Theory and Applications of Trapdoor Functions

(extended abstract)

Andrew C. Yao
Computer Science Division
University of California
Berkeley, California 94720

Abstract

The purpose of this paper is to introduce a new information theory and explore its appplications. Using modern computational complexity, we study the notion of information that can be accessed through a feasible computation.

In Part 1 of this paper, we lay the foundation of the theory and set up a framework for cryptography and pseudorandom number generation. In Part 2, we study the concept of trapdoor functions and examine applications of such functions in cryptography, pseudorandom number generation, and abstract complexity theory.

Part 1: Computational Information Theory

I. Introduction

By now, Shannon's definition of information (in terms of *entropy*) has been universally accepted as the correct measure of statistical events [22]. It possesses a number of desirable properties, and has been shown to be the only possible definition which can satisfy these properties (see Shannon [22]). It appears in several important theorems, always providing the natural interpretation. Then, why are we interested in modifying this fundamental definition, and hence the entire theory?

The answer is that, roughly speaking, sometimes it may take an astronomical amount of computation to extract the Shannon information contained in a string; and on such occasions, the conclusions reached by information theory may become inconsequential. To illustrate this more precisely, let us first review some basic facts in the Shannon theory.

Let $\Sigma = \{a_1, a_2, \ldots, a_s\}$ be an alphabet, i.e., a finite set of symbols, and p a probability density over Σ, with $p_i = p(a_i)$. Consider a device S that stochastically generates an infinite sequence $b_1 b_2 b_3 \ldots$ one symbol at a time, where each b_j is independently distributed according to p. Call such a device S a *source*, and let the *entropy* of S be $H(S) = \sum_i p_i \log_2(1/p_i)$. One could think of $H(S)$ as the amount of *uncertainty*, or *information*, contained in each output symbol generated by S. That this is a reasonable interpretation will be borne out in the next theorem.

Suppose two people A and B are far apart, with a source S on A's side, and a communication medium by which A can send binary bits to B. What is the most efficient way (in terms of minimizing the number of bits sent) for A to inform B of the string generated by S? More precisely, suppose A wants to send n consecutive symbols output by S, what is L_n, the minimum expected number of bits A has to send? (See [22] for a precise definition of L_n).

Shannon's First Theorem [22]. $\lim_{n\to\infty} \frac{L_n}{n} = H(S)$.

In fact, it is true that

$$nH(S) \le L_n \le nH(S) + 1.$$

In other words, the minimum average number of bits to describe one symbol output by S is $H(S)$.

This is one of the two fundamental theorems in information theory; we will review the other one later on. But right now, we are ready to give a detailed illustration of how computational considerations may affect the conclusions supplied by information theory.

Example 1. Let Σ be the set of all k-bit binary strings with $k = 10^4$. For any 100-bit integers x and m, let $\alpha_{x,m}$ denote the string $c_1 c_2 \ldots c_k$ where c_j = parity of $(x^j \bmod m)$. Let $A \subseteq \Sigma$ be the multi-set $\{\alpha_{x,m} \mid x, m\}$. Thus, $|A| = 2^{200}$ and $|\Sigma| = 2^{10000}$. Consider the source S over Σ with distribution density $p(y) = 1/|A|$ if $y \in A$ and $p(y) = 0$ otherwise. Clearly, $H(S) \le \log_2 |A| = 200$. Shannon's First Theorem states that, in principle, A can send n output symbols of S to B using $nH(S) = 200n$ bits. In fact, A can simply represent each output $\alpha_{x,m}$ by the 200-bit string xm. However, in order to do this, A has to compute x and m from the 10000-bit string $\alpha_{x,m}$, as the latter is all A knows. It is not obvious that this can be done in a reasonable amount of computing time.

The above example is only used to demonstrate that Shannon's First Theorem does not immediately guarantee a computationally feasible encoding using $H(S)$ bits per source output. We do not know the true answer to this particular problem.

We now review some more information theory. Again, assume that A wishes to inform B of the output of a source S. This time however, the communication medium C between A and B, while still

0272-5428/82/0000/0080$00.75 © 1982 IEEE

carrying 0, 1 signals, is imperfect in the sense that there is a probability q for a "0" signal to be received as a "1", and similarly a probability q for a "1" to be received as a "0". In the literature, C is called a *binary symmetric channel*, or BSC. How many bits must be sent over the channel C in order to communicate n output bits of S? As errors are inevitable if $q > 0$, we are interested in keeping the probability for error below any pre-assigned level $\epsilon > 0$.

Let $I = \{0,1\}, J = \{0,1\}$ denote the input and output alphabets of C. A *code* of length m is a subset $E \subseteq I^m$; a *decoding rule* is a function $f : J^m \mapsto I^m$. For any $x \in E$, let $P(x)$ be the probability that $f(y) \neq x$ if y is received when x is input to C. Define the *capacity* of C by $\textit{capacity}(C) = 1 - q\log_2(1/q) - (1-q)\log_2(1/(1-q))$.

Shannon's Second Theorem [22]. For any $R < \textit{capacity}(C)$ and $\epsilon > 0$, there exists (for sufficiently large m) a code E of length m and a decoding rule f such that (a) $|E| \geq 2^{Rm}$, (b) $P(x) < \epsilon$ for all $x \in E$.

The ratio $(\log_2 |E|)/m$ is called the rate of the code E, as it is the average number of message bits communicated for each bit sent over the channel when code E is used.

There are converses to the above theorem. Essentially, they state that to transmit at rate $R > \textit{capacity}(C)$ is impossible without making the error $\epsilon \to 1$.

Shannon's Second Theorem is true for channels more general than BSC. Define a *channel* C to be an $r \times t$ matrix (v_{ij}) satisfying $v_{ij} \geq 0$ and $\sum_j v_{ij} = 1$. The interpretation is that C has an input alphabet $I = \{b_1, b_2, \ldots, b_r\}$ and an output alphabet $J = \{c_1, c_2, \ldots, c_k\}$, such that if b_i is input, then the output will be c_j with probability v_{ij}. It can be proved that Shannon's Theorem is true for properly defined *capacity*(C). We refer the readers to textbooks on information theory (e.g. [8]) for details.

In theory, Shannon's Second Theorem makes it possible to construct an explicit code with given rate $R < \textit{capacity}(C)$ and error bound $\epsilon > 0$ in $O(1)$ time, by exhaustive search over all possible codes of length $1, 2, \ldots$. However, such a procedure is in practice computationally prohibitive. The study of finding practical codes has a large literature [8], and there has also been improved theoretical results on the time and storage requirements for codes achieving given R and ϵ (Ziv[26]). It is therefore true that the computational complexity has already received much attention in information theory. However, the computational aspect in which we are interested is quite different. For our purpose, the coding problem for the classical channels has essentially been solved. We are mainly interested in the situation when the source becomes non-classical, i.e., the alphabet size can no longer be regarded as a constant, as illustrated in Example 1, and in the situation when the channel becomes non-classical as we will see later.

2. Effective Entropy

Given a source S with a large alphabet such as in Example 1, how should we define the information contained in its output? We will adopt the view that the minimum average number of bits needed to describe an output in a computationally feasible way is the proper measure. In other words, we will regard Shannon's First Theorem as a definition.

We have in mind the following situation. The source has an alphabet whose symbols are finite binary strings with an average length n (say $n \approx 200$). Person A is interested in communicating to B a sequence σ of n^t output symbols of S (t is a fixed integer, say $t = 3$). The question is, how short a string ρ can A compute in a reasonable amount of time (say in time n^k for some fixed k), so that B, on receiving ρ, can recover σ in a reasonable amount of time? To define this concept precisely, we resort to the well-developed computational complexity theory. In this theory, the complexity of a computational problem is measured by the asymptotic behavior of algorithms as the input length becomes large. To apply the theoretical results to input of a particular length, we tacitly assume that this length is large enough that the asymptotic results can be used. For example, suppose theoretically one can prove that the decision problem for a certain formal logic system has complexity $\Omega(2^{2^n})$, i.e. any algorithm T for the decision problem must have a running time $\geq C_T 2^{2^n}$ for some constant $C_T > 0$. We will then regard that, for formula size $n \approx 1000$, any reasonable algorithm must use time $\approx 2^{2^{1000}}$ for some input formula. Taking this approach, we need to consider not one source, but a sequence of sources, and look at the asymptotic behavior of the quantities of interest.

Definition 1. Let Σ be a fixed, finite alphabet. A *source* S is a probability distribution p over Σ^+ with a finite expected length $\beta(S) = \sum_x p(x)|x|$. A *source ensemble* S is a sequence of sources $S_1, S_2, \ldots$, with probability distributions $p_1, p_2, \ldots$, such that for some fixed $t_2 > t_1 > 0$, $p_n(y) > 0$ implies $n^{t_1} < |y| < n^{t_2}$.

Remark. This last assumption is not essential, but useful for simplifying later discussions; it is satisfied by most applications that we are interested in.

In the following, a *probabilistic algorithm* means a probabilistic multi-tape Turing Machine [9] that always halts. One can alternatively thinks of it as a program on a random access computer that always halts, since we will only prove results that are polynomially invariant.

Notation. We will use the notation $O(v(n))$ for any function $f(n)$ that vanishes faster than $1/n^t$ for every fixed t.

The following definition specifies precisely how A is allowed to encode a sequence of n^k (for some $k > 0$) output symbols from S^n, and how B decodes it.

Definition 2. Let $t, k > 0$ be any fixed number. A (t,k)-*encoding* of S is a triplet of probabilistic algorithms $M = (M_A, M_B, M_C)$ satisfying the following properties:

(a) Given input $\alpha = (n, x_1, x_2, ..., x_{n^t})$, where $p_n(x_i) > 0$ for all i, algorithm M_A will halt in time $O(n^t)$ and leave some binary string β on the output tape of M_A; let $q_n(\alpha)$ denote the probability distribution of β;

(b) Given stochastically an input pair (n, β) where β is distributed according to $q(\alpha)$, the algorithm M_B will halt in time $O(n^t)$ and leave the string α on the output tape of M_B with probability $1 - O(v(n))$;

(c) Let $b > 0$ be any fixed number. Given n and any string $\beta = \beta_1\beta_2...\beta_u$ where each β_i is a possible output from M_A for some α_i and $u = O(n^b)$, the algorithm M_C halts in time $O(n^{b'})$ for some fixed b' and output $\beta_1\beta_2...\beta_u$ correctly with error probability $O(v(n))$.

Define $\ell_n(\alpha)$ to be the expected length of β over $q_n(\alpha)$. Let $p_n(\alpha) = p_n(x_1)p_n(x_2)...p_n(x_{n^k})$. Let $\ell_n(M; S) = \sum_\alpha p_n(\alpha)\ell_n(\alpha)/n^k$, the average number of bits used by M to encode an output symbol x of S_n.

Remark. Properties (a) and (b) stipulate that encoding and decoding can be done in polynomial computation time, and with error $O(v(n))$. Property (c) roughly states that the code is a uniquely decipherable code, in the sense that to transmit $u \cdot n^k$ outputs of the source S_n, one can encode each n^k segments seperately and then concantenate the u blocks for transmission.

Definition 3. A (t,k)-*entropy sequence* for S is a sequence $w_1, w_2, ...$ such that there exists a (t,k)-encoding M for S with $\ell_n(M; S) = w_n$.

Only the asymptotic behavior of w_n is of interest, since for any fixed n, we can choose M with enough states to make $w_n = H(S_n)$, the Shannon entropy of S_n.

Definition 4. We say that the *effective entropy* $H_c(S)$ *is less than* $g(n)$, or in symbols $H_c(S) \leq g(n)$, if there exists $t, k > 0$ and a (t,k)-entropy sequence $\langle w_n \rangle$ for S such that $w_n \leq g(n)$ for all sufficiently large n.

Similarly, we write $H_c(S) \leq h(n)$ if, for every fixed $t, k > 0$, every (t,k)-entropy sequence $\langle w_n \rangle$ for S satisfies $w_n \geq h(n)$ for all sufficiently large n. We will also use notations such as $H_c(S) = O(g(n))$, $\Omega h(n))$, $\Theta(f(n))$, etc.

Thus, we are using the term "effective entropy of a source ensemble" in the same spirit as we speak of "the computational complexity of a decision problem"; neither is a well-defined quantity, but can serve as a useful shordhand. In some cases, however, one has upper and lower bounds tight enough to write equalities such as $H_c(S; n) = g(n) + O(\log n)$. An important case is the source ensemble that corresponds to random numbers.

Definition 5. The *true random number ensemble* T_0 is the source ensemble $S_1, S_2, ...$, where S_n is defined to be the probability distribution $p_n(x) = 2^{-n}$ if $|x| = n$, and 0 otherwise.

Theorem 1. For any source ensemble S, $H_c(S; n) \geq H(S_n) + O(v(n))$.

Corollary. $H_c(T_0; n) = n + O(v(n))$.

The additive $O(v(n))$ appears in the above theorem, because we allow $O(v(n))$ probability error in Definition 2. The form of the equality in the Corollary turns out to have a special significance as will become clear in Section 5.

Are there source ensembles with effective entropies much greater than their Shannon entropies? The answer is yes. Let $g(n)$ be any monotonic function that goes to infinity and $(\log_2 n)^2 < g(n) < n$.

Example 2. The following source ensemble $S = S_1, S_2, ...$ can be shown to have $H(S_n) = g(n)$, while $H_c(S; n) = n + O(v(n))$. Let $\alpha_1, \alpha_2, ...$ be a sequence of binary strings, where α_n has length $n \cdot 2^{g(n)}$ and is of maximum Kolmogroff-Chaitin information [6][14]. Write α_n as $\alpha_{n1}\alpha_{n2}...\alpha_{nG(n)}$, where $G(n) = 2^{g(n)}$, with $|\alpha_{ni}| = n$; let S_n be defined by $p(x) = 2^{-g(n)}$ if $x = \alpha_{ni}$ for some i, and 0 otherwise. It is in fact not hard to modify the above definition to have a constructable S, using a relativized version of algorithmic information (defined by Levin).

3. Reliable Transmission

The centerpiece of information theory is Shannon's Second Theorem, which states that a channel C can reliably transmit information at a rate of R bits per channel transition provided $R < capacity(C)$. Further, a rate of $R > capacity(C)$ is impossible to achieve. The natural quesiton facing us is: How fast can a channel transmit computational information? In this section, we will study one side of this question, namely: Can C tansmit computational information reliably at rate R fbits with $R > capacity(C)$? (We have coined the term *fbit* as a unit for effective entropy.)

Let us first define the above question more precisely. As the details are similar to Definiton 2, we will state the definitions informally. In contrast to the way a code and its rate are defined in the Shannon case, we need to define the concepts of code and rate relative to a source ensemble.

Definition 6. Let $S = \langle S_n \rangle$ be a source ensemble, and let C be a channel with input-output symbol sets I and J. A (t,k)-*coding scheme of* S *over* C is a triplet of probabilistic algorithms $M = (M_A, M_B, M_C)$ that always halt in polynomial time $O(n^t)$. For any n and a string α of n^k outputs from S_n, the *encoder* M_A stochastically computes a string $\beta \in I^*$, sends it across channel C; the *decoder* M_B takes the resulted output string $\gamma \in J^*$ and computes a string δ. The requirement is that, when averaged over the probabilistic

distribution of α and over all the stochastic moves of M_A, M_B and C, the probability of $\delta \neq \alpha$ is of order $O(v(n))$. The strings γ (output from the channel C) are uniquely decipherable by algorithm M_c (as in Definiton 2), again allowing for failure probability $O(v(n))$.

Definition 2 can be regarded as a special case of Definition 6. A (t,k)-encoding of S is essentially a (t,k)-coding scheme of S over C, where C is the binary symmetric channel with crossover probability $q = 0$.

Definition 7. In the previous definition, let $\ell_n(M;S;C)$ be the expected value of $|\beta|/n^k$, when the input α to M_A is generated probabilistically by source S_n.

Thus, $\ell_n(M;S;C)$ is the average number of channel symbols used by M to transmit reliably over channel C one output symbol of the source S_n. To see what "rate", the traditional performance measure of a code, corresponds to, let us for the moment assume that $\mathcal{H}_c(S;n)$ has a sharp asymptotic behavior (e.g. $\mathcal{H}_c(S;n) = \sqrt{n} + O(n^{1/5})$). In such a case, the natural definition of the rate of M (for S over C) is

$$R_n = \frac{\mathcal{H}_c(S;n)n^k \ \textit{fbits}}{\ell_n(M;S;C)n^k \ \textit{channel transitions}} = \frac{\mathcal{H}_c(S;n)}{\ell_n(M;S;C)} \ \textit{fbits/channel transition.}$$

Then, the question of whether it is possible to transmit computational information reliably at a rate higher than the channel capacity becomes "Do there exist an M and a fixed $\epsilon > 0$ such that $\mathcal{H}_c(S;n) > (\textit{capacity}(C) + \epsilon)\ell_n(M;S;C)$? ".

The next theorem says that the answer is "no". (The statement of the theorem is valid even if $\mathcal{H}_c(S;n)$ is not sharply determined.)

Theorem 2. Let M be an encoding scheme for a source ensemble S over a channel C. Then for any fixed $\epsilon > 0$, there exists an entropy sequence $\langle w_n \rangle$ for S such that

$$w_n < (\textit{capacity}(C) + \epsilon)\ell_n(M;S;C)$$

for all sufficiently large n.

Corollary. If $\mathcal{H}_c(S;n) \geq h(n)$, then

$$\ell_n(M;S;C) > \frac{h(n)}{\textit{capacity}(C) + \epsilon}$$

for all sufficiently large n.

The proof of this theorem is quite involved. We will not get into it here.

Let us mention two reasons why Theorem 2 is important for our theory from an aesthetical point of view. First, if the theorem were false, we would have a channel (say with capacity equal to 2) for which each channel input symbol can carry at most 2 bits of Shannon information, but may carry more, say 2.4 fbits, of computational information. The natural interpretation of Shannon's Second Theorem will be lost for transmitting computational informaion. We would also have to accept that, even in simple situations, not all fbits behave alike.

The second point is the there is no obvious reason why the Theorem is true. Consider the coding of a source ensemble S with $\mathcal{H}_c(S;n) \approx \sqrt{n}$. The codewords conceivably can be so sparsely populated that if we use the same coding to transmit over a BSC with small crossover probability q, the displaced codewords after channel transmission are still widely apart. Thus, there is no purely combinatorial obstacle for the encoding to remain a valid one for a BSC, and retain a rate higher than that allowed by Theorem 2. The fact that a consistent interpretation is obtained in a new environment after complexity-type reasoning gives us confidence that our definition is on the right track.

4. INDISTINGUISHABILITY

Let S and S' be two sources with known distinct probability distributions p and p' over Σ^+, where Σ is a fixed alphabet. Suppose a box that simulates one of the sources is given to you, but you are not told which source. The box will emit upon each request a string distributed in accordance with the underlying distribution. Can you tell with confidence which source the box is simulating?

For the classical sources, the answer is "yes" since one can always take enough outputs and observe the frequency of occurrance of any particular string v for which $p(v) \neq p(v')$. In the non-classical case, however, the question is more complex. Even when p and p' differ substantially, say $p(v) = |\Sigma|^{-n}$ for all v of length n and $p'(v) = 1/|T|$ where $T \subseteq \Sigma^n$ and $|T| = |\Sigma|^{\sqrt{n}}$ it is not obvious that there is a way of deciding which alternative is true (an astronomical number of observations will be needed if we use the method mentioned above for classical sources). We now define precisely what we mean by two indistinguishable sources.

Definition 8. Let $S = \langle S_n \rangle$, $S' = \langle S'_n \rangle$ be two source ensembles. A *witness algorithm* M for (S, S') is a probabilistic algorithm such that the following properties are true for some fixed t, k and $\epsilon > 0$:

(a) For any input (n, α), where $\alpha = (x_1, x_2, \ldots, x_{n^k})$ is a sequence of n^k outputs of S_n, the algorithm M halts in time $O(n^t)$ and leaves a boolean output $M(n,\alpha)$; let $f_n(M,S)$ be the probability that $M(n,\alpha) = 1$ when α is generated probabilistically by S_n;

(b) Similarly, let $f_n(M,S')$ be the corrresponding probability for S';

(c) There exists an infinite sequence of (distinct) values $n_1, n_2, \ldots$ such that

$$|f_n(M,S) - f_n(M,S')| > \epsilon \qquad for \quad n = n_1, n_2, \ldots .$$

Definition 9. Two source ensembles S and S' are said to be *indistinguishable* if there exists no witness algorithm for S and S'.

Note that a witness algorithm may not be a suitable algorithm for deciding if a box is simulating source S or S', since condition (c) only guarantees that S and S' behave differently for some values

of n. The definition of witness algorithm is designed to ensure that two indistinguishable sources behave almost identically for any test as $n \to \infty$.

There exist indistinguishable source ensembles with very different underlying probability distributions. In fact , the ensemble S defined in Example 2 and the true random number ensemble $\mathcal{T}_0$ (Definition 5) are indistinguishable. The reason will become clear in the next section.

5. A Theory Of Pseudorandom Nunbers

The study of the concept of randomness in strings has received considerable attention (see Knuth [13]). There are two types of results: the first type deals with the question of "what is a random sequence", and the second type delas with the problem of pserdorandom number generation. The former question is concerned with properties of a single sequence, and has been satisfactorily answered through the work of many researchers (e.g., Kolmogroff[14], Chaitin[6], Martin-Lof[15], Levin[27], Meyer and McCreight[16]); we will be interested in the latter problem.

The need for pseudorandom numbers arises on many occasions such as simulation, sampling, cryptography, etc.. How should one choose a pseudorandom number generator for a particular application? In the literature, there are many proposed methods of generating pseudorandom numbers, and various *statistical tests* are available for measuring the strength of a proposed scheme. If, in an application, it is possible to isolate some simple randomness properties that can guarantee success, then a statistical test based on the desired randomness properties can be used to screen and select a appropriate generator. This, however, is seldom the case. Furthermore, the performance of a pseudorandom number generator under a particular statistical test is usually hard to determine analytically, and often has to rely on empirical evidence.

Wouldn't it be nice if there existed a pseudorandom number generator that is fit for all applications? In this section, we will set up a framework for discussing pseudorandom numbers, and introduce the concept of a *perfect* pseudorandom number generator. In Part II of this paper we will exhibit a class of generators that have strong theoretical evidence to be perfect.

For our purposes, a pseudorandom number generator is an algorithm that takes some true random bits and generates deterministically a much longer sequence of bits. For example, one possible *linear congruential generator* is to choose randomly four n-bit numbers m, a, c, X_0 and generate a n^3-bit sequence $\rho = X_1X_2\cdots X_{n^3}$ by $X_{j+1} = (aX_j + c) \bmod m$. We can regard the probability distribution of the final string as a source. The strength of the pseudorandom number generator can be studied as a property of the source, without regard to how it is generated. Let us first see how one can formalize this in terms of sources.

Definition 10. A source ensemble $S = (p_n)$ is said to be *uniform* if all strings x with $p_n(x) > 0$ have the same length $\xi(n)$, and furthermore $\xi(n) < \xi(n+1)$ for all n. Call $\xi(n)$ the *length function* for S.

For simplicity, we will assume that the alphabet is $\{0, 1\}$ for the rest of this section. Note that by definition, $n^{t_1} < \xi(n) < n^{t_2}$ for some $t_1, t_2 > 0$.

Let $\xi(n)$ be an integral-valued function with $\xi(n) < \xi(n+1)$. Define $\mathcal{T}_\xi$ to be the source ensemble (S_n) where S_n is the source corresponding to the $\xi(n)$-bit random numbers.

Definition 11. A uniform source ensemble S with length function $\xi(n)$ is said to be *perfect* if S and $\mathcal{T}_\xi$ are indistinguishable.

We want to stress that a uniform source ensemble does not necessarily correspond to a pseudorandom number generator, because the underlying distributions may not be generated from a small number of random bits. Before turning to the task of defining pseudorandom number generator, we want to explore the property of being "perfect" in more depth.

How does the concept of statistical test fit into the picture? Let us define the term rigorously.

Definition 12. A *polynomial statistical test* is a probabilistic algorithm M that takes only inputs of the form $(x_1, x_2, \ldots, x_{N^k})$, where each x_i is an N-bit number, halts in time $O(N^t)$, and outputs a binary string y, where t and k are some fixed positive integers.

Definition 13. In the above definition, let $\eta(N, y)$ be the probability that y appears as output when the inputs x_i are independent N-bit numbers. For any uniform source exsemble S with length function ξ, let $\eta_M(N, y; S)$ be the probability y appears as output when the inputs x_i are generated by source S_n where $N = \xi(n)$. ($\eta_M(N, y; S)$ is defined only when $N = \xi(n)$ for some n.)

Informally, the test M takes N^k N-bit outputs from source ensemble S and computes a quantity y. This quantity y is usually used to produce a fraction $s = f(y, N)$ (either by table look-up or by another calculation), which represents a confidence level that S should be rejected as non-random (see e.g. [13]).

Definition 14. Let M be a polynomial statistical test, and S a uniform source ensemble. We say that S *passes the statistical test* M if $\eta_M(N, y; S) - \eta_M(N, y) = O(v(N))$ for all y.

Theorem 3. A uniform source ensemble S is perfect if and only if S passes every polynomial statistical test.

This establishes the link between statistical tests and our definition of a perfect source ensemble. So far, we have discussed this concept of "perfect" completely in terms of computational complexity. The next result shows that we can also express it in terms of computationl information.

Theorem 4. A uniform source ensemble S is perfect if and only if $\chi_c(S;n) = n + O(v(n))$.

The significance of the equality in the corollary to Theorem 1 ashhould now become clear: it characterizes the perfect source ensembles. It also follows from Theorem 4 that the S in example 2 is a perfect source ensemble, and hence indistinguisahable form T_0.

We close this section with by defining pseudorandom number geneators in the present setting.

Definition 15. A *pseudorandom number generator* G is a probabilistic algorithm such that, given input integers k and n, it

(a) halts in polynomial time in n,

(b) uses $O(n)$ true random bits, and

(c) outputs a binary string α of length n^k.

For each k, the above generator G defines a natural source ensemble $\mathcal{W}_k(G) = \langle S_n \rangle$, where S_n is the source that outputs α of length n^k with the same probability as G does (when n is the input).

Definition 16. A pseudorandom number generator G is said to be *perfect* if $\mathcal{W}_k(G)$ is a perfect source ensemble for every fixed k.

6. Mutual Information And Independence

We first review the concepts of *mutual information* and *independence* in Shannon's theory. Let $Q = (T, p)$ be a probability space, where T is a finite sample space and p is a probability distribution on T. For any random variable $X : T \mapsto V_X$, where the number of possible values $|V_X|$ is finite, let the *entropy* of X be

$$H(X) = \sum_{x \in V_X} p_X(x) log_2 \frac{1}{p_X(x)},$$

where $p_X(x) = \sum_{X(t)=x} p(t)$ is the probability that $X = x$. We can regard $H(X)$ as the average amount of information that a single observation of the value of X gives us. The entropy $H(S)$ of a source S can be obtained as a special case, when S is both the sample space and the random variable.

Let Y be another random variable on Q. The *conditional entropy* of X given Y is defined as

$$H(X|Y) = \sum_{y \in V_Y} p_Y(y) \sum_{x \in V_X} Pr\{X = x \mid Y = y\} \log \frac{1}{Pr\{X = x \mid Y = y\}},$$

i.e., the average amount of information an observation of X gives us, when the value of Y is already known. The *mutual information* of Y about X is $I(X,Y) = H(X) - H(X|Y)$. One interesting fact is that the mutual information is symmetric, i.e., $I(X,Y) = I(Y,X)$. Quantitatively, it means that if knowing the value of Y tells you some information about the value of X, then knowing the value of X tells you roughly the same amount of information about Y.

Two random variables X and Y are said to be *independent* if $I(X,Y) = 0$. It can be shown that this agrees with the traditional usage of this term in probability theory, i.e., $Pr\{X = x, Y = y\} = Pr\{X = x\} \cdot Pr\{Y = y\}$ for all x, y.

Let us now define the corresponding notions in our theory. As in defining effective entropy, we begin with an example to illustrate the need for different definitions when computational efforts are taken into account.

Consider the sample space T of all bipartite graphs G between two vertex set of 1000 nodes each, and assume that all such graphs are equally likely to occur. Let X and Y be random variables defined by $Y(G) = G$ and $X(G) =$ the number of perfect matchings in G (mod 3). It is clear that, since one can compute the value of X from that of G, $H(X|Y) = 0$. Thus, $I(X,Y) = H(X)$; we can say that Y contains all the Shannon information about X. However, we do not know any efficient way of computing X from G, and it is conceivable that no efficient algorithms exist (in an average-case sense). If it were the latter case, then we can view the Shannon information which Y contains about X as being *inaccessible*, at least partly, by feasible computations. Thus, a different definition of $H(X|Y)$ is clearly needed to express such possibilities.

We do not know whether efficient algorithms exist for the particular problem discussed above (it is NP-hard, see [24]). It is possible to construct examples where the phenomenon can be rigorously shown to exist.

Let us try to capture the notion of *effective conditional entropy*. Consider a source ensemble $S = \langle S_n \rangle$ over alphabet Σ. A *random variable* X on S is a sequence $\langle X_n \rangle$, where X_n is a random variable on S_n (regarded as a probability space) with values of X in Σ^+. Suppose the source S_n emits a sequence of n^k output symbols $\alpha_1, \alpha_2, \ldots, \alpha_{n^k}$, and person A is told the value of $X(\alpha_i)$ and $Y(\alpha_i)$ for $1 \leq i \leq n^k$, while person B is only told the value of $Y(\alpha_i)$ for $1 \leq i \leq n^k$. Now, if A wants to inform B of the value of $X(\alpha_i)$ for $1 \leq i \leq n^k$, what is the minimum average number of bits A has to send to B? Note that the question reduces to the question one faces in defining effective entropy, when Y is a constant and $X(\alpha_i) = \alpha_i$.

We will just state these definitions informally, as the details are similar to those in Definitions 2-4. A *(t,k)-encoding of X relative to* Y is a triplet of probabilistic algorithms $\mathcal{M} = (M_A, M_B, M_C)$, where the encoder M_A takes input $\alpha = (n, x_1, x_2, \ldots, x_{n^k}, y_1, y_2, \ldots, y_{n^k})$ and computes some binary string β, which if input together with n and $y_1, y_2, \ldots, y_{n^k}$ to the decoder M_B, will enable M_B to recover the string $x_1, x_2, \ldots, x_{n^k}$ with error probability $O(v(n))$; furthermore the codewords β are uniquely decipherable by algorithm M_C; the algorithms M_A, M_B, M_C all halt in $O(n^t)$ time. Let $\ell_n(\mathcal{M}; X|Y)$ denote the average value of $|\beta|/n^k$, that is, the average number of bits used by $\mathcal{M}$ to encode a value x of X. A *(t,k)-conditional-entropy sequence* for $X|Y$ is a sequence $\langle w_n \rangle$ such that there exists a (t,k)-encoding of X relative to Y with $\ell_n(\mathcal{M}; X|Y) = w_n$. We use the abbreviations $H_C(X_n|Y_n) \leq g(n)$, $H_C(X_n|Y_n) = g(n) + O(h(n))$

etc. in the same way as in Definition 4. We will use the term *effective conditional-entropy of* $X|Y$ for $H_C(X|Y)$.

We now turn to the question of mutual information.

Definition 17. Let $g(n)$ and $h(n)$ be any functions. We will write $I_C(X_n|Y_n) \leq g(n)$ if the following is true: For any conditional-entropy sequence $\langle w_n \rangle$ for $X|Y$, there exists an entropy sequence $\langle w'_n \rangle$ for X such that $w'_n \leq w_n + g(n)$ for all sufficiently large n. Expressions such as $I_C(X_n|Y_n) = O(g(n))$, $I_C(X_n|Y_n) \geq h(n)$, etc. can be similarly defined.

In general, $I_C(X_n|Y_n)$ is not symmetric, in contrast to the case of classical mutual information. In fact, this asymmetry property is essential for the possibility of public key cryptography (see § 5 Part 2). In one important special case, however, I_C is nearly symmetric.

Theorem 5. $I_C(X_n|Y_n) = O(v(n))$ if and only if $I_C(Y_n|X_n) = O(v(n))$.

Definition 18. X and Y are said to be *effectively independent* if $I_C(X_n|Y_n) = O(v(n))$

In the classical case, there is an alternative description of independence, namely, $Pr\{X = x, Y = y\} = Pr\{X = x\} \cdot Pr\{Y = y\}$. Is there an analogue? Let us define two new source ensembles $S' = \langle S'_n \rangle$ and $S'' = \langle S''_n \rangle$ for given X and Y. The source S'_n probabilistically outputs a string (x, y) by the following process: Let S_n generate probabilistically an output α, then define $x = X(\alpha)$, $y = Y(\alpha)$. The source S''_n outputs (x, y) by the following process: Let S_n generate independently two output strings α_1, α_2 and define $x = X(\alpha_1)$, $y = Y(\alpha_2)$. Let us write S' as $S(X + Y)$, and S'' as $S(X \times Y)$.

Theorem 6. Let X and Y be random variables on a source ensemble S. Then X and Y are effectively independent if and only if $S(X+Y)$ and $S(X \times Y)$ are indistinguishable.

Theorem 6 implies that any polynomial-time test must fail to detect any correlation between effectively independent X, Y for all sufficiently large n. It also implies that observing $y_1, y_2, \ldots, y_{n^k}$ will lend no noticeable advantage to predicting the value of any function of $x_1, x_2, \ldots, x_{n^k}$.

7. A Theory For Cryptography

In [23] Shannon develped a mathematical theory for cryptography based on information theory. With the tools we have develped, we are ready to give an alternative foundation based on computational complexity theory.

For lack of space, we will give in this abstract only one elementary illustraion. Consider a conventional cryptographic system where two users A and B share a secret key K form a large key space $\mathcal{K}$. Let $E(K, M)$ be the encryption algorithm and $D(K, M)$ the decryption algorithm, i.e., $D(K, E(K, M)) = M$. Let p and q be the probability distributions for message M and key K, respectively. Suppose a eavesdropper taps the line and obtains a ciphertext $J = E(K, M)$. How much can he find out about the plaintext M?

Shannon discussed this situation in [23]. Consider K, M, J as random variables. Then $I(M, J) = H(M) - H(M|J)$ will be the amount of information about M which the eavesdropper obtains. An *unconditionally secure* system is one in which $H(M) = H(M|J)$. It was pointed out that unconditional security cannot be achieved if $H(K) < H(M)$. Thus, if we have 200-bit long keys and 10^6-bit messages, then most likely we cannot have an unconditionally secure system. However, Shannon pointed out that the computational complexity aspect should be taken into account . Let us see how we can approach this problem.

As our theory deals with asymptotic behavior, it only applies to cryptographic systems that can be scaled up or down easily. For definiteness, assume that for each n, the system has n-bit keys K_n, n^4-bit messages M_n and a pair of encription-decription functions $E_n(K_n, M_n)$, $D_n(K_n, J_n)$. The key K_n and the message M_n are distributed according to some probability distributions p_n and q_n. Let us consider the probability space Q_n defined as follows: The sample space is the set of all possible values of $(K_n, M_n, E_n(K_n, M_n))$, and $p_n(K_n = k) \cdot q_n(M_n = m)$ is the probability assigned to the point $(k, m, E_n(k, m))$. Let $Q = \langle Q_n \rangle$, then K, M and $J = E(K, M)$ become random variables on the source ensemble Q. We define the *computational security* of the system by the requirement that $H_C(M) \approx H_C(M|J)$, or $I_C(M_n|J_n) = O(v(n))$. That is, a system is said to be *computationally secure* if the random variables M and J are computationally independent. By the discussions at the end of last section, an eavesdropper on a secure system cannot learn anything about M from the ciphertext when n is large.

Part 2: Trapdoor Functions and Applications

1. Introduction

The concept of one-way functions and trapdoor functions have been suggested by Diffie and Hellman [7] as the foundation for a new type of cryptography. Since then many implementations and applications have been found. However, the question "what is a trapdoor function?" has so far not been answered satisfactorily. The purpose of Part 2 is to propose precise definitions for one-way and trapdoor functions, based on the computational information theory developed in Part 1, and to present improved results and new applications. Concretely, we will show that any trapdoor function can be used to produce a secure encryption scheme as defined in Part 1, and perhaps more surprisingly, can be used to generate "perfect" pseudorandom numbers that will pass any feasible statistical tests. We also give a new function that is a trapdoor function, assuming factorization of large integers is computationally infeasible. Finally an interesting implication of the existence of one-way functions on abstract complexity theory will be presented.

2. BACKGOUND

2.1 ENCRYPTION

Diffie and Hellman [7] invented the concept of *public key encryption* for sending secret messages. In this scheme, every user A puts a *public key* K_A to an *encryption function* E_{K_A} on the public file, and keeps a *private key* K'_A to a *decryption function* $D_{K'_A}$ as private inifor-mation. The pair (K_A, K'_A) has the property that $D_{K'_A}(E_{K_A}(x)) = x$ for any x in the domain of E_{K_A}. Anyone wishing to send a message x to A can encrypt it as $E_{K_A}(x)$, and A then recovers x by applying $D_{K'_A}$ to $E_{K_A}(x)$. For the scheme to work properly, the following properties should be satisfied:

(a) Given K_A and x, the value of $E_{K_A}(x)$ should be easy to compute; given K'_A and y, the value of $D_{K'_A}(y)$ should be easy to compute;

(b) Given $E_{K_A}(x)$, it is computationally difficult to find x;

(c) A random pair of (K_A, K'_A) should be easy to generate.

Since these functions $E_{K_A}(x)$ are easy to compute but hard to invert, they are called *trapdoor* functions.

A concrete implementation was suggested by Rivest, Shamir and Adleman (the RSA scheme [20]). In their scheme, user A generates two random primes p and q, and an integer s such that $\gcd(s, \phi(N)) = 1$, where $N = p \cdot q$ and $\phi(n)$ is the Euler totient function. Let N be the public key K_A, and $r = s^{-1} \bmod \phi(N)$ be the private key K'_A. The function $E_{K_A}(x) = x^s \bmod N$ serves as the encryption function, while $D_{K_A}(x) = x^r \bmod N$ serves as the decryption function.

A variation of the RSA scheme was suggested by Rabin [19]. Instead of using $x^s \bmod N$, he defines $E_{K_A}(x) = x^2 \bmod N$, and decryption is easy for user A who knows the factorization of N. An interesting fact is that a successful inversion of $E_{K_A}(x)$ for any ϵ fraction of the x's will enable one to factor N in random polynomial time.

Two potential problems with the above schemes were raised in Goldwasser and Micali [10]:

(a) Even though $E_{K_A}(x)$ is hard to invert in general, it may be easy when x is of a special form;

(b) One may be able to infer some partial information about x from $E_{K_A}(x)$.

They proposed a scheme which is free from these difficulties. The scheme encrypts the entire message bit by bit, where the encryption of a single bit is based upon the complexity of deciding whether an integer y is a quadratic residue mod$N$ or not. We will outline the method here.

Let $N = p \cdot q$ be the product of two n-bit primes. Denote by Z_N^* the set $\{x \mid 1 \leq x \leq N, \gcd(x, N) = 1\}$, and let $A_N^* \subseteq Z_N^*$ be the subset consisting of those x's with Jacobi symbol $(x/N) = 1$. Half of the members of A_N^* are quadratic residues, and half are non-residues.

User A generates a random n-bit composite integer $N = p \cdot q$ of two primes, and puts N together with a non-residue $y \in A_N^*$ in the public file. Suppose B wants to send a bit b to A. Then B will randomly generate an $x \in Z_N^*$ and send $x^2 \bmod N$ if $b = 0$, and $yx^2 \bmod N$ if $b = 1$. Thus, A can decide if $b = 0$ or 1 by finding out whether the number received is a quadratic residue or not. An eavesdropper not knowing the factorization of N will have difficulty deciding what b is.

Schemes like this produce stochastically a ciphertext for a given message, and are called *probabilistic encryptions* [10].

Goldwasser and Micali [10] showed that under Assumption GM below, for any fixed $0 < \epsilon < 1$, an adversary will not be able to guess correctly the value of b with probability $1/2 + \epsilon$ or more.

Definition 19. Let $C_{n,\epsilon}$ be the minimum size of any circuit that decides correctly quadratic residuosity mod N for a fraction ϵ of all n-bit integers N with two prime factors.

Assumption GM: $C_{n,\epsilon} > Q(n)$ asymptotically for any fixed polynomial Q and any fixed $0 < \epsilon < 1$.

2.2 PSEUDORANDOM NUMBER GENERATION

The use of pseudorandom number generators as an approximate one-time pad is a common mode of secret communication. Shamir [21] considered the following problem. Suppose two persons A and B share a common secret seed s, and use a common pseudorandom number generator. For A to send blocks of plainatext $y_1, y_2, \ldots$ to B, A can use s to generate a sequence of pseudorandom numbers $x_1, x_2, \ldots$, and send ciphertext $x_1 \oplus y_1, x_2 \oplus y_2, \ldots$ to B. These can be decoded easily by B since B can generate the sequence $x_1, x_2, \ldots$. Now imagine the situation when an adversary has some side information on the plaintext which enables him to find out a few initial values of $x_1, x_2, \ldots$. Based on these values, the adversary may be able to generate the rest of the x-sequence, and thus break the ciphertext. Shamir asked the question: Can one design a pseudorandom number generator with the property that, even knowing $x_1, x_2, \ldots, x_k$, it is still hard to compute x_{k+1}? Based on the assumption that the RSA scheme is secure, Shamir gave a pseudorandom number generator that satisfies this requirement.

Shamir's scheme does not guarantee that the next bit is difficult to compute (even though the next *word* is). Blum and Micali [5] recently constructed a stronger pseudorandom number generator based on the discrete logarithm problem. (They also gave a criterion on one-way functions under which this construction will work.) Let $k > 0$ be any integer. The generator takes $O(n)$ true random bits and generates a sequence $x_1, x_2, \ldots, x_{n^k}$ that, under Assumption BM below, has the following property: Let $G_{n,j,\epsilon}$ be the minimum size of any circuit with j boolean inputs that can output the value of x_{j+1} correctly with probability $\frac{1}{2} + \epsilon$ or more, when the values of $x_1, x_2, \ldots, x_j$ are

input to the circuit. Then $G_{n,J,\epsilon} > Q(n)$ asymptotically for any fixed polynomial Q.

To describe the assumption needed, let p be a prime and g a generator of the cyclic multiplicative group $A_p = \{1, 2, \ldots, p-1\}$. For $y \in A_p$, let $f(p, g, y)$ be the value $x \in A_p$ such that $g^x \bmod p = y$. A boolean circuit with three n-bit inputs a, b and c is said to *solve the discrete logarithm problem for* p if, for all g and y its output is equal to $f(p, g, y)$ when $a = p, b = g$ and $c = y$. Let $I_{n,\epsilon}$ be the minimum size of any circuit that solves the discrete logarithm problem for ϵ fraction of all n-bit prime numbers p.

Assumption BM: $I_{n,\epsilon} > Q(n)$ asymptotically for any fixed polynomial Q and any fixed $0 < \epsilon < 1$.

2.3 Discussions

In the context of the preceding review, there are several questions that seem to merit further consideration.

(a) The probabilistic encryption scheme in [10] and the pseudorandom sequence generation in [5] both utilize special properties of the one-way functions employed. Are there general procedures that can utilize any trapdoor functions for the purpose of encryption and generating pseurandom numbers?

(b) The pseudorandom number sequence in [5] is bitwise unbiased for any fixed $\epsilon > 0$, and thus solves the question posed by Shamir. However, a cryptanalyst may adopt a different procedure to analyze the pseudorandom number generator. For example, he may work backwards, examining the last few bits in an attempt to reconstruct the preceding bits. (Indeed, it is still open whether the sequence generated by the scheme in [5] enjoys the same unbiased property if read in reverse.) Can one construct a pseudorandom sequence that can withstand *any* attempt by the cryptanalyst to break the pseudorandom sequence? In a broader context, can one construct pseudorandom sequence that can be used for applications other than cryptography?

(c) Can one weaken the assumptions in Assumption BM and Assumption GM such that, instead of requiring that no algorithm can solve ϵ fraction of the instances for any fixed ϵ, we require only that no algorithm can solve say, 1/2 of the instances?

We will give positive answers to all the above questions in the next two sections.

3. One-way Functions

In this section we formalize the notion of one-way functions, and show how they can be used to construct perfect pseudorandom number generators and to implement secure conventional cryptosystems.

It is helpful to think of a one-way function f as a 1-1 function used in a puzzle Z. In each game, Z picks a random n-bit number x according to some distribution $p_n(x)$, and shows us the value of $f(x)$. We are challenged to find x, with a button to ask for help. The function f will be one-way if we end up asking for help a fraction the time.

Definition 20. Let $P = (p_n)$ be a sequence of probability distributions on Σ^+, where the support of p_n (i.e., the set $\{x \mid p_n(x) \neq 0\}$) consists of only strings of length $\theta(n)$. We will call P a *polynomial-time distribution ensemble* if there exists a probabilistic algorithm which, given input n, will halt in time polynomial in n, and output x with a probability distribution $h(x)$ satisfying $\|h - p_n\| = O(v(n))$, where Notation. $\|h - p_n\| = \sum_x |h(x) - p_n(x)|$.

Definition 21. Let f be a 1-1 function from V to Σ^*, where $V \subseteq \Sigma^*$. Let $P = (p_n)$ be a polynomial-time distribution ensemble such that the support of every p_n is a subset of V. Define the *correlation ensemble of f under P* to be the source ensemble $Q^{f,P} = (Q_n)$, where source $Q_n = (T, q_n)$ has sample space $T = \{(x, f(x)) \mid x \in V\}$, and distribution $q_n(\tau) = p_n(x)$ for $\tau = (x, f(x))$. Let $X^{f,P} = (X_n)$ and $Y^{f,P} = (Y_n)$ be two random variables on $Q^{f,P}$ defined by $X_n(\tau) = x$ and $Y_n(\tau) = f(x)$ for $\tau = (x, f(x)) \in Q_n$.

Definition 22. A 1-1 function f is said to be *one-way* if there exists a polynomial-time distribution ensemble P such that $H_C(X_n^{f,P} \mid Y_n^{f,P}) = \Omega(1/n^t)$ for some fixed t.

The gist of the definition is that, some fbits of information are needed in order to recover x from $f(x)$ under the distribution p_n. The next theorem gives an alternative description directly in terms of complexity.

Definition 23. Let f and P be as in Definition 21. Define $P^f = (p_n^f)$, where $p_n^f(y) = p_n(f^{-1}(y))$.

Theorem 7. A 1-1 function f is one-way if and only if there exists a polynomial-time distribution ensemble P such that the following is true: For any probabilistic algorithm which, on input n and y, with y distributed according to $p_n^f(y)$, halts in time polynomial in n and outputs x, the probability of $x \neq f^{-1}(y)$ is $\Omega(1/n^t)$ for some t.

One-way functions become more interesting when they possess some additional properties. For example, if a one-way function has a certain invariance property, then it can be used to construct a perfect pseudorandom number generator, and hence a secure conventional cryptosystem; if it possesses a *key* plus an *inverse key*, then it becomes a trapdoor function and can be used to build a probabilistic public-key cryptosystem.

One-way Functions with Invariance Properties

Definition 24. If in Definition 22, we also have $P^f = P$, then we call f a *stable* one-way function.

Theorem 8. Any stable one-way function f can be used to construct a perfect pseudorandom number generator G_f.

Remark. The construction is explicit in the sense that, if the descriptions of all the relevant probabilistic algorithms for f and P are given then the description of G_f is immediate.

Corollary to Theorem 8. Any stable one-way function f can be used to construct a computationally secure cryptosystem. (See § 7 of Part 1)

Remark. A seed for the pseudorandom number generator is used as the common private key. This is the computationally-secure realization of the one-time pad.

One-way Functions with Keys

We will use the discrete logarithms problem as an example to illustrate our result.

Example 3. *Discrete Logarithm Function*

Let $G_n = \{(p, g, m) \mid p$ is an n-bit prime, g a generator of A_p, and $m \in A_p\}$. Define $V = \bigcup_n G_n$, and $f : V \mapsto V$ by $f(x) = (p, g, g^m \bmod p)$ if $x = (p, g, m)$.

We regard (p, g), the part of x that remains unchanged under f, as the *key*. It is useful to consider those distributions P that generate x in two steps: first the key (p, g), and then the remaining part m. One such P is given by $\langle p_n \rangle$ defined below. Let

$$p'_n(K) = \frac{1}{(\text{no. of n-bit primes}) \cdot (\text{no. of generators in } A_p)}$$

if $K = (p, g)$ is a possible key, and $p'_n(K) = 0$ otherwise. Let $p''_{n,K} = 1/(p-1)$ if $m \in A_p$, and 0 otherwise. Finally, define $p_n(x) = p'_n(K) \cdot p''_{n,k}(m)$ for $x = (K, m)$.

Notation. We will extend Definition 20 and say that the sequence $\langle p''_{n,K}(m) \rangle$ is a *polynomial-time distribution ensemble* if $p''_{n,K}(m) \neq 0$ implies $m = \theta(n)$, and if there exists a probabilistic algorithm which, given n and K, halts in polynomial time in n and outputs a string z whose probability distribution $h(z)$ satisfies $\|h - p''_{n,k}\| = O(v(n))$.

Definition 25. A one-way function f is said to have a *key* if the domain V of f has the form $V \subseteq \Sigma_1^* \times \Sigma_2^*$ and there exists a polynomial-time distribution ensemble $P = \langle p_n \rangle$ with the following properties:

(a) P makes f a one-way function as in Definition 22,

(b) $p_n(x) = p'_n(K) \cdot p''_{n,K}(z)$ if $x = (K, z)$, where $\langle p'_n \rangle$ and $\langle p''_{n,K} \rangle$ are polynomial-time distribution ensembles,

(c) $f(x)$ can be written in the form $(K, f''_K(z))$ for $x = (K, z)$.

4. Trapdoor Functions And Encryption

A *trapdoor function* is basically a one-way function with a key K such that an inverse key K' can be easily created at the same time, but K' cannot be inferred form K. We will not give the formal definition in this abstract.

A *probabilistic public-key cyrptosystem* (PPKC) is a cryptosystem in which user A has an n-bit key K in the public file, while keeping a key K' as private information. To send a bit $b \in \{0, 1\}$, B will use b and K to compute probabilistically a string x and send it to A. The keys are generated in such a way that, with the knowledge of K', A can uniquely recover b.

Let us call a PPKC *secure* if, for b with bias $p \geq 1/2$, an adversary cannot guess from K and x in polynomial time the value of b correctly with probability more than $p + O(v(n))$.

Theorem 9. Any trapdoor function can be used to construct a secure PPKC.

Factoring large composite numbers is a problem which has a long history of resisting efficient solutions. It would therefore be a most appropriate basis for constructing PPKC or pseudorandom number generators. Below we present a new trapdoor function, which is also a stable one-way function, assuming factoring is hard in the proper sense to be described below. Let T_n be the set of all integers $N = p \cdot q$, where p and q are n-bit primes with $p \equiv q \equiv 3 \bmod 4$.

The Intractability Assumption of Factoring:

For any polynomial-time probabilistic algorithm M that tries to factor integers, there exists an n_0 such that M will fail to factor at least $1/n^{10}$ of the members of T_n for all $n \geq n_0$.

Remark. One can replace $1/n^{10}$ by any $1/n^t$ with any predetermined t.

A Factoring Trapdoor

We generate a pair of keys (K, K') by generating two random n-bit primes p, q with $p \equiv q \equiv 3 \bmod 4$, and setting $K = N (\equiv p \cdot q)$ and $K' = \{p, q\}$. The trapdoor function f is defined by $f(N, z) = z^2 \bmod N$ if z is even, and $f(N, z) = -z^2 \bmod N$ if z is odd, for $z \in A_N^*$.

It is of interest that the factoring trapdoor function defined above also leads to a simple pseudorandom number generator. It is possible to prove that this generator is perfect if the quadratic residuosity problem is hard.

A Quadratic-Residue Pseudorandom Number Generator

Let $n > 0$ and $k > 0$. We will describe how to generate a sequence β of n^k bits using $O(n(\log n)^2)$ true random bits.

We first describe a procedure that generates a *quasi-random sequence* α as follows: Pick a random N from T_n, and a random m with $1 \leq m \leq N - 1$, then compute the following sequence of number $z_1, z_2, \ldots, z_{n^k}$ by $z_1 = m$, $z_{i+1} = f(N, z_i)$. Let $\alpha = \alpha_1 \alpha_2 \cdots \alpha_{n^k}$ where α_i is just the parity of z_i.

To obtain β, we repeat the above procedure to obtain $t = (\log_2 n)^2$ quasi-random strings $\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(t)}$. Now let $\beta = \alpha^{(1)} \oplus \alpha^{(2)} \oplus \cdots \oplus \alpha^{(t)}$ where $\oplus$ denotes bitwise XOR. For fixed k and large n, the pseudorandom sequence β so generated will be indistinguishable from a true random sequence assuming that quadratic residuosity is hard.

5. What Makes the Trapdoor Work?

It is of interest to give an in-depth view of the results in the previous two sections. As we will see, these results can be deduced naturally, if one keeps track of the computational information contained in the various strings and considers how the information can be manipulated. In this abstract, we will give such an analysis to the encryption by using a trapdoor function.

For definiteness, consider a one-way function f that maps n-bit strings to n-bit strings for each n, and let (p_n) be the distribution ensemble that makes f one-way. For a given n, f can be viewed as a channel with large alphabets, in fact with 2^n input symbols and 2^n output symbols. Thus, f is a non-classical channel, and we have to consider accessible information instead of the Shannon information.

Let (X_n, Y_n) denote the pair of random variables corresponding to $(x, f(x))$ where x is distributed according to p_n. Let us dramatize the situation by assuming that party A is on the input side of the channel f sending string x with probability $p_n(x)$ across the channel to party B, who receives the string $f(x)$. Clearly, the Shannon conditional entropies $H(Y_n|X_n)$ and $H(X_n|Y_n)$ are both 0, and classically f is a noiseless channel. However, when we consider accessible information, we find the $H_C(Y_n|X_n)$ is still 0, but $H_C(X_n|Y_n)$ is $\Omega(1/n^t)$. That is, A has no uncertainty about what B receives, but B has at least some uncertaity about what A has sent, admittedly the uncertainty may be as small as $1/n^t$ fbit. Thus, from B's standpoint, he is on the receiving end of a noisy channel with a noise at least $1/n^t$, a non-neglible quantity in polynomial-time calculations.

Let us now visualize the above picture in a slightly different way. Assume that f is in fact a trapdoor function, and a third party G is the owner of the secret key, while A is really transmitting x to G as $f(x)$. It is a clear channel to G, since he has the secret key to decode $f(x)$. The role of B now is an eavesdropper, who, without the secret key, is trying to wiretap the line with a low-grade equipment. But this situation has an exact analogue in the classical information theory, known as the Wyner wiretap channel [25]. Wyner showed that even when the noise in B's channel is small, A can magnify the noise by properly encoding his messages. For example, suppose B has a binary symmetric channel with crossover probability 10^{-4}, and A encodes a bit b as a random 10^6-bit string α such that α has an even number of 1's if and only if $b = 0$. Then, of the transmitted bits, B can guess the value of each individual bit with high confidence, yet B knows that he is going to be wrong in the values of about 100 bits. It becomes difficult to estimate whether he has missed an even number of bits or an odd number. Indeed it was shown that schemes of the kind indeed will completely baffle B about the true value of b.

In our case, the noise is somewhat like n^{-t}, and the approximate estimate on the length of α is about n^t, which is large but can be accomplished in polynomial time.

It is also of interest to follow in detail the information accounting in pseudorandom number generations; we will leave that to the full paper.

6. A Theorem In Abstract Complexity Theory

Let R be the class of decision problems solvable in random polynomial time, as defined in Adleman [1]. The relationship between R and the deterministic complexity hierarchy has been a subject of considerable interest (Adleman [1], Aleliunas, et. al. [3], Bennett and Gill [4], Gill [9]). Two of the well-known results are that, any decision problem in R can be computed by a polynomial-size boolean circuit [1], and that the obvious relation $R \subseteq \bigcup_{\epsilon>0} DTIME(2^{n^\epsilon})$ holds, where $DTIME(g(n))$ is the class of problems slovable in deterministic time $g(n)$. In this section, we will prove a stronger form of the latter relation under the assumption that one-way functions exist. (The definition of one-way functions in this section will be somewhat different from that of the previous sections.)

Let f be a 1-1, onto function defined on a subset of $\{0,1\}^*$. Suppose $P = (p_n)$ is a polynomial-time distribution ensemble invariant under f. A boolean circuit C is said to *invert f under p_n* if $\sum_{y \in T} p_n(y) > 1/2$, where T is the set of input strings y for which C gives the output $f^{-1}(y)$. Let $B_n(f,P)$ be the size of the smallest circuit that inverts f under p_n.

Definition 26. f is said to be a *strong one-way function* if there exists a P such that $B_n(f,P) > Q(n)$ asymptotically for any polynomial Q.

The main theorem of this section is the following.

Theorem 10. If there exists any strong one-way function, then

$$R \subseteq \bigcap_{\epsilon>0} DTIME(2^{n^\epsilon}).$$

This formula means that, for any decision problem in R and any $\epsilon > 0$, there is a deterministic Turing machine solving it in time $O(2^{n^\epsilon})$. It is of interest to note that this is a case where a lower bound on non-uniform complexity has consequences on the upper bound of uniform complexity (cf. Karp and Lipton[], Pippenger [17], Pippenger and Fischer [18]).

The hypothesis of Theorem 10 will be satisfied if either the factoring of integers is hard or the discrete logarithm problem is hard in the appropriate sense. Let $F(n)$ be the minimum size of any boolean circuit that can factor 4/5 of the $2n$-bit composite numbers N with two n-bit prime factors.

The Strong Intractability Assumption of Factoring: $F(n) > Q(n)$ asymptotically for any fixed polynomial Q.

Corollary to Theorem 10. Under the strong intractability assumption of factoring,

$$R \subseteq \bigcap_{\epsilon>0} DTIME(2^{n^\epsilon}).$$

It is clear that Theorem 10 is also true if the discrete logarithm problem is hard in the sense of Assumption BM. However, a much weaker assumption suffices in this case. We will state that result below as Theorem 11.

Definition 27. Let p be an n-bit prime number, and let g be any generator of A_p. The *discrete logarithm problem* $D_{p,g}$ is: Given input $y \in A_p$, find x such that $g^x \bmod p = y$.

Let $\ell(D_{p,g})$ be the minimum boolean circuit size for solving $D_{p,g}$. Define $L(n)$ to be $\max\{\ell(D_{p,g}) \mid \log_2(p+1) \leq n\}$. We make the following assumption:

The Intractability Assumption of Discrete Logarithm: $L(n) > Q(n)$ asymptotically for any fixed polynomial Q.

Note that this assumption is weaker than Assmuption BM. In fact, it is not concerned with average-case complexity, unlike the rest of this paper.

Theorem 11. Under the intractability assumption of discrete logarithm, we have

$$R \subseteq \bigcap_{\epsilon>0} DTIME(2^{n^\epsilon}).$$

The discrete logarithm problem is a classical number-theoretic problem, for which no effcient algorithm is known. The intractability assumption of this problem, in one form or another, has been the basis of several cryptographic protocols (Diffie and Hellman [7], Blum and Micali [5]). So far the best algorithm known [2] runs in time roughly $2^{\sqrt{n}}$. If the discrete logarithm problem has in fact a complexity much higher than polynomial, then we can obtain results stronger than Theorem 2. For example, one has:

Theorem 12'. If $L(n) > 2^{n^\epsilon}$ for some fixed $\epsilon > 0$ and for all n, then $R \subseteq DTIME(2^{(\log n)^c})$ for some constant $c > 0$.

References

[1] L. Adleman, "Two theorems on random polynomial time," Proc. 19th IEEE Symp. on Foundations of Computer Science, Ann Arbor, Michigan, Oct. 1978, 75-83.

[2] L. Adleman, "A subexponential algorithm for the discrete logarithm problem with applications to cryptography," Proc. 20th IEEE Symp. on Foundations of Computer Science, Puerto Rico, Oct. 1979, 55-60.

[3] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovasz, C. Rachoff, "Random walks, universal sequences, and the complexity of maze problems," Proc. 20th IEEE Symp. on Foundations of Computer Science, Puerto Rico, Oct. 1979, 218-223.

[4] C. H. Bennett and J. Gill, "Relative to a random oracle, PA = NPA = co-NPA with probability 1," SIAM J. on Computing 10 (1981), 96-113.

[5] M. Blum and S. Micali, "How to generate cryptographically stron sequences of pseudo random bits," this proceedings.

[6] G. Chaitin, "A theory of program size formally identical to information theory," Journal of ACM 22 (1975), 329-340.

[7] W. Diffie and M. E. Hellman, "New directions in cryptography," IEEE Trans. on Inform. Theory IT-22, 6 (1976), 644-654.

[8] R. Gallager, Information Theory and Reliable Communication, Wiley, New York, 1968.

[9] J. Gill, "Computational complexity of probabilistic Turing machines," SIAM J. om Computing 6 (1977), 675-695.

[10] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information," Proc. 14th ACM Symp. on Theory of Computing, San Francisco, May 1982.

[11] J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, Mass., 1979.

[12] R. M. Karp and R. J. Lipton, "Some connections between nonuniform and uniform complexity classes," Proc. 12th ACM Symp. on Theory of Computing, Los Angeles, April 1980, 302-309.

[13] D. E. Knuth, The Art of Computer Programming, Vol. 2, Addison-Wesley, Reading, Mass., second edition, 1981.

[14] A. N. Kolmogorov, "Three approaches to the concept of the amount of information," Probl. Pered. Inf. (Probl. of Inf. Transm.) 1/1 (1965).

[15] P. Martin-Lof, "The definition of random sequences," Information and Control 9 (1966), 602-619.

[16] A. R. Meyer and E. M. McCreight, "Computability complex and pseudorandom zero-one valued functions," in Theory of Machines and Computations, Z. Kohavi and A. Paz, eds., Academic Press, New York 1971, 19-42.

[17] N. Pippenger, "On simultaneous resource bounds," Proc. 20th IEEE Symp. on Foundations of Computer Science, Puerto Rico, Oct. 1979, 307-311.

[18] N. Pippenger and M. J. Fischer, "Relations among complexity measures," Journal of ACM 26 (1979), 361-381.

[19] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," MIT/LCS/TR-212, 1979.

[20] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of ACM 21 (1978), 120-126.

[21] A. Shamir, presented at Crypto-81, Santa Barbara, 1981.

[22] C. E. Shannon, "A mathematica theory of communication," Bell System Technical Journal, 27 (1948), Part I, 479-523, Part II, 623-656.

[23] C. E. Shannon, "Communicatin theory of secrecy systems," Bell System Technical Journal 28 (1949), 656-715.

[24] L. Valiant, "The complexity of computing the permanent," Theoretical Computer Science 8 (1979), 189-201.

[25] A. D. Wyner, "The wire-tap channel," Bell System Technical Journal 54 (1975), 1355-1387.

[26] J. Ziv, IEEE Transaction on Information (1965).

[27] A. K. Zvonkin and L. A. Levin, "The complexity of finite objects and the algorithmic concepts of information and randomness," Uspekhi Mat. Nauk (Russian Math. Surveys 25/6 (1970), 83-124.