One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract)

Russell Impagliazzo[1] and Michael Luby[2]

TR-89-031

May 1989

Abstract

In much of modern cryptography, the security of a protocol is based on the intractability of a problem such as factorization of randomly chosen large numbers. The problems assumed intractable all have the same form; they are based on a one-way function, i.e. one that is easy to compute but hard to invert. This is not a coincidence. We show that for many cryptographic tasks any secure protocol for the task can be converted into a one-way function, and thus any proposed protocol for these tasks is implicitly based on a one-way function. Tasks examined here are chosen to cover a spectrum of cryptographic applications: private-key encryption, identification/authentication, bit commitment and coin-flipping by telephone. Thus, unless one-way functions exist, secure protocols for these tasks are impossible.

1 Department of Mathematics, University of California, Berkeley, California. Research partially supported by NSF grant CCR 88-13632.

2 International Computer Science Institute, Berkeley, California. On leave of absence from the University of Toronto. Research partially supported by NSERC of Canada operating grant A8092.

One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract)

Russell Impagliazzo*
Department of Mathematics
U.C. Berkeley
Russelli@ernie.berkeley.edu

Michael Luby†
International Computer Science Institute
Berkeley, California
Luby@icsi.berkeley.edu

1 Introduction

In much of modern cryptography, the security of a protocol is based on the intractability of a problem such as factorization of randomly chosen large numbers. The problems assumed intractable all have the same form; they are based on a one-way function, i.e. one that is easy to compute but hard to invert. This is not a coincidence. We show that for many cryptographic tasks any secure protocol for the task can be converted into a one-way function, and thus any proposed protocol for these tasks is implicitly based on a one-way function. Tasks examined here are chosen to cover a spectrum of cryptographic applications: private-key encryption, identification/authentication, bit commitment and coin-flipping by telephone. Thus, unless one-way functions exist, secure protocols for these tasks are impossible.

For some of these tasks we prove the stronger result that a secure protocol implies the existence of a pseudo-random generator. (In the non-uniform model of security, [Impagliazzo, Levin, Luby 89] shows that the existence of a one-way function and that of a pseudo-random generator are equivalent. In the uniform model of security this question is still open. The constructions here all work in both models of security.) This allows us to put protocols for these tasks into a "normal form"; i.e., given any secure protocol for the task, we can convert it into a secure protocol with a particular structure. For example, we show that any secure

*Research partially supported by NSF grant CCR 88-13632

†On leave of absence from the University of Toronto, research partially supported by NSERC of Canada operating grant A8092

protocol for private-key encryption, (even one with many communication rounds that is probabalistic and has some chance of error in decryption) yields a pseudo-random generator. Combining this with [Goldreich, Goldwasser, Micali 84] and [Luby, Rackoff 86], we have a way of converting any private-key encryption scheme into a "pseudo-random block cipher" (a provably secure DES-like function). This resolves an open problem stated in [Rackoff 88]. Similar results hold for identification and bit commitment versus a strong committer.

The model of cryptography we assume, the computational model, views participants and adversaries in a protocol as time-bounded probabilistic Turing machines (alternatively, circuits). Two other basic models of cryptography exist. The information theory model [Shannon 48] views participants and adversaries as having unlimited computational resources. Unfortunately, most cryptographic tasks, including those considered in this paper, are impossible in the information theory model (e.g. [Rackoff 85] shows that a version of private key encryption is not possible if $P{=}NP$). We informally conjecture that any cryptographic task that is not possible in the information theory model requires a one-way function in the computational model. A third model bases cryptosystems on the security of physical devices. [Bennett, Brassard, Breidbart, Weisner 82] introduce the notion of cryptosystems based on quantum physics. More recently, [Crepeau, Kilian 88] base certain protocols on physical devices such as a noisy channel. Our results have no relevance to models based on physical devices.

Notation : Let x and y be bit strings. Then, $|x|$ is the length of x, $x \circ y$ is the concatenation of x and y, x_i is the i^{th} bit of x and $x_{\leftarrow i}$ is the first i bits of x. If α is a number, then $|\alpha|$ is the absolute value of α.

2 Background

2.1 Security

In this paper we consider both uniform and non-uniform models of security. The difference between the two models of security is that, in the uniform model, the adversary is a fast algorithm, whereas, in the non-uniform model, the adversary is a small circuit. In this paper, the form of a result is that any secure protocol for a particular cryptographic task can be used to construct a one-way function, in either model of security. For all the tasks we consider, a pseudo-random generator can be used to construct a secure protocol for the task, in either model of security. In the non-uniform model of security, [Impagliazzo, Levin, Luby 89] show that any one-way function can be used to construct a pseudo-random generator. Thus, in the non-uniform model of security, the existence of a secure protocol is equivalent to the existence of a one-way function. However, in the uniform model of security, whether a pseudo-random generator can be constructed from any one-way function is still an open question (although it is known that a pseudo-random generator can be constructed from one-way functions with additional properties). Thus, whenever we can, we prove the stronger result that any secure protocol for the task can be used to construct a pseudo-random generator. Whenever this is possible we are allowed to conclude that the existence of a secure protocol is equivalent to the existence of a pseudo-random generator in both models of security.

Definition (resources) : A *resource class* R is class of functions from N to N that includes the identity function $r(n) = n$. If $r'(n) \leq r(n) \in R$ and $r'(n)$ is monotone then $r'(n), r'(n)+1$ and $r'(n^2)$ are all in R. For all $r \in R$ it is required that $\log_n(r(n)) < n$.

We use resource classes to parameterize the resources allowed for adversaries in breaking one-way functions, pseudo-random generators and other cryptographic tasks. For example, R might be all polynomial bounded functions or all functions bounded by $2^{O(\log^c n)}$ for some constant $c > 1$.

Definition (uniform and non-uniform) : A *uniform* algorithm is a probabilistic Turing machine. The *time bound* $T(n)$ for a uniform algorithm is the maximum running time on inputs of length n. A *non-uniform* algorithm is a pair of Turing machines A and M. A is a preprocessing algorithm that on input of length n produces a string $A(n)$. The running time of $A(n)$ is not limited. Algorithm M accepts as input x and $A(|x|)$. The *time bound* $T(n)$ for a non-uniform algorithm is maximum running time of M over all inputs x and $A(|x|)$. From the results of [Karp, Lipton 80], a non-uniform algorithm with time bound in resource class R is equivalent to a (recursive) family of circuits with the size of the circuit for inputs of length n bounded by a function $r(n)$ where r is in resource class R.

Definition (feasible) : A uniform (non-uniform) algorithm is *feasible* with respect to resource class R if the time bound function is in R.

Definition (negligible) : A function $p : N \rightarrow N$ is *negligible* with respect to resource class R if for all $r \in R$, for almost all n, $p(n) \leq 1/r(n)$.

For the remainder of the paper, unless stated otherwise, a feasible adversary algorithm is always with respect to an arbitrary but fixed resource class R. For each cryptographic task that we define, there are implicitly two definitions being made simultaneously, one with respect to the uniform and the other with respect to the non-uniform model of security. Unless otherwise stated, each definition, lemma, proposition and theorem has two versions, one in each model.

2.2 One-way functions

Intuitively, a function f is *one-way* if it is easy to compute but hard to invert, i.e. given x the value of $f(x)$ can be computed in polynomial-time but every feasible algorithm that receives as input $f(x)$ (when x is a randomly chosen string of length n) can output a y such that $f(y) = f(x)$ with only negligible probability. It has not yet been proven that one-way functions exist (if P=NP then they certainly do not exist, but even if $P \neq NP$ it is not clear that they exist), but there are many examples of functions that seem to be one-way in practice and that are conjectured to be provably one-way. Some examples of conjectured one-way functions are factoring a composite number N that is the product of large randomly chosen primes, square roots modulo such an N, discrete log modulo a large randomly chosen prime, problems from coding theory and the subset sum problem.

Notation (functions and probability ensembles) : A *length (function)* $l(n)$ is a monotone increasing function from N to N such that $l(n)$ is computable in time polynomial in n.

A *function* f with input length n and output length $l(n)$ specifies for each $n \in N$ a function $f_n : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ that is computable in polynomial time. For simplicity, we write $f(x)$ in place of $f_n(x)$. D is a *probability ensemble* if $D = \{D_n : n \in N\}$, where D_n is a probability distribution on $\{0,1\}^n$. We use the notation $x \in_D \{0,1\}^n$ to mean that x is randomly chosen from $\{0,1\}^n$ according to D_n. The Shannon entropy [Shannon 48] of D_n is given by $Ent(D_n) = -\sum_{x\in\{0,1\}^n} D[\{x\}] \cdot \log(D[\{x\}]$. The *uniform probability ensemble* U assigns to each positive integer n the uniform probability distribution U_n on bit strings of length n. When the context is clear, we simply say that x is chosen randomly when $x \in_U \{0,1\}^n$.

Definition (one-way function) : We say that f is *somewhat one-way* if, for some constant $c > 0$, for every feasible algorithm M, the inverting probability $\Pr[f(x) = f(M(f(x)))]$ when $x \in_U \{0,1\}^n$ is at most $1 - 1/n^c$. We say that f is *one-way* if, for every feasible algorithm M, the inverting probability $\Pr[f(x) = f(M(f(x)))]$ when $x \in_U \{0,1\}^n$ is negligible.

Notation (copies of functions) : Let $q(n)$ be a length function. Let f^q be the function with input and output lengths $n \cdot q(n)$ and $l(n) \cdot q(n)$, respectively, given by:

$$f^q(x_1 \circ \ldots \circ x_{q(n)}) = f(x_1) \circ \ldots \circ f(x_{q(n)}),$$

where $x_1, \ldots, x_{q(n)} \in \{0,1\}^n$.

The following is implicitly used in [Yao 82].

Proposition (somewhat one-way $\rightarrow$ one-way) : If f is somewhat one-way with associated constant $c > 0$, then f^q is one-way, where $q(n) = n^{c+1}$.

2.3 Pseudo-random generators

Informally, a polynomial-time computable function f is *pseudo-random* if $f(x)$ is strictly longer than x and if every feasible algorithm can distinguish $f(x)$ from a truly random string of the same length (when x is chosen randomly) with only negligible probability. Intuitively, $f(x)$ "looks" just like a random string to any feasible algorithm, even though it is generated from a string x that is strictly shorter. This intuition is captured in the following definition of [Blum, Micali 82], [Yao 82].

Definition (prg) : f is a *generator* if $l(n) > n$ for all $n \in N$. The *distinguishing probability* $p(n)$ of algorithm M for f is $|\Pr[M(f(x)) = 1] - \Pr[M(y) = 1]|$ when $x \in_U \{0,1\}^n$ and $y \in_U \{0,1\}^{l(n)}$. We say that f is *pseudo-random* if every feasible algorithm has negligible distinguishing probability for f.

The normal definition of a pseudo-random generator insists that the generator can stretch the input by any polynomial amount. The following shows the definition above is equivalent, and follows from [Goldreich, Goldwasser, Micali 84].

Proposition (polynomial stretching) : If there is a pseudo-random generator f then, for every constant $c > 1$, there is a pseudo-random generator with input length n and output length n^c.

The following proposition is from [Impagliazzo, Levin, Luby 89].

Proposition (one-way $\rightarrow$ prg) : If there is a one-way function in the non-uniform model of security then there is a pseudo-random generator in the non-uniform model of security.

The following definition appears in [Goldwasser, Micali 82], [Yao 82] and [Goldwasser, Micali, Rackoff 85].

Definition (comp. indistinguishable) : Let D and E be probability ensembles. The *distinguishing probability function* $p(n)$ of algorithm M for D and E is $|\Pr[M(x) = 1] - \Pr[M(x') = 1]|$ when $x \in_D \{0,1\}^n$ and $x' \in_E \{0,1\}^n$. D is *computationally indistinguishable* from E if every feasible algorithm M has negligible distinguishing probability.

The following definition is from [Impagliazzo, Levin, Luby 89].

Definition (false entropy) : We say function f has *false entropy* if there is a constant $c > 0$ such that the probability ensemble D defined by $D_n = f(x)$ where $x \in_U \{0,1\}^n$ is computationally indistinguishable from a probability ensemble E, where $Ent(E_n) \geq Ent(D_n) + 1/n^c$. In the uniform model of security, the definition requires that the distribution E_n can be generated by a probabilistic polynomial time Turing machine. (This is true by definition for D_n, since f is computable in polynomial time.)

The following proposition is from [Impagliazzo, Levin, Luby 89].

Proposition (false entropy $\rightarrow$ prg) : If there is an f that has false entropy then there is a pseudo-random generator.

2.4 Hash Functions

The concept of a universal hash function, introduced in [Carter, Wegman], has proved to have far reaching and a broad spectrum of applications in the theory of computation.

Definition (hash functions) : Let $H_{n,m}$ be a family of functions from n bit strings to m bit strings. We say $H_{n,m}$ is a family of *pairwise independent universal hash functions* if, for all $x, y \in \{0,1\}^n$, $x \neq y$, $h(x) \circ h(y) \in_U \{0,1\}^{2m}$ when $h \in_U H_{n,m}$. A *system of hash functions* consists of one such family for all pairs n and m. The following system of pairwise independent universal hash functions has several nice properties. Let $H_{n,m}$ be the set of all m by $n+1$ matrices over the field with two elements. We think of a hash function from this system as $h = (M, b)$, where M is an m by n bit matrix and b is a bit vector of length m. Then, $h(x) = (M \cdot x) \oplus b$, where $\oplus$ is vector sum mod 2 (i.e. bitwise parity). We can choose $h \in_U H_{n,m}$ by choosing $h \in_U \{0,1\}^{(n+1)m}$. Hereafter, whenever we refer to a family or system of hash functions, we mean the family defined here.

3 Technical Definitions and Tools

One of the main stepping stones in our proofs is a distributionally one-way function (defined below). The notion of a distributionally one-way function was inspired by [Rudich 88]. We give some motivation before making a formal definition. Intuitively, a one-way function is easy to compute but hard to invert. When the one-way function in question is a permutation the meaning of inverting the function is unambiguous: given $f(x)$ the inverting algorithm must find x. When f is many-to-one the task of the inverting algorithm is less clear. For example, if $f(x) = x'$ where x' is all but the last bit of x, there is no way any algorithm given $f(x)$ can find x with probability greater than $1/2$. A standard definition of a one-way function handles this problem by having the inverter succeed if it finds any y such that $f(y) = f(x)$. However, this allows inverters that might never guess the specific x that was chosen to generate $f(x)$. For example, let $g(x)$ be any one-way permutation and define $f(x, y)$ to be x if $y = 0$ and $g(x)$ otherwise. Then, the algorithm that on input $z = f(x, y)$ outputs $(z, 0)$ always finds some inverse of f but an atypical one. The f defined in this example intuitively has a lot of "one-wayness" built into it (via g) although it is not at all one-way by the standard definition. Thus, we informally define a weaker notion of one-way function that captures the "one-wayness" of this f: f is *distributionally one-way* if, given $f(x)$, it is computationally infeasible to *randomly and uniformly generate* a preimage of $f(x)$. Another way of viewing this is as follows: Alice chooses a random value x and publicly announces $f(x)$. Bob's goal is to imitate Alice; if Bob can randomly generate a preimage of $f(x)$ then to the world (that only knows the value of $f(x)$) there is no way of distinguishing Alice from Bob. Then, f is distributionally one-way if Bob is unable to imitate Alice in this sense.

Definition (statistically indistinguishable) : Let D_n and E_n be probability distributions on $\{0,1\}^n$. D_n and E_n are *statistically indistinguishable within* δ if for every $X \subseteq \{0,1\}^n$, $|\Pr[x \in X] - \Pr[y \in X]| \leq \delta$ when x is randomly chosen according to D_n and y is randomly chosen according to E_n.

Definition (distributionally one-way) : We say that f is *distributionally one-way* if, for some constant $c > 0$, for every feasible (probabilistic) algorithm M, the distribution defined by $x \circ f(x)$ and the distribution defined by $M(f(x)) \circ f(x)$ are statistically distinguishable by at least n^{-c} when $x \in_U \{0,1\}^n$.

If f is a distributionally one-way function then it is computationally infeasible to randomly generate preimages of $f(x)$. With this definition, we fix a parameter c of distinguishability: an algorithm that randomly generates preimages of $f(x)$ in a manner statistically close but still n^{-c} distinguishable from the uniform distribution is considered to be unsuccessful. This is a weak form of a one-way function, but the following lemma shows that such a one-way function can be used to construct a one-way function in the usual sense.

Lemma 1 : If there is a distributionally one-way function then there is a one-way function.

Proof Sketch : Let f be a distributionally one-way function with associated constant c. Let $g(h \circ i \circ x)$ be $h \circ i \circ f(x) \circ (h(x)_{\leftarrow i+2c\log n})$ (where $i \in \{1, \cdots, n\}$ and $h \in H_{n,n+2c\log n}$). In the full paper, we prove that g is a somewhat one-way function. The proof of this claim uses techniques and lemmas from [Impagliazzo, Levin, Luby 89]. By Proposition (somewhat one-way $\rightarrow$ one-way), this suffices to construct a one-way function. $\Box$

For all of our tasks, we show that any secure protocol can be used to construct a distributionally one-way function, which, by Lemma 1, can be used to constuct a one-way function.

Whenever possible, we show the stronger result that any secure protocol for a particular cryptographic task can be used to construct a pseudo-random generator.

4 Applications

Definition (protocol) : A *protocol* consists of a message length function, $m(n)$, a number of rounds function, $r(n)$, and two polynomial time deterministic Turing machines A and B, known as the participants "Alice" and "Bob". On input x for Alice and y for Bob, the conversation between Alice and Bob is defined by $C_{AB}(x,y) = m_1, m_2, \ldots, m_{r(n)}$, where $m_i = A(x, m_1, \ldots, m_{i-1})$ if i is even, and $m_i = B(y, m_1, \ldots, m_{i-1})$ if i is odd. Bob has a private output $OUT_B^{AB}(x,y)$ which is defined to be the output of B when the input to A is x and the input to B is y. Note that $OUT_B^{AB}(x,y)$ is completely determined by y and $C_{AB}(x,y)$. $OUT_A^{AB}(x,y)$ is defined similarly to be the output of A for the same inputs.

A well-known fact about protocols is:

Proposition 2 : If, for a particular protocol, $C_{AB}(x,y) = C_{AB}(x',y')$ then $C_{AB}(x,y) = C_{AB}(x,y') = C_{AB}(x',y) = C_{AB}(x',y')$. In other words, the set of input pairs x, y that yield the same conversation is $(x,y) \in X \times Y$, where X is a set of inputs for Alice and Y is a set of inputs for Bob.

4.1 Identification

Intuitively, an identification protocol allows Alice to convince Bob that she is the same person he was talking to at an earlier time.

Definition (identification protocol) : This consists of two protocols, P^1, called the "introduction" protocol, and P^2, called the "recognition" protocol. In the introduction protocol, Alice (A^1) has random input x and Bob (B^1) has random input y, and this produces the conversation $C_{A^1B^1}(x,y)$. In the recognition protocol, the input for Alice (A^2) is x, $C^1_{A^1B^1}(x,y)$ and x', where x' is a new randomly chosen input. Similarly, the input for Bob (B^2) is y, $C_{A^1B^1}(x,y)$ and y'. At the conclusion of the recognition protocol, Bob either outputs "ACCEPT" or "REJECT". A *secure identification protocol* has the following two properties:

- $\Pr[OUT_{B^2}^{A^2B^2}((x, x', C_{A^1B^1}(x,y)), (y, y', C_{A^1B^1}(x,y))) = \text{ACCEPT}] \geq .9$, i.e. Bob should recognize Alice with high probability if Alice participated in the original introduction protocol.
- For all but finitely many lengths n,

$$\Pr[OUT_{B^2}^{LB^2}((0, x', C_{A^1B^1}(x,y)), (y, y', C_{A^1B^1}(x,y))) = \text{ACCEPT}] \leq .1$$

for any feasible algorithm L that has access to the conversation from the introduction protocol but does not have the value of x that Alice has. (The random input x' to L is allowed to be of any length that L requires, i.e. x' may be much longer for L than it is for A). Intuitively, any listener L who heard the introduction conversation between Alice and Bob and who tries to impersonate Alice in the recognition protocol is caught with high probability by Bob.

Theorem : Any secure identification protocol can be used to construct a distributionally one-way function.

Proof Sketch : Let $f(x \circ y) = C_{A^1B^1}(x, y)$. We prove that if the identification protocol is secure then f is a distributionally one-way function by contradiction. Let $X \times Y$ be the set of pairs of inputs to A^1 and B^1, respectively, such that for all $(a, b) \in X \times Y$, $C_{A^1B^1}(a, b) = C_{A^1B^1}(x, y)$. Let M be the feasible algorithm that on input $f(x \circ y)$ produces random (a, b) such that the distribution is statistically indistinguishable within n^{-c} of the uniform distribution on $X \times Y$. The idea is to use M to help listener L simulate Alice during the recognition protocol without knowing x. L takes as input x' and $C_{A^1B^1}(x, y)$. L simulates M on input $C_{A^1B^1}(x, y)$, and this produces (a, b). During the recognition protocol, L simulates exactly what A would do when $x = a$. By Proposition 2 and because M produces close to the uniform distribution on $X \times Y$, the distribution on conversations from the recognition protocol produced by B interacting with L is statistically indistinguishable within n^{-c} from the distribution on conversations from the recognition protocol produced by B interacting with A. Thus, the difference between the probabilities of Bob's accepting in these two cases is not at least .8, as required for a secure identification protocol. □

4.2 Private Key Encryption

Private key encryption is a system by which two people who have previously met and privately selected a random n bit key k, can send messages of length longer than n securely over an insecure channel. In other words, a computationally limited listener who does not know k should not be able to deduce any information about which message (of length at least $n+1$) was sent from overhearing the conversation on the insecure channel. This notion of an interactive probabilistic private key encryption system was suggested in [Rackoff 88]. More formally:

Definition (weak private key encryption protocol) : Let $l(n) > n$ be a length function. The idea is that Alice and Bob share a secret key k of length n, and Alice wants to send to Bob a message m of length $l(n)$ securely, where m is randomly chosen. i.e. so that no feasible listener L (who does not know k) can determine anything about m. Alice and Bob are probabilistic polynomial time Turing machines. A has input the key k, the message m and a random tape x. B has input the key k and a random tape y. A and B have a conversation such that at the end of the conversation with high probability B knows m. (The fact that m is assumed to be randomly chosen is the reason we call this a weak protocol; usually m is assumed to be chosen from an arbitrary probability distribution. Our result that a weak protocol implies the existence of a pseudo-random generator immediately implies that a weak protocol can be used to build a strong private key encryption protocol.) More formally, a *secure weak private key encryption protocol* satisfies the following properties:

- With probability at least .9 Bob outputs the message m at the end of the protocol, i.e. $\Pr[OUT_B^{AB}((k,m,x),(k,y)) = m] \geq .9$.

- No feasible algorithm L can learn anything about m from the conversation between A and B, i.e. the distribution defined by $C_{AB}((k,m,x),(k,y)) \circ m$ is computationally indistinguishable from the distribution defined by $C_{AB}((k,m,x),(k,y)) \circ r$, where k, x, y, m, r are all uniformly and randomly chosen and $|r| = |m|$.

Theorem : Any secure weak private key encryption protocol can be used to construct a pseudo-random generator.

Proof Sketch : Let $f(x \circ y \circ k \circ m) = C_{AB}((k,m,x),(k,y)) \circ m$. We claim that the entropy of the distribution $C_{AB}((k,m,x),(k,y)) \circ m$ is significantly smaller than that of the distribution $C_{AB}((k,m,x),(k,y)) \circ R$. Thus, f has false entropy. (This is true in both the uniform and non-uniform model of security, since both distributions are polynomial samplable and they are computationally indistinguishable by assumption.) The theorem follows from Proposition (false entropy $\rightarrow$ prg).

The reason for the difference in entropies is that the $l(n)$ bit string m is determined with probability at least .9 by the conversation and the n bit string k, i.e. there is an algorithm (not necessarily efficient) that given the conversation and k produces m with probability at least .9. The algorithm randomly generates a y' so that Bob's behavior on input (k, y') is consistent with the conversation. Note that, for a fixed conversation and key k, the distribution on (m, y') is the same as the distribution on (m, y), where y is the original random tape for Bob. Therefore, the probability that the algorithm outputs m is the same as the corresponding probability for B. Thus, the expected conditional entropy on m given the conversation is at most $n + .1(l(n) - n) \leq l(n) - .9$ (because $l(n) \geq n + 1$), whereas that of R is $l(n)$. From this it follows that f has false entropy at least .9. A more complete argument will be included in the final version of this paper. □

4.3 Bit Commitment

Intuitively a bit commitment protocol makes Alice "commit" herself to a bit without allowing her to change her mind later in the protocol. Alice does not want Bob to know the value of her bit until a later time, and thus Alice sends encrypted information to Bob about the bit from which Bob is unable to determine its value with probability significantly greater than 1/2. On the other hand, when Alice finally does release the value of her bit by releasing information about the encryption, Bob would like to be sure that Alice can't change her mind about the original value to which she committed herself.

Definition (bit commitment protocol) : This consists of two protocols, P^1, called the "commit" protocol, and P^2, called the "release" protocol. In the commit protocol, Alice (A^1) has random input x and a random bit b, and Bob (B^1) has a random input y. Bob either outputs "ACCEPT" or "REJECT" at the conclusion of this protocol, and if and only if Bob outputs "ACCEPT" the release protocol is executed. In the release protocol, Alice (A^2) has input x, b, a random x', and the conversation from the commit protocol; Bob (B^2)

has y, a random y', and the commit conversation. At the end of the release protocol, Bob either outputs "REJECT" or a bit b'. If Alice and Bob obey the protocol, Bob should accept with high probability in the commit protocol, and should output $b' = b$ with high probability in the release protocol. Informally, the protocol is secure if: immediately after the commit protocol, no feasible Bob can predict b with probability more than $1/2 +$ a negligible amount; and, every feasible Alice has only one choice of bit b' she can make Bob output. Formally,

1. $OUT_{B^1}^{A^1B^1}((x,b),y) =$"ACCEPT", with high probability, for x, y chosen at random.

2. $OUT_{B^2}^{A^2B^2}((x,b,C_{A^1B^1}((x,b),y),x'),(y,C_{A^1B^1}((x,b),y),y')) = b$ with high probability, for x, y, x', y', b chosen at random.

3. Let B' be any feasible algorithm (for the commitment protocol), and let L be any feasible algorithm that tries to guess b after the commitment protocol and let A^1 be Alice's (real) algorithm for the commitment protocol. Then, $\Pr[OUT_L(y^*, y, C_{A^1B'}((x,b),y)) = b] \leq 1/2+$ a negligible amount, where y^* is the random tape input for L. (Note: y, the random input to cheating algorithm B', may be longer than the y input to B^1 would be.)

4. Let A'^1 and A'^2 be any feasible algorithms (for the two stages of the protocol, impersonating Alice), and let B^1 and B^2 be Bob's (real) algorithm for the two stages of the protocol. Let x, x', y, y' be the random tapes for A'^1, A'^2, B^1, B^2, respectively (x and x' may be longer than the x and x' used by the real A^1 and A^2). Then the probability that:

   (a) $OUT_{B^1}^{A'^1B^1}((x,b),y) =$ "ACCEPT".

   (b) $OUT_{B^2}^{A'^2B^2}((x,b,C_{A'^1B^1}((x,b),y),x'),(y,C_{A'^1B^1}((x,b),y),y')) \neq b'$ for some fixed value $b' \in \{0,1\}$.

   is negligible. (In other words, at the end of the commit protocol at most one value in $\{0,1\}$ is a possibility to be output by Bob at the end of the release protocol.)

If we remove the word "feasible" in the third clause above, we call the protocol "secure vs. a strong receiver". This is the type of bit commitment considered in [Brassard, Crepeau 86], [Chaum 86]. Similarly, if we remove the word "feasible" from the fourth clause, we call the protocol "secure vs a strong committer." This type of bit commitment is considered in [Goldreich, Micali, Wigderson 86].

Theorem :

1. Any secure bit commitment protocol with respect to a strong committer can be used to construct a pseudo-random generator.

2. Any secure bit commitment protocol can be used to construct a distributionally one-way function.

Proof Sketch :

1. Let $f(x \circ b \circ y) = C_{A^1B^1}((x,b),y) \circ y \circ b$. Let D_n and E_n be the following probability distributions. D_n is defined by $f(x \circ b \circ y)$ when x, y and b are randomly chosen. E_n is defined by $C_{A^1B^1}((x,b),y) \circ y \circ \beta$, where x, y, b and β are randomly chosen and β is a single bit. Every feasible algorithm L can only distinguish between D and E with negligible probability because of security condition 3. On the other hand, A^2 is a strong committer and can spend an unlimited amount of time to find an x' if one exists such that B^2 outputs a bit different from b in the release protocol with non-negligible probability. By security conditions 2 and 4b, no such x' can exist. Thus, with high probability, the bit b is information-theoretically determined by y and $C_{A^1B^1}((x,b),y)$. Thus, E has 1 more bit of entropy than D. Since E can be sampled in polynomial time, it follows that f has false entropy, and the result follows from the Proposition (false entropy $\rightarrow$ prg).

2. It is fairly easy to see that a bit-commitment scheme can also be used for identification. In the introduction protocol, Alice commits to several randomly chosen bits. In the recognition protocol, she releases these bits. If an eavesdropper Eve (without access to the random tape used by Alice in the introduction protocol) can cause Bob to accept her as Alice in the recognition protocol, one of the two following are possible.

   (a) Bob can simulate Eve to deduce the bits Alice committed to in the introduction protocol. If with probability significantly greater than $1/2$, the bits Eve causes Bob to think she releases to him are in fact the bits committed to in the introduction protocol, Bob has a good predictor for the committed bits, which violates security condition 3 of the bit commitment protocol.

   (b) Alice can cause Bob in the recognition protocol to output a bit she did not not commit to in the introduction protocol. She simulates Eve in place of implementing the recognition protocol. With probability around $1/2$, Bob outputs a bit not equal to one of the corresponding bits Alice committed to in the introduction protocol, which violates security condition 4b of the bit commitment protocol. □

[Naor 88] shows how a bit commitment protocol secure versus a strong committer can be constructed from any pseudo-random generator. [Goldwasser, Micali, Rackoff 85] define the notion of a zero-knowledge proof system, and [Goldreich, Micali, Wigderson 86] show how any bit commitment protocol that is secure versus a strong committer can be used to construct a zero-knowledge proof system for any problem in *NP*. Combining our result with Naor's, we get the following: if any protocol for bit commitment versus a strong committer exists, then a one round protocol for this exists. In the non-uniform model of security, this can be strengthened to: if any bit commitment protocol exists, a one round protocol secure versus a strong committer exists.

4.4 Coin Flipping on the Phone

Coin flipping by telephone [Blum 82] is a method for two participants who do not trust each other to choose a random bit. The protocol $P = (A, B)$ should have the following properties.

At the end of the protocol, both Alice and Bob output either a bit in $\{0,1\}$ or the word "REJECT". If both parties obey the protocol, with high probability, both Alice and Bob output a bit, and the bit output is the same for both parties. When both parties obey the protocol, this common bit should have bias $1/2+$ a negligible amount. A party that cheats and desires a particular outcome $b \in \{0,1\}$ should only succeed with probability negligibly greater than $1/2$ in getting the other party to output b. We will give a formal definition and prove the following in the full paper.

Theorem : If there is a secure protocol for coin-flipping by telephone, then there is a distributionally one-way function.

5 Summary

The results in this paper and several other recent papers in theoretical cryptography as summarized in the following table for the uniform model of security. The first column is for tasks that are possible with no assumptions. The second column is for tasks such that the existence of any secure protocol is equivalent to the existence of a one-way function. The third column shows tasks such that any pseudo-random generator can be used to construct a secure protocol, and any secure protocol can be used to construct a one-way function. The fourth column shows tasks such that the existence of a secure protocol is equivalent to the existence of a pseudo-random generator. In the non-uniform model of complexity, columns 2-4 collapse into one column by the results in [Impagliazzo, Levin, Luby 89].

no assumptions	$\equiv$ one-way	prg $\rightarrow$ task $\rightarrow$ one-way	$\equiv$ prg
secret sharing	informationally one-way function	bit commitment	private key encryption
one time pad	weak identification	strong identification	pseudo-random fnc. generators
		coin flipping by telephone	pseudo-random perm. generators
		zero-knowledge facts about committed values	strong bit commitment
			false entropy generator

References

[1] Bennett, C., Brassard, G., Breidbart, S., and Weisner, S., "Quantum Cryptography, or Unforgeable Subway Tokens," *Proceedings of CRYPTO 1982.*

[2] Blum, M., "Coin Flipping by Phone," *IEEE Spring COMPCOM*, Feb. 1982, pp. 133-137.

[3] Brassard, G., Crepeau, C., "Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond," *Proceedings of FOCS 1986.*

[4] Blum, M., and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," *SIAM J. on Computing*, Vol. 13, 1984, pp. 850-864, *FOCS 1982*.

[5] Carter, J., and Wegman, M., "Universal Classes of Hash Functions," *JCSS*, 1979, Vol. 18, pp. 143-154.

[6] Chaum, D., "Demonstrating that a public predicate can be satisfied without revealing any information about how," *CRYPTO 1986*.

[7] Crepeau, C., Kilian, J., "Achieving Oblivious Transfer Using Weakened Security Assumptions," *Proceedings of* 29th *FOCS*, pp. 42-52, 1988.

[8] Goldreich, O., S. Goldwasser, and S. Micali, "How to Construct Random Functions," *J. of ACM*, Vol. 33, No. 4, 1986, pp. 792-807, *FOCS 1984*.

[9] Goldreich, O., Micali, M. and Wigderson, A., "Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design," 27th *FOCS*, 1986, pp. 174-187, *Tech. Report TR498*, Comp. Sci. Dept., Technion, submitted to *JACM*.

[10] Goldwasser, S. and Micali, S., "Probabilistic Encryption," *JCSS*, Vol. 28, No. 2, April 1984, pp. 270-299, *STOC 1982*.

[11] Goldwasser, S., Micali, S. and Rackoff, C., "The Knowledge Complexity of Interactive Proof Systems," *SIAM J. on Computing*, Vol. 18, No. 1, 1989, pp. 186-208, *STOC 1985*.

[12] Impagliazzo, R., Levin, L., Luby, M., "Pseudo-random generation from one-way functions," *Proceedings of STOC 1989*.

[13] Karp, R., Lipton, R., "Turing Machines that Take Advice," *L'Enseignement Mathematique*, Vol. 28, pp. 191-209 (1982), *STOC 1980*

[14] Luby M., and Rackoff, C., "How to Construct Pseudorandom Permutations From Pseudorandom Functions," *SIAM J. on Computing*, Vol. 17, 1988, pp. 373-386, *STOC 1986*

[15] Naor, M., personal communication, 1988.

[16] Rackoff, C., *Scribe Notes*, Department of Computer Science, U. of Toronto, 1985.

[17] Rackoff, C., "A Basic Theory of Public and Private Cryptosystems," *Proceedings of CRYPTO 1988*

[18] Rudich, S., ????, *Ph.D. Thesis*, Department of Computer Science, U.C. Berkeley, 1988.

[19] Shannon, C., "A Mathematical Theory of Communication," *Bell Systems Technical Journal*, 27, 1948, pp. 379-423 and pp. 623-656.

[20] Yao, A.C., "Theory and Applications of Trapdoor Functions," 23rd *FOCS*, 1982, pp. 80-91.