Limits on the Provable Consequences of One-way Permutations

Russell Impagliazzo[*]
Computer Science Division
University of California at Berkeley
Berkeley, California 94720

Steven Rudich[†]
Computer Science Department
University of Toronto
Toronto, Canada M5S 1A4

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 permutation 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. 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."

1 Introduction

A typical result in cryptography will be of the form: With assumption X, we can prove that a secure protocol for task P is possible. Because the standard crypto-

[*]Research partially supported by NSF grant CCR 88-13632.
[†]Research partially supported by NSF grant CCR 88-13632 and an IBM doctoral fellowship.

graphic assumptions are, at present, unproved, many results focus on weakening the assumptions needed 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 task 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 [DH76], [GM84]. However, researchers have been frustrated by unsuccessful attempts to base it on the weaker assumption that one-way permutations exist.

We provide strong evidence that it will be difficult to prove that secure secret-key agreement is possible assuming only that a one-way permutation exists. We model the existence of a one-way permutation by allowing all parties access to a randomly chosen permutation oracle. A random permutation oracle is provably one-way in the strongest possible sense. We show that any proof that secure secret-key agreement is possible in a world with a random permutation oracle would simultaneously prove $P \neq NP$. (Formally, $P = NP$ implies there is no secure secret-key agreement relative to a random permutation oracle.) We conclude that it is as hard to provably base a secure secret-key agreement protocol on an arbitrary one-way permutation as it is to prove $P \neq NP$. Furthermore, we can use the above result to construct an oracle relative to which one-way permutations exist, but for which secure secret-key agreement is impossible. (This oracle O is constructed by starting with an oracle for which P=NP, and adding on a random permutation oracle.) This means that any proof that the existence of a one-way function implies that of a secure secret-key agreement protocol cannot relativize (i.e., hold relative to any oracle). Non-relativizing proofs are few and far between not only in cryptography, but in complexity theory as a whole. Since the technique of examining complexity relative to an oracle was introduced in [BGS75], relativization results have been used to provide evidence for the difficulty of resolving questions in complexity theory [BG81]. (We will later briefly discuss the possibility that a non-relativizing proof basing secure secret-key agreement on a one-way permutation can be found.) Relativized complexity has not been frequently used in cryptography ([Bra, Bra83] is one exception to this rule); we hope the framework developed here will have wide applicability in separating the strengths of cryptographic assumptions.

Our result also has some implications for "black box" reductions between various other cryptographic assumptions and tasks. Instead of formalizing the notion of "black box" reducibility to an assumption or task, which would involve going into the specifics of these assumptions and tasks, we will use the phrase "A is black-box reducible to B" as an abbreviation for "If B holds relative to an oracle O, then A also holds relative to O". (In [I88], a general notion of "black box" proof is developed, and shown to be basically equivalent to that given in the preceding.) Since the definitions of the cryptographic tasks and assumptions mentioned here are lengthy, technical, and often not unique, to describe them formally would require a separate paper. (In fact,

A	B
One-way permutations exist	Secret-key agreement is possible
Signature schemes exist	Oblivious transfer is possible
Pseudo-random generators exist	Trapdoor functions exist
Private-key cryptosystems exist	Voting Schemes exist
Telephone coin flipping is possible	
Bit commitment with strong receiver is possible	
Bit commitment with strong sender is possible	
Collision free functions exist	

papers have been written describing various ways of formalizing some of the terms used here; for others, such papers do not presently exist but are greatly needed.) Therefore, rather than attempt to define these terms here, we will give references to papers introducing these concepts and/or papers clarifying them. We hope the reader not familiar with cryptography will still be able to follow the general idea of the following discussion.

Cryptographic tasks to be discussed here include: coin flipping by telephone ([Blu82]), electronic signatures ([DH76] , [GMR84]), private-key cryptography ([GM84, GGM84, LR86, Rac88]), bit-commitment (both the strong committer version ([GMW87]) and the strong receiver version ([BCC87])), identification ([DH76], [FFS86]), electronic voting ([Ben87]), oblivious transfer ([Blu81, Rab81]), and secret-key exchange itself ([DH76, Mer78]). General assumptions which have been used in cryptography include the existence of : one-way permutations ([P74]), pseudo-random generators ([BM84, Yao82]), trap-door permutations ([DH76]), and two-to-one collision-free functions. This last is a function f which is easily computable, is two-to-one on strings of length n, and where no polynomial-time algorithm, given n, can find strings x and y of length n with $f(x) = f(y)$.

Many reductions between the various assumptions and tasks listed above are known. In particular, it is known that the existence of a one-way permutation implies the following: pseudo-random generators exist ([Yao82]), private-key encryption is possible ([GM84, GGM84, LR86]), strong committer bit commitment is possible ([Yao82, GMW87]), telephone coin flipping is possible ([Blu82]), and electronic signatures are possible ([NY]). All of the preceding results relativize. We construct an oracle O relative to which one-way permutations exist, but for which no secret-key agreement protocol is possible. From relativized versions of these results, it follows that O will also have the property that, relative to O, pseudo-random generators exist, strong committer bit commitment is possible, etc. Thus, none of the preceding assumptions can imply that secure secret-key agreement is possible in a way which relativizes.

Furthermore, we can add to this list several statements which are not known to follow from the existence of a one-way permutation, but which O can be proved to satisfy because a truly random permutation is used in O's construction. For example, it is unknown whether a one-way permutation can be used to construct a two-to-one collision-free function, but it is easy to see that for a random permutation p, the function which outputs all but the last bit of p will be such a function. [NY] show that the existence of a two-to-one collision-free function suffices to construct a protocol for strong receiver bit commitment. Thus, neither the existence of two-to-one collision-free function nor that of a strong receiver bit commitment protocol suffices to construct a secret-key agreement scheme via a relativizing proof.

Similarly, if an assumption is sufficient to prove the possibility of secret-key agreement in a relativizing manner, it itself cannot be proven from the existence of a one-way permutation via a "black box" reduction. Examples of such assumptions include oblivious transfer ([Blu81, Rab81]), voting ([Ben87]), and trap-door functions ([DH76, GM84]).

To summarize:

There is an oracle O relative to which all A's hold, but all B's do not. (See above table.)

Some caution is needed in interpreting these results, since at least one non-relativizing construction in cryptography is known. In [I88] it is shown that the theorem proved in [GMW87], "the existence of a one-way permutation implies the existence of zero-knowledge protocols for all languages in NP", fails with respect to a random permutation. In contrast, the [FFS86] construction of an identification protocol based on any one-way function will *not* be possible with just a black box for a random permutation. Their construction is as follows. Every person chooses a random x, and announces publicly $f(x)$ as their I.D. To prove you are the person with I.D. I, you give a zero-knowledge proof of knowledge that you know an x with $f(x) = I$. However, to give a zero-knowledge

proof as in GMW and FFS that you know such an x, it is necessary to have an actual circuit that computes f, not just a black box which gives the value of f. In fact, if f is a random permutation, no such zero-knowledge proof will be possible. Thus, the [FFS86] scheme does not relativize. The [FFS86] protocol is exceptional even for those constructions involving zero-knowledge proofs. Most applications of zero-knowledge will in fact relativize, even though the literal statement of the [GMW87] theorem does not. In a world with a random permutation oracle, it *is* possible to give a zero-knowledge proof for any property *actually* in NP, as opposed to NP relativized to this oracle. It is only applications which attempt to "bootstrap", proving things concerning the values of the same function used to make the protocol zero-knowledge, which fail to relativize.

The above example is the only non-black box construction in cryptography known to the authors for a result in a general form (as opposed to results involving specific crypto-systems). Thus, it is fair to say that the result presented here shows that most of the standard techniques in cryptography cannot be used to construct a secret-key exchange protocol from a one-way permutation.

2 Notation and definitions

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.)

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.

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.

We will use the following 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.)

3 Uniform Generation

3.1 Polynomial-time relations

A relation, R, is polynomial-time if we can decide xRy in time polynomial in $\|x\| + \|y\|$. In this paper, 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.

3.2 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.

3.3 $P = NP$ and uniform generation

Theorem 3.1 (JVV) *For any polynomial-time relation, there exists a $PPTM$ equipped with a $\sum_2^P$ oracle that uniformly generates it.*

Theorem 3.2 $P = NP \Longrightarrow$ *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 3.1 $P = NP \Longrightarrow$ *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 3.2 $P = NP \Longrightarrow$ *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. ■

3.4 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 3.3 $P = NP \Longrightarrow$ *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 3.2, 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. ■

4 Random Oracles

4.1 Random function oracles

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 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 state without proof a theorem a standard theorem concerning random functions:

Theorem 4.1 *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.*

4.2 Random oracles and uniform generation

Theorem 4.1 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. Our goal is, assuming $P = NP$, to break secret-key exchange in a random world. (In theorem 3.3, 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.

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 4.2 $P = NP \Longrightarrow$ *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 3.2, 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 4.3 $P = NP \Longrightarrow$ *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 3.2 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 4.4 *$P = NP \Longrightarrow$ 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. ∎

5 Random Permutation Oracles

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.

¿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 5.1 *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 $D_{T_n} < q^2/2^n$. ∎

The above theorem will allow us to first prove our results relative to a random oracle, and then extend them to a random permutation oracle.

It is a standard theorem that random permutations are very hard to invert.

Theorem 5.2 *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.*

6 Cryptographic Lower Bounds

6.1 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 5.2, 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 is 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 5.1 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.

6.2 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.

6.3 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.

6.4 Eve's sample space

We need to define the probability distributions Eve samples from during her algorithm. They have already been described in section 4.2, Theorem 4.4. 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>.

6.5 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 4.4, Eve picks a random run of Alice from the space $A_{E_{r,i-1}}^{C_r}$. (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: $BPQ_{r,i} = BPQ_{r,0} - E_{r,i}$.

6.6 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.

6.7 The efficacy of Eve's algorithm

Theorem 6.1 *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 6.1, 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 6.1 *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 $1 - S_l/n$.

The proof of this lemma can be found in STOC'89

Theorem 6.2 *Theorem 6.1 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 5.1. ∎

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 6.3 *$P = NP \Longrightarrow$ 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

6.1 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 6.1 *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 we 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 6.4 *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.

The proof of this theorem can be found in STOC'89

7 Related Work and Open Problems

In the work presented here, as in much of theoretical cryptography, we do not go into exactly how much time the adversary will take to break the protocol, as long as this time is polynomial. However, in real life, a protocol taking a large degree polynomial time to break may be almost as good as one secure against any polynomial time adversary. Merkle[Mer78] has suggested a protocol, based on any one-way function, the breaking of which would require an eavesdropper to take time quadratic in the time taken by the participants. (Here, time is measured as the number of calls to a black box for the one-way function.) We showed that for a protocol in normal form, an eavesdropper can always break the protocol in time $O(n^3 \log n)$; however, to put the protocol into normal form may square n, so our eavesdropper is actually taking time $O(n^6 \log n)$. This leaves open Merkle's question of whether his scheme is optimal.

Another general question brought up by this research is whether similar statements can be proved for other cryptographic applications. We have previously given a list of applications at least as strong as secret key agreement; that these are unlikely to be a consequences of the existence of a one-way permutation follows from the result here. However, it would be interesting to show that there is some natural application which cannot even be based on a much stronger assumption, such as the existence of a trapdoor permutation.

8 Acknowledgements

We are especially grateful to Manuel Blum and Amos Fiat, who asked us the question of whether a one-way permutation suffices for secret agreement, and presented a model in which it might be disproved. We would also like to thank Noam Nisan, Charlie Rackoff, and Umesh Vazirani. We give Manuel and Umesh a second helping of thanks for their support and encouragement during the many times we found a fatal flaw in what we thought was a proof.

References

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=NP question. SIAM J. Comp., 4 (1975) pp. 431-442.

[BG81] C. H. Bennett and J. Gill. Relative to a random oracle A, $P^A neNP^A neCo - NP^A$ with probability 1. SIAM J. Comp. 10 (1981)

[BCC87] G. Brassard, D. Chaum, and C. Crépeau. 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.

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

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comp. 13 (1984) pp. 850–864

[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.

[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.

[FFS86] U. Feige, A. Fiat and A. Shamir. Zero-knowledge proofs of identity. STOC, 1987.

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. In *Proceedings of the 25th Annual Foundations of Computer Science.* ACM, 1984.

[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.

[GM84] S. Goldwasser and S. Micali. Probabalistic Encryption. *JCSS*, 28:270–299, 1984.

[GMR84] S. Goldwasser, S. Micali, and R. Rivest. A "paradoxical" solution to the signature problem. In *Proceedings of the 25th Annual Foundations of Computer Science.* ACM, 1984.

[I88] R. Impagliazzo Proofs that relativize, and proofs that do not. Unpublished manuscript, 1988.

[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.

[NY] M. Naor and M. Yung. Universal One-Way Hash Functions and Their Applications. These precedings.

[P74] G. P. Purdy A high security log-in procedure. *CACM,* 17:442-445, 1974.

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

[Rac88] C. Rackoff. A basic theory of public and private cryptosystems. Crypto 88.

[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.