# The Use of Interaction in Public Cryptosystems.

(extended abstract)

Steven Rudich
School of Computer Science
Carnegie Mellon University
rudich@cs.cmu.edu

Abstract

For every k, we construct an oracle relative to which secret agreement can be done in k passes, but not in k-1. In particular, for k=3, we get an oracle relative to which secret agreement is possible, but relative to which trapdoor functions do not exist. Thus, *unlike* the case of private cryptosystems, there is no black box reduction from a k-pass system to a k-1 pass system. Our construction is natural– suggesting that real-world protocols could trade higher interaction costs for an assumption strictly weaker than the existence of trapdoor functions. Finding a complexity theoretic assumption necessary and sufficient for public cryptosystems to exist is one of the important open questions of cryptography. Our results make clear the possibility that this question is impossible to answer because it contains a false hidden assumption: the existence of a 2-pass public cryptosystem follows from the existence of a k-pass system. The question should really be a family of questions: given k find an assumption equivalent to the existence of a k-pass public cryptosystem.

## 1 Introduction

An important project in cryptography is to classify protocols according to the complexity theoretic assumptions that are necessary and sufficient to guarantee their existence. This classification not only lends a structure and coherence to the field, but bases the viability of cryptography on the most general possible assumptions (as opposed to those involving specific problems such as factoring). An excellent example of this type of result is that private-key cryptography (under any reasonable definition) is possible if and only if one-way functions exist[ILL89, IL89]. This example leads one to seek a corresponding result for public cryptosystems (being able to send a secret message to a person with whom you share no secret information). Researchers, however, have been unsuccessful in finding a natural, complexity-theoretic assumption that is equivalent to the existence of public cryptosystems. We shed new light on the difficulty of this problem; in the process, we solve some of the main open problems posed in "A Basic Theory of Public and Private Cryptosystems"[Rac88].

Following Rackoff's example, we consider private-key cryptography as the ability to send secret messages to a person with whom you share a secret string; public cryptosys-

J. Feigenbaum (Ed.): Advances in Cryptology - CRYPTO '91, LNCS 576, pp. 242-251, 1992.

tems as the ability to send secret messages to a person with whom you share no secret information. In neither type of cryptography, do we include a constant bound on the number of rounds of interaction as part of the definition. Thus, we call it "public cryptography" and not "public-key cryptography" as defined by [DH76]. (From now on, we measure the amount of interaction in terms of passes rather than rounds, e.g., Alice to Bob to Alice is two passes, but only one round.) Although all known protocols for public cryptosystems[DH76, RSA78, GM84] can be easily altered to use no more than two passes, there is no a priori reason to suspect this will always be so. We consider secret agreement as the ability to agree on a secret bit with a person with whom you share no secret information. Under our definitions, secret agreement and public cryptosystems are equivalent; if one is possible so is the other. We will use them interchangeably.

We show that for any $k \geq 2$ there exists an oracle relative to which public cryptosystems are possible in k passes, but not in k-1 passes. Furthermore, the internal structure of the oracle is not an unnatural construct achieved through diagonalization: *it uses a random function which shares features with the discrete-logarithm problem, used by Diffie and Hellman[DH76] in their original protocol for public-key cryptography.* Our result applies directly to Rackoff's problem of finding "a convincing example where extra passes appear to help" achieve secure public cryptosystems. In contrast, Rackoff's problem for private-key cryptography was resolved oppositely. It has been shown that a k-pass private-key cryptosystem exists if and only if a one pass private-key cryptosystem exists[ILL89, IL89]; furthermore, these results have the form of standard *black box* reductions. Our result rules out any black box reduction for public cryptosystems. We think this difference is counter-intuitive. Our oracle is natural and is intended to reflect real world cryptography.

Black box reductions are quite natural to cryptography. Unlike the situation in complexity theory, a black box reduction from A to B is the preferred result. This is because cryptographers ultimately want to implement their work. The reductions should be effective, simple, and modular, e.g., black box. Thus, cryptographers seek other kinds of reductions only as a last resort. Black box reductions are the only type possible when assumption A is physical (e.g., envelopes).

Setting $k = 3$, we get an oracle, $\Gamma$, relative to which three pass public cryptosystems exist, but two pass systems do not. Oracles separations of this kind rule out the black box arguments used in all known cryptographic reductions among protocols which pass information from one party to another (as opposed to zero-knowledge protocols). One interpretation of this result is that the existence of public cryptosystems is not provably equivalent to any complexity-theoretic assumption that in turn implies the existence of a 2-pass cryptosystem. For example, all the notions of a "trapdoor" function

or "trapdoor" permutation appearing in the literature[DH76, RSA78, Yao82] imply the existence of a 2-pass public-key cryptosystem. Thus, relative to $\Gamma$, public cryptosystems are possible, but trapdoor functions do not exist; trapdoor functions are not a necessary consequence of public cryptosystems. In the other direction, previous work by Impagliazzo and Rudich[IR89] has shown similar evidence that the existence of a one-way permutation is not sufficient for the existence of public cryptosystems. Combining these results, we are at a loss for a good candidate for a complexity-theoretic assumption characterizing the nature of public cryptography. Perhaps the requirements of public cryptosystems have no natural expression in complexity theory (again, in contrast to private-key cryptography).

The usual protocol design philosophy minimizes the amount of interaction. However, current public cryptosystems are based on specific number theoretic problems (e.g. factoring, discrete-log) which might someday be broken. Our results suggest that public cryptosystems could trade higher amounts of interaction for weaker assumptions less likely to be broken. In particular, we justify the following open problem: Find a 3-pass system based on an assumption which seems to be weaker than the existence of trapdoor functions.

The proof requires a very precise analysis of the information possessed by Alice and Bob at each step. After concluding that the flow of information in a two pass protocol has a restricted form, we show any protocol with this form is insecure. This requires extending the techniques in [IR89].

## 2 Overview of the oracle for three passes versus two

The oracle is inspired by the Diffie-Hellman protocol[DH76]: Alice picks a prime $p$ and a random $a < p$. Alice picks a random $x$, Bob picks a random $y$. Alice sends $a, p$, and $a^x \bmod p$. Bob sends $a^y \bmod p$. Alice computes $(a^y)^x \bmod p$. Bob computes $(a^x)^y \bmod p$. Both parties arrive at the same result. The eavesdropper, Eve, has only seen $a, p, a^x \bmod p$, and $a^y \bmod p$; the only known way for Eve to compute $a^{xy}$ is to compute the discrete-log. The salient feature of this protocol is that commutativity allows Alice and Bob to arrive at the same result by a different calculation.

The oracle will contain two random functions, $\xi$ and $DH$. Function $\xi$ takes an input and returns a random string three times its length. With probability 1, $\xi$ is a 1-1 function for sufficiently large input strings. (Moreover, $\xi$ is the same "annihilating" function defined by Kurtz, Mahaney, and Royer[KMR89]. We will use this fact later.) The function $DH$ is designed to maintain the following property: $DH(x, \xi(y \cdot \xi(x))) =$

$DH(y,\xi(x\cdot\xi(y)))$. ($\cdot$ means concatenation.) Apart from this property, $DH$ is a random function, i.e. we know the above pair is equal to the same random bit. (We also include a PSPACE-complete portion of the oracle to insure that protocols which do not use the random functions portion of the oracle are not secure. For example, we want to be sure no two-pass protocol based on factoring or discrete-log could be secure.)

Using this oracle the following three pass protocol is secure: Alice picks a random $x$, Bob picks a random $y$. Alice sends $\xi(x)$. Bob sends $\xi(y)$ and $\xi(y\cdot\xi(x)))$. Alice sends $\xi(x\cdot\xi(y)))$. Alice computes $DH(x,\xi(y\cdot\xi(x)))$. Bob computes $DH(y,\xi(x\cdot\xi(y)))$. They both arrive at the same result.

Why is there no 2-pass protocol relative to this oracle? The precise answer is quite technical. Very generally, we show that no two pass protocol can use the oracle to arrive at the same result by different calculations, i.e., the queries to the oracle that turn out to be useful are exactly those that both parties query. Once this is shown we can use the techniques developed in [IR89] to break the protocol.

More precisely, a *dual pair* is a pair of queries to the $DH$ portion of the oracle having the form $\{(x,\xi(y\cdot\xi(x))),(y,\xi(x\cdot\xi(y)))\}$. The fact that $\xi$ is 1-1 (for suff. large inputs) insures that any two distinct dual pairs are disjoint (for suff. large inputs). A protocol *hits* a dual pair if each query in the pair is made by at least one of the participants. We show that at the moment a two pass protocol hits a dual pair, one of the two participants has already queried both $x$ and $y$. Thus, he/she can calculate both members of the dual pair. This means that one of the two calculations is the same as the one done by his/her partner (in the language of [IR89], an intersection query). The oracle is not being used to arrive at the same result by a different calculation from one's partner. The situation is not substantively different from using a random function oracle with a PSPACE-complete oracle. It has already been shown that relative to such an oracle secure secret agreement (public cryptosystems) is impossible[IR89].

# 3 Notation and definitions

We will abbreviate probabilistic polynomial-time Turing machine with the notation $PPTM$. We use the notation *poly* to refer to some polynomial function. Thus, we can use the freewheeling arithmetic $poly * poly = poly$. By $x\cdot y$ we mean the concatenation of the strings $x$ and $y$. The three oracle $PPTM$s (alice, Bob, and Eve) which we will consider will all have access to the same fixed oracle.

A *secret agreement protocol* is a pair of oracle $PPTM$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 parameter $l$ written in unary on their input tapes; Alice and Bob run, communicating via the common tape; Alice and Bob both write a bit on their secret tapes. Alice and Bob run in time polynomial in $l$. ($l$ is often called the *security parameter.*) If this bit is the same, Alice and Bob are said to *agree.* It should be noted that a protocol which can agree on a bit can be run multiple times to agree on a string; we will consider only protocols which agree on a single bit. 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 the same bit.

An oracle $PPTM$ Eve *breaks* a secret agreement protocol if Eve, given only $l$ and the conversation, can guess the bit with probability greater than $1/2 + 1/poly(l)$. Eve can only use time bounded by a polynomial in $l$. A protocol is *secure* if no Eve can break it.

# 4 Construction of the oracle for $k = 3$

Let $\xi$ be a random function which takes any input string to a string of independent random bits three times the length of the input string. Let $\mu$ be random function from unordered pairs of strings to a random bit. To any particular choice of $\xi$ and $\mu$ we can associate a binary function $DH$ defined by the following property (assuming $\xi$ is 1-1):

$$DH(p, \xi(q \cdot \xi(p))) = DH(q, \xi(p \cdot \xi(q))) = \mu(\{p, q\})$$

$DH$ is undefined everywhere else. Of course, $\xi$ might not be 1-1. But with probability one, $\xi$ fails to be 1-1 for finitely many inputs[KMR89]. So with probability one, $DH$ is well defined for sufficiently large inputs; we can make all the small inputs evaluate to zero.

We construct a random $\Gamma$ as follows. Pick a random $\xi$ and $\mu$. Now construct an oracle with three parts:

- The function $\xi$.
- The function $DH$ which is associated with $\xi$ and $\mu$.
- A PSPACE-complete oracle.

By the above remarks, we know that with probability one, $\Gamma$ is well defined. It is important to note that when we say with probability one over random $\Gamma$, we mean over

random $\xi$ and $\mu$. This should not be confused with saying that $\Gamma$ is a random oracle. The results in this paper are not being proved relative to a random oracle.

# 5 A three pass public cryptosystem

We can use $\Gamma$ to acheive a three pass public cryptosystem which we call the *natural cryptosystem* associated with $\Gamma$. The following is a protocol for Alice and Bob to agree on a random bit:

Alice and Bob agree on a security parameter $l$. Alice picks a random $x$ of length $l$, Bob picks a random $y$ of length $l$. Alice sends $\xi(x)$. Bob sends $\xi(y)$ and $\xi(y \cdot \xi(x)))$. Alice sends $\xi(x \cdot \xi(y)))$. Alice computes $DH(x, \xi(y \cdot \xi(x)))$. Bob computes $DH(y, \xi(x \cdot \xi(y)))$. They both arrive at the same result.

**Theorem 5.1** *The natural cryptosystem associated with $\Gamma$ is correct and secure with probability one. (If Eve asks a polynomially bounded number of oracle queries, not even infinite computational power will help her guess their secret bit with probability greater than $1/2 + poly(l)/2^l$.)*

**Proof:** The protocol is clearly correct as long as $\Gamma$ is well defined. This happens with probability one.

For the security of the protocol it suffices to argue that knowing the conversation, i.e., $\xi(y)$, $\xi(x)$, $\xi(x \cdot \xi(y)))$, and $\xi(y \cdot \xi(x)))$, does not reveal any information about $x$ or $y$. The proof is standard and we omit it here. ∎

# 6 Three passes are required

In this section we argue that three passes are required for secure secret agreement relative to a random $\Gamma$.

## 6.1 Defining knowledge about the oracle.

It is possible to formalize the notion of an oracle $TM$ knowing certain facts about a random $\Gamma$. We say an oracle $TM$ $T$ *tried* $q$ at a certain point in its computation if it

makes a query to $\Gamma$ of the form: $\xi(*q*)$, $DH(*,q)$, or $DH(q,*)$ where * is any string. We say a protocol tried $q$ if either Alice or Bob tried $q$. We say a protocol queries a *dual pair* involving $p$ and $q$ if the protocol has made the queries: $DH(p,\xi(q\cdot\xi(p)))$ and $DH(q,\xi(p\cdot\xi(q)))$.

**Lemma 6.1** *With probability one, over random $\Gamma$, for any oracle PPTM $T$ using $\Gamma$, for sufficiently large $n$ and any string $q$ of length greater than or equal to $n$, the probability (over random tapes of $T$) that $T(n)$ tried $\xi(q)$ at time $t$ without having queried $\xi(q)$ at a time prior to $t$ is bounded by $poly(n)/2^n$.*

Proof description: The proof of lemma 3.4 in [KMR89] already proves the above result for a random oracle $\xi$ (as opposed to $\Gamma$). Their proof techniques can be generalized to $\Gamma$ because the $DH$ portion of the oracle that can be queried without first querying $\xi(q)$ is independent from $\xi(q)$.

This theorem can be equally well applied to protocols rather than to individual machines; a protocol can be simulated by a single $PPTM$.

## 6.2 Two pass protocols have restricted form

**Lemma 6.2** *Any run of a secret agreement which satisfies the following two properties must make at least three passes: 1) The protocol queries some dual pair involving $p$ and $q$, when neither party has tried both $p$ and $q$. 2) The protocol never tries $\xi(q)$ without first having queried $\xi(q)$.*

**Proof:** Suppose Alice made the query $DH(p,\xi(q\cdot\xi(p)))$ and Bob made the query $DH(q,\xi(p\cdot\xi(q)))$. Thus, Alice tried $p$, but not $q$. Bob tried $q$, but not $p$. The protocol tried $\xi(q\cdot\xi(p))$. By property 2, someone queried $\xi(q\cdot\xi(p))$; it must be Bob since it wasn't Alice who tried $q$. Thus, Bob tried $\xi(p)$. Symmetrically, Alice tried $\xi(q)$.

We can now reason about who tried what first. For example, before the protocol tried $\xi(p)$ it must have queried $\xi(p)$ (property 2). Thus either Alice or Bob queried $\xi(p)$; it must have been Alice lest Bob have tried both $p$ and $q$ (contradicting property 1). Thus Alice tried $p$ before Bob tried $\xi(p)$.

Alice tried $\xi(q \cdot \xi(p))$, but did not make that particular query lest Alice have already tried $q$. By property 1, that query was made previously. It must have been made by Bob. Before Bob could have made it, Bob must have tried $\xi(p)$. Thus, Bob tried $\xi(p)$ before Alice tried $\xi(q \cdot \xi(p))$.

Summarizing more succinctly: By property 2 and the definition of trying, Alice tried $p$ before Bob tried $\xi(p)$ before Bob queried $\xi(q \cdot \xi(p))$ before Alice tried $\xi(q \cdot \xi(p))$. Symmetrically, Bob tried $q$ before Alice tried $\xi(q)$ before Bob tried $\xi(p \cdot \xi(q))$. Alice and Bob are both involved in two separate passes. There must have been at least three passes in all. ∎

We say a protocol has *restricted form* if, with probability one, over random $\Gamma$, for sufficiently large $n$, the protocol will have probability greater than $1 - poly(n)/2^n$ of having a run in which each time the protocol queries a dual pair involving $p$ and $q$ of length greater than $n$, at least one of the two parties has tried both $p$ and $q$.

Combining lemmas 6.1 and 6.2 we get the following result:

**Lemma 6.3** *Any two pass protocol has restricted form.*

## 6.3 Two pass protocols can be broken by Eve

**Lemma 6.4** *With probability one, over random $\Gamma$, any protocol of restricted form can be broken by Eve.*

Proof description: In previous work by Impagliazzo and Rudich[IR89], it is shown that no protocol is secure relative to an oracle with just the $\xi$ and PSPACE-complete portions. Intuitively, a protocol of restricted form just can't make any other use of the $DH$ portion of the oracle besides extracting random bits from it; thus, such a protocol is no more secure than it would be using only the $\xi$ portion of the oracle.

[IR89] rely on the notion of an *intersection query*, a query to the oracle that is made by both Alice and Bob. They show that if Eve has figured out all the intersection queries that occurred before a given time, then Eve has a high probability of figuring out the next intersection query. By maintaining this inductive procedure Eve can figure out the final intersection query, namely, the secret itself.

If a protocol using $\Gamma$ has restricted form, we can assume without loss of generality when the protocol queries a dual pair the party that tried both $p$ and $q$ (with high

probability one of the parties tried this, unless the queries are very short and clearly useless) can ask both queries in the dual pair. Thus, Alice and Bob will intersect on this dual pair. Using the techniques Impagliazzo and Rudich, it can be shown that Eve can use the same inductive procedure on the intersection queries.

# 7 Putting it together

Combining lemmas 5.1 and 6.3 with the above theorem we get:

**Theorem 7.1** *With probability one over random $\Gamma$, a three pass public cryptosystem exists, but a two pass system does not.*

**Corollary 7.1** *There exists an oracle relative to which public cryptosystems exist, but trapdoor functions do not.*

This corollary should be contrasted with the result in [IR89] stating the existence of an oracle relative to which one-way permutations exists, public cryptosystems do not.

To generalize the oracle construction to $k > 3$, we simply have to change the definition of the $DH$ function portion of the oracle. For example, for $k = 4$, $DH$ is defined by the following property:

$$DH(p, \xi(q \cdot \xi(p \cdot \xi(q)))) = DH(q, \xi(p \cdot \xi(q \cdot \xi(p)))) = \mu(\{p, q\})$$

The proof techniques in this paper remain unaffected.

We get:

**Theorem 7.2** *For any $k \geq 2$, there exists an oracle relative to which $k$ pass public cryptosystems are possible, but $k$-1 pass systems do not.*

# 8 Acknowledgments

I would like to thank Charlie Rackoff, Jim Mcinness, and Russell Impagliazzo for helpful discussions.

# References

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

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[IL89] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE, 1989.

[ILL89] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random number generation from one-way functions. In *Proceedings of the 21st Annual Symposium on Theory of Compu ting*, ACM, 1989.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual Symposium on Theory of Compu ting*, ACM, 1989.

[KMR89] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle. In *Proceedings of the 21st Annual Symposium on Theory of Compu ting*, ACM, 1989.

[Rac88] Charles Rackoff. A basic theory of public and private cryptosystems. In *Proceedings of Advances in Cryptography*, CRYPTO, 1988.

[RSA78] R. Rivest, A. Shamir, and L. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications. ACM.*, 21(2):120–126, Feb 1978.

[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.