a $1 - poly(n)/2^n$ proportion of 0's. By the pigeonhole principle, $1 - 1/n^2$ proportion of the columns have more than $1 - (n^2 poly(n))/2^n$ proportion of 0's. The result follows. ∎

We say T with f *inverts better than* $< poly(n), n >$, when the expectation that $f(T(x)) = x$, on a random x of length n, is more than $poly(n)/2^n$.

Lemma 6.3 *For every oracle PPTM T, there exists a poly such that for most oracles, T with f inverts better than $< poly(n), n >$ for only finitely many n.*

Proof: Fix T. By lemma 6.2, there exists a poly such that the measure of f's for which T inverts better than $< poly(n), n >$ is bounded by $1/n^2$. $\sum_{n=0}^{\infty} 1/n^2$ converges; by the Borel-Cantelli lemma, measure one of oracles invert better than $< poly(n), n >$ only finitely often. ■

Theorem 6.1 *For most oracles, the function associated with the oracle is one-way in the strongest possible sense: For every oracle PPTM, there exists a poly, such that the machine has expectation no more than $poly(n)/2^n$ of inverting the inputs of length n.*

Proof: By lemma 6.3, for every oracle $PPTM$, we can throw out the measure zero of oracles where the machines invert well infinitely often (as above). There are only countably many machines; we have thrown out measure zero of oracles in all. Therefore, the remaining measure one of oracles has the property that for every machine, there exists a *poly*, such that there are only finitely many lengths where the machine can invert more than $poly(n)/2^n$ of the inputs. By adding a large constant to *poly*, we can deal with the finite number of lengths on which the machine is able to invert frequently.

This is the strongest possible sense because a machine that samples f at $poly(n)$ random points has a $poly(n)/2^n$ chance of finding the inverse. Such a machine has expectation equal to $poly(n)/2^n$ of inverting a random ~~inputs~~ input of length n. ■

In [Bra83, Bra81], Brassard solves what seems to be the more difficult problem of explicitly constructing a strongly one-way function oracle. (He uses no randomness or counting techniques.)

6.3 Random oracles and uniform generation.

Theorem 6.1 implies that uniform generation is impossible in a random world; it is impossible to uniformly generate an inverse to the function associated with the oracle. One of the projects in this thesis is, assuming $P = NP$, to break secret key exchange in a random world. (In theorem 5.3, we saw how to break it in the real world.) Even though we can't hope for uniform generation in a random world (which would make life very easy), we can prove weak analogues of the uniform generation results, which will be helpful in realizing our goal.

The idea is not to generate the computation of an oracle $PPTM$, M, with a particular random oracle, but rather, with a *random* random oracle; we want a random computation of the machine over all possible oracles. Let $M^{I,O}$ be the finite set of possible computations of M given input I, output O, using some oracle. (These computations are indexed by the random-bit tape, and the oracle query-answer pairs used during the computation.) A natural probability distribution to put on $M^{I,O}$ is to weight each computation by the probability that it occurs using a random oracle. We want to be able to pick a random element of the space $M^{I,O}$. Note: This time the distribution on the underlying set is not necessarily uniform. The probability of a computation with q queries being chosen is $2^{-q}/2^{-p}$ as likely as a computation with p queries being chosen.

Theorem 6.2 *$P = NP \Longrightarrow$ there exists a $PPTM$ that picks a random element from the probability space $M^{I,O}$ in expected polynomial time.*

Proof: From the oracle $PPTM$ M, we construct a $PPTM$ M', such that a uniformly generated computation of M' given input I and output O, when suitably syntactically modified, yields a random element of the probability space $M^{I,O}$. Intuitively, M' is an oracle machine that makes up its own oracle on the fly.

Without loss of generality, assume the computation of M never makes the same oracle query twice; keep track of queries asked in a table, and use the oracle only when the table does not have the answer. Let $t(n)$ be a polynomial bound on the number of oracle queries M asks given an input of length n. M' starts its computation by writing down $t(n)$ random bits on a separate tape, called the *answer tape*. M' then proceeds as M would, except that when M asks the oracle for a query answer, M' answers the simulated query with the first unused bit from the answer tape. By corollary 5.2, we can generate a random computation m' of M', with input I and output O, in expected polynomial time. To make m' look like a random computation of M, strip away the answer tape, pretending that all answers came from an oracle; call the computation that remains m. The probability associated with an m asking q queries is proportional to 2^{-q}. Hence, m is a random element of $M^{I,O}$. ∎

We can strengthen our result slightly by fixing some finite portion of the oracles we wish to consider. Let E be a finite set of oracle addresses and their contents. An oracle is said to be *consistent* with E if the content-address pairs in E are also in the oracle. We define a space similar to $M^{I,O}$: $M_E^{I,O}$ is a finite set of computations of M given I and O, using oracles consistent with E. Each element in $M_E^{I,O}$ is weighed by the probability of it occurring using a random oracle consistent with E. Once again, we wish to pick a random element of the space.

Theorem 6.3 $P = NP \Longrightarrow$ *there exists a PPTM that picks a random element of the probability space* $M_E^{I,O}$ *in expected polynomial time.*

Proof: Same as the proof of the previous theorem with one important modification: Hardwire the answers to oracle queries in E into the finite state control of M'. When M' asks a query in E, *do not* use a bit from the answer tape. ■

We can now prove the analogue of corollary 5.2 using **oracle** $PPTM$s Alice and Bob. In the case where oracle Alice and oracle Bob have conversation C, and E is a finite set of queries and answers, we define another similar space: A_E^C is the space of possible computations of oracle Alice consistent with the conversation C, where each computation is weighed by its probability of occurring with a random oracle consistent with E. The next theorem will be very important in the results on secret key agreement.

Theorem 6.4 $P = NP \Longrightarrow$ *there exists a PPTM that picks a random element of* A_E^C *in expected polynomial time.*

Proof: From Alice's point of view a conversation is a set of inputs and outputs occurring at certain prescribed times during her computation. No further modification of the above proof technique is required. ■

7 Random Permutation Oracles.

7.1 Introduction.

Random permutation oracles are similar to the random function oracles discussed in the previous section, except that the random functions must be 1-1 onto. A *random permutation oracle* Π is a random length-preserving function from the set of finite strings *onto* itself. Again, the function is chosen from the uniform distribution. The relationship between random oracles and random permutation oracles is one of the subjects of this thesis.

7.2 Random oracles are quite similar to random permutation oracles.

From the point of view of oracle $PPTM$s, there is no difference between the two types of oracles. We will formalize this in the spirit of pseudo-randomness.

A *tester* is an oracle $PPTM$ which, given n and a function oracle from n-bit strings to n-bit strings, outputs either 0 or 1. Let T be a tester. Let P_n be the probability that T will output a 0, when given n and a random function from n-bit strings to n-bit strings. Let P'_n be the probability that T will output a 0, when given n and a random permutation from n-bit strings to n-bit strings. Let $D_{T_n} = |P_n - P'_n|$. Thus, D_{T_n} measures how well the tester can distinguish between the two types of oracles.

Theorem 7.1 *For every tester T, $D_{T_n} < poly(n)/2^n$*

Proof: Assume T makes $q < poly(n)$ queries. In the case of a random function oracle, the answer to a previously unasked query is a random n-bit number, independent of the answers to previously asked queries. Thus, for each query made the probability that it gets the same answer as a previously made query is less than $q/2^n$. Summing, we conclude that the probability that two queries received the same answer is less than $q^2/2^n$. Next we observe that the distribution on possible query answers, given that all query answers are different, is the same for random function oracles and random permutation oracles; the probability that T will output a 0 given that all query answers are different, is the same for the two types of oracles. It follows that $D_{T_n} < q^2/2^n$. ∎

There are further similarities between the two types of oracle:

Theorem 7.2 *Measure one of random permutation oracles are one-way in the strongest possible sense: For every oracle PPTM, there exists a poly, such that the machine has expectation no more than $poly(n)/2^n$ of inverting the inputs of length n.*

Proof: Simply substitute the word "permutation" for the word "function" in the proofs of the lemmas leading up to Theorem 6.1. ∎

Proposition: Given a black box for a random permutation oracle, we can efficiently simulate a random oracle.

Proof: When we get the query "is x in the oracle?", answer yes iff $\pi(2x) > \pi(2x+1)$. This is an independent random bit. ∎

The proof of the next theorem is due to M. Naor[Nao].

Theorem 7.3 *We can efficiently simulate an (almost) random permutation oracle given a random oracle.*

Before we prove this theorem, we cite a theorem of Aldous and Diaconis[AD86].

Given a deck of $2n$ cards, a *random shuffle* consists of the following operations: Make n pairs of cards; pair i has cards number i and $n+i$. Order the cards in each pair randomly (according to a fair coin flip). Make up the shuffled deck of $2n$ cards by taking the (now ordered) pairs in order.

Theorem 7.4 (Aldous and Diaconis) *After $O(\log^2 n)$ random shuffles a deck with $2n$ cards is in nearly random order. (I.e., The distribution on possible orders is exponentially close to the uniform distribution.)*

Proof:[of theorem 7.3] Think of the random oracle as indexing the results of the coin flips in a random shuffle. Thus, a stretch of the oracle determines a random shuffle. If we want to know where the card in the ith position will be after a shuffle, we simply check one bit of the oracle.

Given an n-bit number x, we want to compute $\pi(x)$. Think of the possible x's as a deck of 2^n. By the Aldous-Diaconis result, we know $O(\log^2(2^n)) = O(n^2)$ random shuffles will (almost) perfectly shuffle the deck. The random oracle can index the required sequence of shuffles. What we do is figure out where all those shuffles take the xth card. The first takes x to x_1, the second takes x_1 to x_2, the third takes x_2 to x_3, and so on. By above remarks, it only takes one query to the oracle (and some trivial arithmetic) to compute x_{i+1} from x_i. Hence, we can compute $\pi(x)$ in $O(n^2)$ time. ∎

7.3 Similarities can be deceiving.

The two types of oracle might seem interchangeable. However, the astute reader will notice that in the above theorem the simulated permutation oracle is **invertible** relative to the random oracle from which it was constructed! Thus, random oracle worlds have random permutation oracles, but not necessarily one-way permutation oracles. (Though random permutation oracles are super-one-way relative to themselves, they might always be invertible relative to a more primitive black box from which they are built.) Despite the other strong similarities, one can't always assume that anything true relative to a random permutation oracle is true relative to a random oracle. The question of whether a one-way permutation oracle can be built from a random oracle is the subject of a later section.

8 Cryptographic Lower Bounds.

8.1 Introduction.

We will show that the existence of a very strong one-way permutation is not an assumption likely to yield a proof that secure secret key agreement is possible. By theorem 7.2, we know that a random permutation oracle is one-way in the strongest possible sense. Therefore, we will use the availability of a random permutation oracle to model the existence of an ideal one-way permutation. We will show that it is as hard to prove secure secret key agreement is possible using a common random permutation oracle as it is to prove $P \neq NP$. The result will take the form of the contrapositive: $P = NP$ implies that any secret-key agreement protocol can broken even when a random permutation oracle is available to all parties.

Summarizing the results of this section: We first show that $P = NP$ implies there is no secret-key agreement protocol that is secure with measure $1/poly$ of random oracles (random function oracles). Theorem 7.1 will be used to extend the result to random permutation oracles. Further strengthening the result by swapping the quantifiers, we show $P = NP$ implies for measure one of oracles there is no secure secret-key agreement. A corollary of this result is the existence of an oracle relative to which one-way permutations exist, but secure secret-key agreement is impossible. We also distinguish between two strong senses of breaking a secret-key agreement protocol.

8.2 A normal form for secret-key agreement.

To facilitate our analysis, we will assume that the secret-key agreement protocol has a normal form. Communication takes place in n rounds. Each round involves one person speaking and computing. Before each round, the party who is to speak asks the oracle a single query, and then does some computation. If Alice speaks first, the protocol would take the following form: Alice queries the oracle; Alice computes; Alice speaks (i.e. writes on the communication tape); Bob queries the oracle; Bob computes; Bob speaks; Alice queries the oracle; Alice computes; Alice speaks; Bob queries the oracle; ...

Any protocol can be converted to normal form with only a polynomial blow-up in running time.

8.3 Notation and definitions.

We wish to investigate a random world where Alice and Bob attempt to agree on an l-bit secret. In other words, we vary over runs of Alice, Bob, and Eve; and over oracles. Formally, a *world situation* is a five-tuple $< l$, $random_{Alice}$, $random_{Bob}$, $random_{Eve}$, $R >$. l, the input to Alice, Bob, and Eve, is the length of the secret being agreed upon. $random_{Alice}$, $random_{Bob}$, and $random_{Eve}$ are random bit tapes for Alice, Bob, and Eve to use during their computations (the random bit tapes are just long enough that they never get used up). R is a random oracle. Let WS_l be the set of all world situations where Alice and Bob attempt to agree on an l-length secret (l is the first entry of the five-tuple). We will also think of WS_l as a probability space with the uniform distribution. A world situation determines a random run of the protocol with a random oracle. With each world situation we can associate the following variables:

C_r, the conversation up to and including round r.

q_r, the query asked in round r.

A_r, the query-answer pairs Alice knows up to and including round r.

B_r, the query-answer pairs Bob knows up to and including round r.

If it is ambiguous which world situation C_r comes from, we write C_R^w to mean the conversation comes from world situation w.

World situation w *satisfies* C_r (written $w \models C_r$) means that the conversation between the machines in w is identical to C_r for the first r rounds. We will use the $\models$ notation with the other world situation variables as well.

Notice that none of the three polynomial time machines involved will be able to access the oracle past some very large address. Thus, without any loss, we can think of the oracle as finite. This means that the probability space WS_l is finite. Similarly any space we will discuss can be considered finite. This technical point will prevent the reader from suspecting any measure-theoretic fallacy.

8.4 Eve's sample space.

We need to define the probability distributions Eve samples from during her algorithm. They have already been described in section 6.3, Theorem 6.4. We define them again here.

Call a random tape for Alice *consistent* with conversation C_r and oracle R if the run of Alice, determined by the random tape and input from Bob's portion of C_r, outputs Alice's portion of C_r. (What she does after round r does not matter.) Let E be a finite set of query-answer pairs.

Let $AS_E^{C_r}$ be the set of <oracle, random tape for Alice> pairs such that E is in the oracle and the random tape for Alice is consistent with C_r and the oracle. Eve will be sampling from the space $A_E^{C_r}$ of computations of Alice consistent with C_r and the query-answer pairs in E. The distribution on $A_E^{C_r}$ is induced from the uniform distribution on $AS_E^{C_r}$; sample a point in $AS_E^{C_r}$, that point corresponds to a computation of Alice: An <oracle, random tape for Alice> pair corresponds to a <finite portion of the oracle used during the computation, random tape for Alice pair>.

8.5 Eve's algorithm.

We now give an algorithm for Eve to break a secret-key agreement protocol in a random world. This algorithm runs in polynomial time under the assumption that $P = NP$. S_l is a function of the form $1/poly$ (called a security parameter), which determines Eve's probability of failure. The smaller S_l, the longer Eve must run to break the protocol.

For each of n rounds of communication between Alice and Bob, Eve does $m = \lceil 3(n/S_l)\ln(2n/S_l)\rceil$ segments. Each segment has a simulate phase and an update phase. We will describe these phases in segment i for round r.

Without loss of generality, assume Alice speaks in round r. Let $E_{r,i-1}$ be the finite set of query-answer pairs that Eve knows about the oracle so far; $< q, a > \in E_{r,i-1}$ iff prior to round r, segment i, Eve has asked if q is in the oracle ($q \in R$?), and received answer a. Recall that C_r is the conversation that has occurred up to this round.

SIMULATION PHASE:

Using the method described in theorem 6.4, Eve picks a random run of Alice from the space $A^{C_r}_{E_{r,i-1}}$. (If Bob speaks in round r, Eve would instead simulate Bob.) Let $F_{r,i}$ be the set of queries that the simulated run of Alice asks her simulated oracle. (Note that so far in this segment, we have not asked any real oracle queries. Recall that when simulating a random Alice, we make up the answers to the oracle queries.)

UPDATING PHASE:

Eve asks all the queries in $F_{r,i}$ of the actual oracle R. Thus, $E_{r,i}$ equals $E_{r,i-1}$ union the new query-answer pairs Eve learned by asking $F_{r,i}$ of the oracle.

The following variables are also associated with any world situation:

$E_{r,i}$, the query-answer pairs Eve knows up to and including the ith segment of her simulation of round r.

$E_{r,0}$, the query-answer pairs Eve knows before she simulates round r. ($E_{r,0} = E_{r-1,m}$.)

$BPQ_{r,i}$, the query-answer pairs Bob knows and Eve does not, up to and including round r, segment i. BPQ stands for Bob's private queries. Note the relation: $BPQ_{r,i} = BPQ_{r,0} - E_{r,i}$.

8.6 Intersection queries and the secret.

Intersection queries are the queries Alice and Bob ask in common during an execution of their protocol. A particular query becomes an intersection query, not when it is first asked by one party, but rather when it is later asked by the other party. For conceptual unity, we can assume without loss of generality that the secret is an intersection query; assume that as their final act Alice and Bob query the oracle at the location addressed by the secret.

The next Theorem will prove that with high probability Eve finds all the intersection queries. Thus, Eve will have a polynomial-length list containing the secret; Eve breaks the protocol.

8.7 The efficacy of Eve's algorithm.

Theorem 8.1 *Suppose Alice and Bob attempt to agree on an l-length secret. The probability that Eve finds all the intersection queries is greater than $1 - S_l$. Formally,*

$PROB_{x \in WS_l}[A_n \bigcap B_n \subseteq E_{n,m}] > 1 - S_l$.

Proof: (We show the stronger result that Eve probably anticipates (asks) a query before it becomes an intersection query.) Eve's algorithm has n rounds. If Eve fails to find all intersection queries, there must be a first round where she fails to anticipate an intersection query that occurs in the next round; there exists a first time $q \in A_r \bigcap B_r$ and $q \notin E_{r-1,m}$. To formalize the event that Eve fails for the first time to anticipate an intersection query in the next round, we write it as the conjunct of three events:

- Eve has, in previous rounds, anticipated all intersection queries about to happen. (Thus, Eve knows all intersection queries to date.)
- q_{r+1}, the query asked in the next round, is an intersection query.

  AND
- Eve fails to find q_{r+1}. ($q_{r+1} \notin E_{r,m}$.)

Lemma 8.1, the technical heart of the proof, will show this event has probability no more than S_l/n by showing that the complementary event has probability greater than $1 - S_l/n$. Thus, for each round the probability of failing for the first time to anticipate an intersection query in the next round is less than S_l/n. Summing the error probability for each round, we get a total error probability bounded by S_l. ∎

Lemma 8.1 *The probability that in round r, either*

- *In a previous round, Eve failed to anticipate the intersection query about to happen,*
- *q_{r+1} is not an intersection query,*

  OR
- *Eve finds q_{r+1}*

is greater than $1 - S_l/n$.

Proof: Without loss of generality assume that Alice spoke in round r. We will prove the *stronger* result that *even when conditioned on any consistent choice* *of* C_r, $E_{r,0}$, $random_{Bob}$, and $BPQ_{r,0}$, the probability that one of the following occurs is greater than $1 - S_l/n$:

1. $A_r \bigcap BPQ_{r,0} \neq \emptyset$. (In the last round, Eve did not know all the intersection queries for round r: Alice asked a query in Bob's private query set making an intersection query that Eve had not anticipated. This is stronger than Eve failing to anticipate an intersection query in a previous round.)

2. $q_{r+1} \notin A_r$. (For $q_{r+1}$to be an intersection query in round r, Alice must have asked it *previously*.)

   OR

3. $q_{r+1} \in E_{r,m}$. (The definition of Eve finding q_{r+1}.)

Fix any consistent C_r^*, $E_{r,0}^*$, $random_{Bob}^*$, and $BPQ_{r,0}^*$. (The * superscript will later serve to disambiguate sets that have been previously defined from similar sets occurring as free variables in expressions.) An important consequence of fixing these variables is to fix q_{r+1}. $q_{r+1}$is to be asked by Bob in round $r + 1$; we have fixed his tape ($random_{Bob}$), his input (C_r^*), and his oracle queries and answers (contained in $E_{r,0}^* \cup BPQ_{r,0}^*$); thus, Bob's computation, up until he queries in round $r + 1$, has been fixed. $q_{r+1}$is therefore determined.

W will be the set of world situations satisfying these conditions:

$$W = \{w \in WS_l : (w \models C_r^*, E_{r,0}^*, random_{Bob}^*, BPQ_{r,0}^*)\ \}$$

We wish to show that a $1 - S_l/n$ fraction of W satisfies conditions 1, 2, or 3. Partition W into three sets P_1, P_2, and P_3:

$P_1 = \{w \in W :\ w \not\models (condition\ 1),\ w \not\models (condition\ 2)$, and

$$\exists i\ PROB_{x \in W}[\ x \models (conditions\ 1\ or\ 2) \mid x \models E_{r,i}^{w}] > 1 - S_l/2n\}$$

(Thus, $w \in P_1 \Rightarrow w \not\models (condition\ 1)$ and $w \not\models (condition\ 2)$, but an observer who only knew the $E_{r,i}$ portion of w would guess otherwise.)

$P_2 = \{w \in W : \; w \not\models (condition\ 1),\; w \not\models (condition\ 2)$, and

$$\forall i \; PROB_{x \in W}[\, x \models (conditions\ 1\ or\ 2) \mid x \models E^{w}_{r,i}] \leq 1 - S_l/2n\}$$

$P_3 = \{w \in W : \; w \models conditions\ 1\ or\ 2\ \}$

Clearly, $|P_1| + |P_2| + |P_3| \; = \; |W|$.

Claim 1 : $\frac{|P_1|}{|W|} \leq \frac{S_l}{2n}$

Proof:

For each element $x \in W$, we can associate a number $First_x$: The first i such that $PROB_{y \in W}[\, y \models (conditions\ 1\ or\ 2) \mid y \models E^{x}_{r,i}] > 1 - S_l/2n$; if there is no such i, let $First_x = \infty$.

Partition W as follows: Place all x such that $First_x = \infty$ in the same block. The remaining x's index the other blocks. $Block_x$ consists of all y such that $y \models E^{x}_{r,First_x}$ and $Random^{x}_{Eve} = Random^{y}_{Eve}$. (We have a partition because x, y in the same block implies that $First_x = First_y$; the Eve's in x and y must behave exactly the same until the $First_x$th segment.)

We will show that each block of the partition has a small intersection with P_1. The block where $First_x = \infty$ has no intersection with P_1. Consider one of the remaining blocks, $Block_x$. The proportion of elements of $Block_x$ not in P_1 is given by:

$$PROB_{w \in W}[\, w \models (conditions\ 1\ or\ 2)$$
$$\mid w \models E^{x}_{r,First_x} \text{ and } Random^{x}_{Eve} = Random^{w}_{Eve}]$$

Conditions 1 and 2 are independent of $Random^{w}_{Eve}$. Therefore, we rewrite our proportion:

$$PROB_{w \in W}[\, w \models (conditions\ 1\ or\ 2) \mid w \models E^{x}_{r,First_x}]$$

Which, by definition of $First_x$, is greater than $1 - S_l/2n$. Thus, P_1 makes up less than $S_l/2n$ proportion of each block, and hence of the whole space W.

■

Thus, P_1 is small and can be ignored. By definition, all of P_3 satisfies conditions 1 or 2, and hence can only help us. We now show that for most world situations in P_2, Eve finds q_{r+1}.

We give an overview of our strategy: First, we argue that when Eve (in P_2) samples from her space, and happens to produce an Alice whose queries do not intersect $BPQ_{r,i}$, she has a pretty good chance of finding q_{r+1}. Second, we note that when Eve samples an Alice who does intersect $BPQ_{r,i}$, Eve will learn a new query in $BPQ_{r,i}$ in her update phase. (This can only happen size of $BPQ_{r,i}$ times.) Therefore, there will be many segments where Eve has a pretty good chance of finding q_{r+1}.

By definition,

$\forall w \in P_2 \forall i,$

$$PROB_{x \in W}[\, x \models (\mathit{conditions}\ 1\ \mathit{or}\ 2) \mid x \models E^w_{r,i}] \leq 1 - S_l/2n$$

Equivalently, we can describe the probability of the complementary event:

$\forall w \in P_2 \forall i,$

$$PROB_{x \in W}[A^x_r \bigcap BPQ^x_{r,0} = \emptyset\ \mathit{and}\ q_{r+1} \in A^x_r \mid x \models E^w_{r,i}\,] > S_l/2n$$

Recall $x, w \in W$. $BPQ^x_{r,0} = BPQ^*_{r,0} = BPQ^w_{r,0}$. Furthermore, $BPQ^w_{r,i} = BPQ^w_{r,0} - E^w_{r,i}$, implying $BPQ^w_{r,i} \subseteq BPQ^x_{r,0}$. Substituting we get:

$\forall w \in P_2 \forall i,$

$$PROB_{x \in W}[A^x_r \bigcap BPQ^w_{r,i} = \emptyset\ \mathit{and}\ q_{r+1} \in A^x_r \mid x \models E^w_{r,i}\,] > S_l/2n$$

Which implies the weaker statement:

$\forall w \in P_2 \forall i,$

$$PROB_{x\in W}[q_{r+1} \in A_r^x \mid x \models E_{r,i}^w \ and \ A_r^x \bigcap BPQ_{r,i}^w = \emptyset\,] > S_l/2n$$

The next claim is very important. It relates distribution W to the space $A_{E_{r,i}}^{C_r^*}$ from which Eve samples. Let e be a random element of $A_{E_{r,i}}^{C_r^*}$; e_r will denote all the oracle queries the Alice in e asks up to and including round r.

Claim 2 : *For any fixed $E_{r,i}^*$ and $BPQ_{r,i}^*$,*

$$PROB_{x\in W}[q_{r+1} \in A_r^x \mid (x \models E_{r,i}^*, BPQ_{r,i}^*) \ and \ A_r^x \bigcap BPQ_{r,i}^* = \emptyset\,] =$$
$$PROB_{e\in A_{E_{r,i}^*}^{C_r^*}}[q_{r+1} \in e_r \mid e_r \bigcap BPQ_{r,i}^* = \emptyset]$$

Proof: By definition of W,

$$PROB_{x\in W}[q_{r+1} \in A_r^x \mid (x \models E_{r,i}^*, BPQ_{r,i}^*) \ and \ A_r^x \bigcap BPQ_{r,i}^* = \emptyset\,] =$$
$$PROB_{x\in WS_l}[q_{r+1} \in A_r^x \mid$$
$$(x \models E_{r,i}^*,\ E_{r,0}^*,\ C_r^*,\ random_{Bob}^*,\ BPQ_{r,0}^*,\ BPQ_{r,i}^*) \ and \ A_r^x \bigcap BPQ_{r,i}^* = \emptyset\,]$$

We will consider the two relevant probability spaces as sets with the uniform distribution:

$S_1 = \{x \in WS_l :$

$(x \models E_{r,i}^*,\ E_{r,0}^*,\ C_r^*,\ random_{Bob}^*,\ BPQ_{r,0}^*,\ BPQ_{r,i}^*) \ and \ A_r^x \cap BPQ_{r,i}^* = \emptyset\,\}$

$S_2 = \{e \in AS_{E_{r,i}^*}^{C_r^*} : A_r^e \cap BPQ_{r,i}^* = \emptyset\,\}$

Partition S_1 by equating $x, x' \in S_1$ iff they are the same on the oracle and $random_{Alice}$. (They are all the same on $random_{Bob}^*$ and may differ on $random_{Eve}$.) Each block has the same size; if a $random_{Eve}$ is possible with one <oracle, $random_{Alice}$ > pair, it is possible with another. (Simply recall that $E_{r,i}$ and C_r are fixed.) Notice that if $q_{r+1} \in A_r^x$ for one element of a block, then the same holds for all elements of the block. A block is described by a $random_{Alice}$ and an oracle containing $BPQ_{r,i}^* \cup E_{r,i}^*$.

Partition S_2 by equating $e, e' \in S_2$ iff they are the same everywhere except the oracle locations in the set $BPQ^*_{r,i}$. Each block of the partition has the same size $(2^{|BPQ^*_{r,i}|})$; recall that $A^e_r \bigcap BPQ^*_{r,i} = \emptyset$. If $q_{r+1} \in A^e_r$ for one element of a block, then the same holds for all elements of the block. A block is described by a $random_{Alice}$ and an oracle containing $E^*_{r,i}$ in addition to "don't care" as the contents of the addresses contained in $BPQ^*_{r,i}$. Thus, there is a 1-1 correspondence between the blocks of $S1$ and the blocks of $S2$.

These facts yield the desired claim. ∎

We conclude:

$\forall w \in P_2 \, \forall i,$ (1)

$$PROB_{e \in A^{C^*_r}_{E^w_{r,i}}} [q_{r+1} \in e_r \mid e_r \bigcap BPQ^w_{r,i} = \emptyset] > S_l/2n$$

Claim 3 : *In every world situation, for each round r, there are at least $m - n$ segments where $F_{r,i} \bigcap BPQ_{r,i} = \emptyset$.*

Proof: At the beginning of round r, Bob's private query set has no more than n elements (there is only one query per round). After each segment, the cardinality of Bob's private query set can only decrease. In a segment where $F_{r,i} \bigcap BPQ_{r,i} \neq \emptyset$, Eve will discover a new query-answer pair in Bob's private query set during the update phase; thus, Eve will decrease Bob's private query set by at least one. By the above remarks, this can only happen n times. The other $m - n$ segments must therefore satisfy $F_{r,i} \bigcap BPQ_{r,i} = \emptyset$. ∎

We can think of a random $w \in W$ as being generated, segment by segment, as follows: Conditioned on everything so far, pick random values for the variables in the next segment. In fact, in each segment, we pick a random $e \in A^{C^*_r}_{E^w_{r,i}}$

(recall $random^{*}_{Bob}$ is fixed).

Furthermore, we can think of a random $e \in A^{C^*_r}_{E^w_{r,i}}$ as being generated as follows: Let b be the probability that a random $e \in A^{C^*_r}_{E^w_{r,i}}$ has no intersection with $BPQ_{r,i}$. Flip a coin of bias b. If heads, then pick a random $e \in A^{C^*_r}_{E^w_{r,i}}$ given that $e \bigcap BPQ_{r,i} = \emptyset$. If tails, pick a random $e \in A^{C^*_r}_{E^w_{r,i}}$ given that $e \bigcap BPQ_{r,i} \neq \emptyset$.

The above claim tells us that there are at least $m - n$ segments where $F_{r,i} \bigcap BPQ_{r,i} = \emptyset$. Therefore, in the generation of each world situation there are $m - n$ segments where the coin comes up heads. In each of these segments, we pick a random $e \in A^{C^*_r}_{E^w_{r,i}}$ conditioned on $e_{r,i} \bigcap BPQ_{r,i} = \emptyset$. By inequality 1, we conclude that in each $w \in P_2$, there are $m - n$ segments where we have a greater than $S_l/2n$ chance of finding q_{r+1}. The proportion of P_2 where we fail to find $q_{r+1}$is therefore bounded by $(1 - S_l/2n)^{m-n}$.

We now calculate, using the fact that $(1 - 1/x)^x < 1/e$. Also $m - n > 2(n/S_l)\ln(2n/S_l)$.

$$(1 - \frac{S_l}{2n})^{2\frac{n}{S_l}\ln\frac{2n}{S_l}}$$
$$= \left((1 - \frac{S_l}{2n})^{\frac{2n}{S_l}}\right)^{\ln\frac{2n}{S_l}}$$
$$< (\frac{1}{e})^{\ln\frac{2n}{S_l}}$$
$$= \frac{S_l}{2n}$$

We conclude that a $1 - S_l/2n$ portion of P_2 satisfies condition 3. Recall, P_1 accounts for no more than a $S_l/2n$ portion of W, and all of P_3 satisfies conditions 1 or 2. Therefore, at least a $1 - S_l$ portion of W satisfies conditions 1, 2, or 3. This yields the desired result.

■

Theorem 8.2 *Theorem 8.1 is true relative to a random permutation oracle:*

Given any secret-key agreement protocol and a random permutation oracle, the probability that Eve finds all the intersection queries is greater than $1 - S_l/2$.

Proof: Assume not. We will construct a tester to distinguish between a random function oracle and a random permutation oracle. We start with a protocol where Eve will find all the intersection queries with probability less than $1-S_l/2$ if a random permutation oracle is used, and probability greater than $1 - S_l$ if a random function oracle is used. A tester can simulate runs of Alice, Bob, and Eve, counting the fraction of times Eve finds all the intersection queries. The essence of the situation is that the tester is flipping a coin with two possible biases: $1 - S_l/2$ and $1 - S_l$; the tester must guess which. If the tester flips the coin $1/{S_l}^2$ times, even a very weak form of the law of large numbers would tell us that Eve can guess the bias of the coin at least 99% of the time. This very strongly contradicts theorem 7.1. ∎

Notice the order of the quantifiers in the above result. We picked the protocol between Alice and Bob, then we picked the oracle (since the protocol is bound by definition to work with a random oracle). Then, we showed Eve can break the protocol. We prove a stronger result which reverses the quantifiers. First, we pick a random oracle; then a protocol for Alice and Bob (this time the protocol need not work properly on other oracles). Then, we show that Eve can break the protocol relative to the chosen oracle.

Theorem 8.3 $P = NP \Longrightarrow$ *relative to a random permutation oracle, any secret key agreement scheme can be broken.*

Proof: First, we argue that for every secret-key agreement protocol, there are only measure zero of oracles where it can't be broken. Fix a protocol. The $P = NP$ assumption allows us to use Eve's algorithm as before. Choose $S_l =$

$1/l^{2+\epsilon}$. Theorem 8.1 tells us that in $1 - S_l/2$ of world situations we succeed in breaking the protocol. By the pigeon-hole principle, for each length l, there are $1 - \sqrt{S_l/2}$ oracles relative to which there is a $1 - \sqrt{S_l/2}$ chance of Eve breaking the protocol. Call all such oracles good for length l. The probability that a random oracle fails to be good for length l is $\sqrt{S_l/2}$. $\sum_{l=0}^{\infty} \sqrt{S_l/2}$ converges; by the Borel-Cantelli lemma, measure one of oracles are good on all but finitely many lengths. For measure one of the oracles, past some length, Eve has a $1 - \sqrt{S_l/2}$ chance of breaking the protocol. (We can even non-uniformly boost Eve's ability to break protocols for finitely many lengths.) Thus, there are only measure zero oracles where the protocol can't be broken.

For each of the countably many protocols we throw out the measure zero of oracles where the protocol is secure. We have thrown out measure zero in all. Every protocol can be broken relative to the measure one of remaining oracles.

■

Corollary 8.1 *There exists an oracle relative to which a strongly one-way permutation exists, but secure secret-key agreement is impossible.*

Proof: Consider any oracle world where $P = NP$. Add a random permutation oracle to this world. Because all the techniques in our theorem relativize, we can conclude that secure secret-key agreement is not possible in the resulting world.

Construct an example of such an oracle as follows: The even numbers form an oracle for PSPACE (a PSPACE-complete problem), the odd numbers form a random permutation oracle. $P = NP$ relative to a PSPACE-complete oracle. We know the random permutation is one-way in the strongest possible sense.

■

The only other relativized result that I know in cryptography is Brassard[Bra83, Bra]. He explicitly constructs an oracle where secret-key agreement is possible.

So far, our sense of breaking a secret key agreement consists of finding a polynomial-sized list with the secret on it somewhere. The strongest sense of breaking secret key agreement is clearly to find the secret itself. We show how to extend Eve to actually find the secret. For the same reasons as before, the argument works equally well with both random oracles and random permutation oracles.

Eve's strategy can be extended as follows: Eve's final round will be her simulation of the $n-1$th round of the protocol. In each segment of her final round, Eve records her last query to the oracle. (Recall that the last query to the oracle should be thought of as the secret.) Of the final queries Eve has recorded, she outputs the one which occurs the majority of the time. (If there is no majority, output "failure".)

Theorem 8.4 *Suppose that Alice and Bob agree on a secret with probability at least $1-\alpha$ over world situations in WS_l. Then, for every $\delta > 0$, there exists an Eve who can guess the secret with probability at least $1-\alpha(2+\delta)$ over world situations in WS_l.*

Proof: Without loss of generality, assume Alice asks her final query (the secret) in round $n-1$; then Bob asks his final query (the secret) in round n. The proof hinges on two inequalities. The first is the formal statement of what we are assuming:

$$PROB_{w\in WS_l}[\text{Alice and Bob agree on a secret }] \geq 1-\alpha_l$$

The second is a consequence of Theorem 8.1; if no intersection queries were missed, then Alice could not make an unanticipated intersection query in her final round.

$PROB_{w \in WS_l}[A^w_{n-1} \bigcap BPQ^w_{n-1,i} \neq \emptyset] \leq S_l$

We abbreviate the expression of two events: $FINDSECRET^w_i$ iff in the ith segment of the last round of Eve's simulation in world situation w, Eve's last query is the secret. $ASKBPQ^w_i$ iff in the ith segement of the last round of Eve's simulation in world situation w, Eve asks a query in $BPQ_{n-1,i}$. $NOTASKBPQ^w_i$ is the complementary event.

Fix a segment i. We will lower bound the following quantity in a four step calculation:

$PROB_{w \in WS_l}[FINDSECRET^w_i$ or $ASKBPQ^w_i]$

$\geq PROB_{w \in WS_l}[FINDSECRET^w_i \mid NOTASKBPQ^w_i]$

$= PROB_{w \in WS_l}[$Alice and Bob agree on a secret $\mid A^w_{n-1} \bigcap BPQ^w_{n-1,i} \neq \emptyset]$

(Partition WS_l according to $random_{BOB}$, C_{n-1}, $E_{n-1,i}$, and $BPQ_{n-1,i}$, and apply claim 2 of lemma 8.1 to each block. Alice and Bob will agree on the secret iff $q_{r+1} \in A^w_{n-1}$.)

$\geq PROB_{w \in WS_l}[$Alice and Bob agree on a secret$]-$

$PROB_{w \in WS_l}[A^w_{n-1} \bigcap BPQ^w_{n-1,i} \neq \emptyset]$

$\geq 1 - \alpha_l - S_l$

Thus, for any segement i, $PROB_{w \in WS_l}[FINDSECRET^w_i$ or $ASKBPQ^w_i] \geq 1 - \alpha_l - S_l$. By the pigeonhole principle, there is a $1 - (\alpha_l + S_l)((2 - \epsilon)$ fraction of world situations where a $1/((2-\epsilon)$ fraction of the segments of the final round satisfy $FINDSECRET^w_i$ or $ASKBPQ^w_i$. By Claim 3 of lemma 8.1, the fraction of segments of any round of any world situation which satisfy $ASKBPQ^w_i$ is smaller than n/m. As l increases, n/m is smaller than any constant. Therefore, there is a $1-(\alpha_l+S_l)((2-\epsilon)$ fraction of world situations where the majority of segments of the final round find the secret. By choosing sufficiently small S_l and ϵ, Eve can guess the secret with probability at least $1 - \alpha_l(2 + \delta)$. ∎

8.8 Related results.

A result similar to theorem 8.4 is claimed by Judit Bar-ilan and Michael Ben-or, however, I am unable to understand their proof.

Merkle[Mer78] has shown that relative to a random oracle there is a secret-key agreement protocol where Alice and Bob can agree on a secret in time n, but Eve requires $O(n^2)$ time to break it. Choosing $S_l = 1/100$ in theorem 8.4, we see that Eve can break any protocol in time $O(n^3 \ln n)$. (Bar-ilan, Ben-or claim the tight result that Eve can break any protocol in time $O(n^2)$.)

9 Random functions vs. one-way permutations.

9.1 Introduction

Two natural objects in cryptography are one-way functions and one-way permutations. We address the question: Assuming the existence of one-way functions, can one prove the existence of one-way permutations? We wish to answer this question in the negative, using the formal strategy developed in earlier sections.

Once again, we model a one-way function by a random function (oracle). We wish to show: If one can prove a one-way permutation exists in a random world, then one can prove $P \neq NP$. Formally, $P = NP$ implies there are no one-way permutations in a random world. However, we are unable to prove a combinatorial conjecture which is key to obtaining this result. Therefore, we present the theorem that the conjecture implies the desired result. The conjecture itself, though resistant to proof, is quite plausible.

Some earlier work of Blum[Bl87], Hartmanis and Hemachandra[HH87], and Tardos[Tar] shows that $P = NP$ implies no oracle $PPTM$computes a one-way permutation on all oracles. We can interpret their result as saying that a uniform efficient method of constructing a one-way permutation from a random (one-way) function would prove $P \neq NP$. This does not rule out standard non-uniform constructions. (For example, one could imagine using a one-way function to construct a one-way function which computed a permutation on all but finitely many lengths; then concluding that a finite modification of the function would produce the desired one-way permutation.) Our target result rules out these constructions by arguing that (assuming $P = NP$) for most oracles no machine computes a one-way permutation. Furthermore, our result implies the existence of an oracle where one-way functions exist, but one-way permutations do not.

9.2 The combinatorial conjecture.

Let V be a set of variables, i.e., $V = \{w, x, y, z, \ldots\}$. Let T be a set of conjunctive terms over the variables in V, $\overset{eg}{\text{i.e.}}$, $T = \{xyz, wx, \bar{x}\bar{z}, \ldots\}$. For every $S \subset T$, $P_S \equiv$ the probability over uniformly chosen assignments to the variables in V that at least one term in S is satisfied. (A random assignment can be thought of as flipping a fair coin for each variable.) $U_S \equiv$ the probability over random assignments to variables in V that *exactly one* term of S is satisfied. For every term $t \in T$, $N(t) \equiv \{t' \in T : t'$ has a variable in common with $t.\}$. (x and $\bar{x}$ are considered the same variable.) $N(t)$ can be thought of as $\{t\}$ union the neighbor set of t in the graph whose nodes are the elements of T placing an edge between two nodes iff they have a variable in common.

Examples:

Let $V = \{x_1, x_2, x_3, \ldots, x_{10}\}$. Let T be the set of all 10 variable terms. There are 2^{10} such terms. $P_T = U_T = 1$. For every $t \in T$, $P_{\{t\}} = U_{\{t\}} = 1/2^{10}$. For every $t \in T$, $N(t) = T$.

Let $V_n = \{x_1, x_2, x_3, \ldots, x_{n2^n}\}$. Let T_n be a set of 2^n disjoint n-variable terms: $\{x_1x_2x_3\ldots x_n, x_{n+1}x_{n+2}x_{n+3}\ldots x_{2n}, \ldots, \ldots, x_{n2^n-n+1}x_{n2^n-n+2}x_{n2^n-n+3}\ldots x_{n2^n}\}$. $\lim_{n\to\infty} P_{T_n} = 1-(1/e)$. $\lim_{n\to\infty} U_{T_n} = 1/e$. For every $t \in T$, $N(t) = \{t\}$.

Conjecture: $\exists \epsilon, \delta > 0$ s.t. $\forall T$, $U_T > 1-\epsilon \Longrightarrow \exists t \in T$ such that $P_{N(t)} > \delta$.

9.3 Definitions and notation.

A non-deterministic polynomial-time oracle machine M with oracle O is an $NP \bigcap coNP$ machine if on each input M^O has either an accepting path or a rejecting path, but not both (although it might have many of either kind). Notice that a machine might be $NP \bigcap coNP$ for one oracle, but not for another.

L_M^O is the language accepted by such a machine on oracle O. *Query complexity* is a measure of complexity where we only count the number of queries to the oracle. The class QP^O contains all languages L_M^O where M is a deterministic machine asking only polynomial many queries to O on each input.

9.4 Prior work

The techniques we will be using are inspired by the next theorem. This theorem was discovered independently by Blum[Bl87], Hartmanis and Hemachandra[HH87], and Tardos[Tar].

Theorem 9.1 *Let M be an $NP \bigcap coNP$ machine on all oracles. $L_M^O \in QP^O$ for all oracles O.*

Proof: We will show how to decide "$x \in L_M^O$?" for any x on any oracle O. First some notation and an observation. Let E be any finite set of oracle queries and answers. For each oracle there is at least one *poly*-sized subset of oracle queries and answers which would cause M to accept or reject x. Let Y_E be the set of all such subsets which cause M to accept x on some oracle consistent with E. Let N_E be the set of all such subsets which cause M to reject x on some oracle consistent with E. Now for the critical observation: Any $y \in Y_E$ and any $n \in N_E$ have a query in common which is not contained in E. (If not, we can build an oracle consistent with the queries and answers in both y and n. $M(x)$ is not well defined on this oracle, contradicting our assumption.)

Now for the algorithm. At any point in the algorithm, E will denote the finite set of query answer pairs we have gotten from asking queries of O. (E starts as the empty set.) We terminate when $Y_E = \emptyset$ or $N_E = \emptyset$; In particular, if $Y_E = \emptyset$ then the answer to "$x \in L_M^O$?" must be no (and vice-versa). As long as the termination conditions are not met, we proceed as follows: Pick any

$y \in Y_E$. Ask O all the queries mentioned in y.(Update E.) Repeat.

We argue that the algorithm terminates after *poly* many queries. Each time we ask the queries in y, we ask only *poly* many queries. By our critical observation, each time we ask the queries in y, we reduce the number of unasked (not in E) queries in each element $n \in N_E$. Therefore, if we do *poly* many reductions without termination, there is an $n \in N_E$ all of whose queries are in E. Again by our critical observation, this implies $Y_E = \emptyset$, which causes termination. ∎

Corollary 9.1 *Let M be an $NP \bigcap coNP$ machine on all oracles. $P = NP \Longrightarrow L_M^O \in P^O$ for all oracles O.*

Proof: $P = NP$ implies that we can pick an element of Y_E or decide it is empty in polynomial time: The question "Does M have an accepting path whose oracle query answers are consistent with E?" is in NP. Use self-reducibility to actually find a path, if one exists. ∎

Corollary 9.2 *Let Π be an oracle PPTM which for every oracle and every n computes a permutation from n-bit strings to n-bit strings. $P = NP \Longrightarrow \Pi^{-1}$ is polynomial time computable relative to all oracles.*

Proof: There clearly exists an $NP \bigcap coNP$ machine which inverts Π on all oracles. ∎

This last corollary already shows that it would be very hard to construct a polynomial time machine which takes any one-way function (a random oracle) and uses it to compute a one-way permutation. We are interested in the stronger result that it would be hard to prove a one-way permutation exists given that a one-way function exists. A proof might be non-uniform. For example, a proof might produce a one-way function which is a permutation on all but finitely many lengths, and then conclude a one-way permutation exists.

Thus, we are motivated to prove the a stronger version of the above result. We wish to show $P = NP$ implies that there are no one-way permutations relative to a random oracle.

If we assume to the contrary that there are one-way permutations in a random world, then there must be some oracle $PPTM$ which computes a one-way permutation in some positive measure ϵ of worlds. (There are only countably many oracle $PPTM$s, if they all computed a one-way permutation on a measure zero of oracles, then almost all oracles would have no machine computing a one-way permutation.) Let M be an oracle $PPTM$ which computes a one-way permutation on a positive measure of oracles. By the Lebesgue Density Theorem, we can make a machine M' which has a finite number of oracle answers hardwired, and computes a one-way permutation on $1 - \delta$ of oracles (For any $\delta > 0$ we choose).

The above theorem says that, if $P = NP$, there is no M' for which $\delta = 0$. To contradict our assumption that there are one-way permutations in a random world, we need to prove the same result for some $\delta > 0$.

9.5 Applying the conjecture.

Assume the conjecture is true. There are particular constants ϵ, δ which make it work. Rename δ as β. Pick an $\alpha < \epsilon$ s.t. $\sqrt{\alpha} < \beta$. Thus, α, β still work as constants in the conjecture and $\sqrt{\alpha} < \beta$.

Theorem 9.2 (Assuming conjecture) *If Π is an oracle $PPTM$ which computes a permutation on $1-\alpha$ oracles, then there is a polynomial query complexity* *algorithm which, for each x, will compute $\Pi^{-1}(x)$ on $1-\sqrt{\alpha}$ oracles.*

Proof: Fix an x. Let E be any finite set of oracle queries and answers. A *proof* that $\Pi(y) = x$ for some oracle O is the pair $< y$, the set of queries and answers Π asks O during the computation$>$. If an oracle agrees with the query answer pairs contained in a proof, it is said to be *consistent* with that proof. Let T_E be the set of proofs that $\Pi(y) = x$ with some oracle consistent with E where all the query answer pairs in E are deleted from the proofs. Notice that if each possible query is considered as a variable, then each query answer pair can be considered as a variable or the negation of that variable. Furthermore, the query answer part of each proof can be considered a term (as in conjecture). Observe, each oracle on which Π computes a permutation is consistent with only one proof in $T_\emptyset$. Therefore, $U_{T_\emptyset} > 1-\alpha$. Call an oracle *good* if there is exactly one proof in $T_\emptyset$ which is consistent with that oracle. Otherwise, call an oracle *bad*. Consider any decision tree for asking oracle queries. A node of the tree is *bad* if it is the first node along its path from the root such that the set E of oracle queries and answers learned so far has the property $U_{T_E} < 1-\sqrt{\alpha}$. The critical observation is that at most a $\sqrt{\alpha}$ fraction of oracles will cause us to hit a bad node in the decision tree. (At least a $\sqrt{\alpha}$ fraction of oracles which hit a particular bad node are bad oracles. No oracle hits two bad nodes. Thus, if there were more than a $\sqrt{\alpha}$ fraction of oracles which hit a bad node, we would contradict the fact that there is less than an α fraction of bad oracles.)

Now for the algorithm. At any point in the algorithm, E will denote the finite set of query answer pairs we have gotten from asking queries of O. (E starts as the empty set.) Notice that the proofs in T_E contain only queries we have not asked so far. We proceed as follows: If T_E is a singleton, output the indicated y such that $\Pi(y) = x$. If $U_{T_E} \leq 1-\sqrt{\alpha}$, output "failure". Otherwise, pick a proof p such that $P_{N(p)} > \beta$. Ask O all the queries in p. (Update E.)

Repeat.

By the conjecture, as long as we have $U_{T_E} > 1 - \sqrt{\alpha}$, there will be a proof p such that $P_{N(p)} > \beta$. By the critical observation, only a $\sqrt{\alpha}$ fraction of oracles will ever cause our algorithm (decision tree) to reach a state where $U_{T_E} \leq 1 - \sqrt{\alpha}$. Thus, our algorithm will output "failure" on at most a $\sqrt{\alpha}$ fraction of oracles.

We will argue that the number of proofs asked is no more than a polynomial. Define Q_E as follows: Average the contribution over all oracles, where each oracle not consistent with E contributes zero, each bad oracle contributes zero, and each good oracle consistent with E contributes the number of queries in its unique proof but not in E. Because no proof is longer than $poly(\|x\|)$, $Q_\emptyset < poly(\|x\|)$. Observe, if we reach a stage where $Q_E = 0$, the algorithm will terminate. Each time the algorithm asks a proof p, we know two properties: $P_{N(p)} > \beta$ and $U_{T_E} > 1 - \sqrt{\alpha}$. Therefore, we decrease Q_E by at least $\beta - \sqrt{\alpha}$: Of the oracles consistent with E, at least β of them intersect p in at least one query (none of which are in E); at most $\sqrt{\alpha}$ of those oracles are bad. Therefore, the number of stages we can have without termination is bounded by $poly/(\beta - \sqrt{\alpha})$.

■

Theorem 9.3 (Assuming conjecture) $P = NP \Longrightarrow$ *If* Π *is an oracle PPTM which computes a permutation on* $1 - \alpha$ *oracles, then there is a polynomial time algorithm which, for each* x*, will compute* $\Pi^{-1}(x)$ *on* $1 - \sqrt{\alpha}$ *oracles.*

Proof: Each time the above algorithm picks a proof p such that $P_{N(p)} > \beta$,a polynomial time algorithm can choose a proof p in T_E such that $P_{N(p)} > \beta/2$: Approximate counting is in the polynomial time hierarchy[Sto83], and hence in P (Recall we assume $P = NP$). Thus, there is a polynomial time predicate that will say "yes" to all p such that $P_{N(p)} > \beta$, and will say "no" to all p such that $P_{N(p)} < \beta/2$. Using self-reducibility, we can find a proof which satisfies this predicate. (One must exist since there is a p such that $P_{N(p)} > \beta$.)

Assuming α was chosen sufficiently small that $\beta/2 > \sqrt{\alpha}$, we will only repeat this selection process $poly/(\beta/2 - \sqrt{\alpha})$ times (argue as above). Thus, we require only polynomial time in all.

If we come to a stage where there is no p which satisfies our predicate then the above algorithm must have output "failure". In this case, the polynomial time algorithm should also output "failure". ∎

Theorem 9.4 (Assuming conjecture) $P = NP \Longrightarrow$ *there is no one-way permutation relative to a random oracle.*

Proof: Assume not. Pick a $\lambda < \min(1/16, \alpha)$. By the remarks at the end of section 9.4, there is a Π which computes a one-way permutation on a $1 - \lambda$ fraction of oracles. The polynomial time algorithm above will invert Π on any given input for all but a $\sqrt{\lambda}$ fraction of oracles. By the pigeonhole principle, there are $1 - \sqrt{\sqrt{\lambda}}$ oracles for which the are infinitely many n where Π will be inverted on all but a $\sqrt{\sqrt{\lambda}}$ fraction of inputs of length n. This violates the definition of one-way permutation given in section 4.2. Thus, we have a contradiction when $\sqrt{\lambda} + \sqrt{\sqrt{\lambda}} < 1$. This is guaranteed by $\lambda < 1/16$. ∎

Corollary 9.3 (Assuming conjecture) *There exists an oracle relative to which one-way functions exist, but one-way permutations do not.*

Proof: Construct the oracle as follows: The even numbers form an oracle for PSPACE, the odd numbers form a random oracle. To see this oracle works, mimic the proof of Corollary 8.1. ■

10 Conclusions.

Previous results in cryptography had the form: From assumption X we are able to prove Y. We have introduced a strategy for obtaining a new type of result; we explain why certain assumptions are not sufficient to obtain a desired consequence. In particular, we have studied two problems which have frequently frustrated researchers in cryptography. Assuming our conjecture is correct, we have given evidence that the existence of one-way function is not a sufficently powerful assumption from which to prove the existence of a one-way permutation. Moreover, we have given evidence that a one-way permutation is not a sufficiently strong assumption from which to prove the existence of a secure secret key agreement protocol. We have observed that this last result has many natural similar corollaries.

References

[AD86] D. Aldous and P. Diaconis. Strong uniform times and finite random walks. Technical Report 59, Department of Statistics, Berkeley, Feb 1986.

[AHU74] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.

[BCC87] G. Brassard, D. Chaum, and C. Crepeau. Minimum disclosure proofs of knowledge. Technical Report PM-R8710, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, 1987.

[Ben87] J. Cohen Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, Sept 1987. YALEU/DCS/TR-561.

[BG81] C.H. Bennett and J. Gill. Relative to a random oracle A, $P^A \neq NP^A \neq$ co-NP^A with probability 1. *Siam Journal of Computing*, 10:96–113, 1981.

[BI87] M. Blum and R. Impagliazzo. Generic oracles and oracle classes. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, 1987.

[Blu81] M. Blum. Three applications of the oblivious transfer: Part i: Coin flipping by telephone; part ii: How to exchange secrets; part iii: How to send certified electronic mail. Department of EECS, University of California, Berkeley, CA, 1981.

[Blu82] M. Blum. Coin flipping by telephone: A protocol for solving impossible problems. In *Proceedings of the 24th IEEE Computer Confer-*

ence (CompCon), pages 133–137, 1982. reprinted in *SIGACT News*, vol. 15, no. 1, 1983, pp. 23–27.

[Bra] G. Brassard. An optimally secure relativized cryptosystem. *Advances in Cryptography, a Report on CRYPTO 81*, Technical Report no. 82-04, Department of ECE, University of California, Santa Barbara, CA, 1982, pp. 54–58; reprinted in *SIGACT News* vol. 15, no. 1, 1983, pp. 28–33.

[Bra81] G. Brassard. A time-luck tradeoff in relativized cryptography. *Journal of Computer and System Sciences*, 22:280–311, 1981.

[Bra83] G. Brassard. Relativized cryptography. *IEEE Transactions on Information Theory*, IT-19:877–894, 1983.

[CKS81] A.K. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *JACM*, 28:114–133, 1981.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, 1976.

[Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*, volume I. John Wiley & Sons, New York, third edition, 1968.

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of Advances in Cryptography*. CRYPTO, 1986.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for proto cols with honest majority. In *Proceedings of the 19th Annual Symposium on Theory of Computing.* ACM, 1987.

[HH87] J. Hartmanis and L.A. Hemachandra. One-way functions, robustness, and the non-isomorphism of np-complete sets. Cornell University DCS TR86-796, 1987.

[IY87] R. Impagliazzo and M. Yung. Direct minimum-knowledge computations. In *Proceedings of Advances in Cryptography.* CRYPTO, 1987.

[JVV86] Mark Jerrum, Leslie Valiant, and Vijay Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[LR86] M. Luby and C. Rackoff. How to construct pseudo-random permutations from pseudo-random functions. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, 1986.

[Mer78] R. C. Merkle. Secure communications over insecure channels. *CACM*, 21(4):294–299, April 1978.

[Nao] M. Naor. Personal communication.

[NY] M. Naor and M. Yung. Personal communication.

[Rab81] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard University, 1981.

[Sip85] Michael Sipser. Lecture notes in complexity theory. Manuscript, MIT, 1985.

[Sto83] L. Stocmeyer. The complexity of approximate counting. In *Proceedings of the 15th Annual Symposium on Theory of Compu ting.* ACM, 1983.

[Tar] Gabor Tardos. Query complexity, or why is it difficult to separate $NP^A \bigcap coNP^A$ from P^A by random oracles A? Manuscript. Eotvos University, Budapest.

[Yao82] A.C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE, 1982.