Knowledge and Efficient Computation

by
Silvio Micali
Computer Science Department
MIT
545 Technology Square
Cambridge, MA 02139

Abstract

We informally discuss "knowledge complexity": a measure for the amount of knowledge that can be feasibly extracted from a communication. Our measure provides an answer to the following two questions:

1) How much knowledge should be communicated for proving a theorem?

2) How to prove correctness of cryptographic protocols?

We sympathize with the readers who are distressed by the level of informality of this short abstract. Most of the material is contained in the reference [GMR]. We encourage the readers to consult it for precise definitions.

1. Knowledge Complexity

Everyone would agree that communication is *the* tool for transfering or exchanging knowledge. This, however, does not answer basic questions like

Which communications convey knowledge?

How much knowledge is contained in a communication?

knowledge Complexity, a notion introduced in [GMR], provides an answer to these questions in a framework where computation is bounded. In sections 2 and 3 we indicate how these ideas can be applied to two contexts in which knowledge is particularly relevant: theorem proving and cryptographic protocols.

1.1 The Basic Scenario

In our framework communications that convey knowledge are those transmitting the result of an infeasible computation, therefore a computation that we cannot perform by ourselves. Before proceding any further, we have to settle the question of which computations should be considered infeasible. Theoretical computer science regards as "feasible" those computations that can be performed in polynomial time (i.e. in time polynomial in the length of the input) and as "infeasible" those requiring, say, exponential time. (1)

Recently, coin tossing has been proved useful for efficient computation (see for example the probabilistic primality tests of Solovay and Strassen [SS], Rabin [R] and Goldwasser and Killian [GK]). Since the ability of flipping a coin is common to anyone, we will consider infeasible those computations that cannot be performed in probabilistic polynomial time (i.e. in polynomial time and making random choices as well).

Our scenario consists of two "agents" (read Turing machines) A and B and an input x known to both of them. We think of A as the "communicator" and of B as the "receiver". They "talk" back and forth about x. (In a more restricted scenario, as in the case of radio broadcasting, A will be the only one to speak and B will only listen.) As we are interested in quantifying how much knowledge can be "efficiently extracted" from a communication, B is bound to compute in probabilistic polynomial time, while A has no restrictions on its computational power. In many relevant cases, however, A may also be bound to feasible computations, but happens to possess more "insights" about x. One such case is discussed in the next sub-section and more natural examples arise in the context of section 3.

Our scenario is deceivingly resemblant the one of Information Theory. There A is the *only* witness of some event, the occurrence of which it communicates to B. More precisely, B knows the probability of the possible events,

(1) It is apparent that the computation time necessary for solving a problem depends on the size of the problem, i.e. the number of bits necessary for describing it. For instance, multiplying two integers will require more time for longer integers than for shorter ones. The running time is thus considered as a function of the input length, usually denoted by k. A problem is solvable in polynomial time if there is a constant c and an algorithm (Turing machine) that solves of its instances of size k within time (number of steps) k^c. It may seem arbitrary to equate "feasibility" with polynomial time, as k needs to be pretty big before k^{50} (that is a particular polynomial running time) becomes less than, say, $k^{\log(\log k)}$ (that is a non-polynomial running time). However, there are many reasons justifying the choice of polynomial time as "efficient time". For example, polynomial time is a robust notion in that it appears to be independent of the particular computational model (computer) used. Another advantage is that the composition of two feasible computations is itself a feasible computation. This is so as "polynomial of polynomial is polynomial", very much the same way as the finite union of finite sets is finite.

but is totally unaware of which specific event actually occurred.

One fundamental difference between other models and ours is that x is a common input. Another difference is that B is only capable of feasible computations and cannot, for instance, derive all the logical consequences of the information in its hand. (2)

Let, for instance, B be a scientist (bound, as a human, to feasible computation only). Let the common input x be Nature (that indeed is under everybody's eyes) and A be an "angel" willing to reveal some of nature's best kept secrets. The angel and the scientist will discuss about x. It is in this scenario that we address the question of how much knowledge has the scientist gained by this conversation. Given what we said above, answering is easy in at least two cases. If A sends to B n random bits, for example, though this will be n bits of information, it would be no *knowledge* because B could easily generate random bits by himself. Similarly, the result of any probabilistic polynomial-time computation will not contain any knowledge.

We do not present here a formal and general upper bound (expressed in bits) for the amount of knowledge that can be communicated to a polynomial time receiver. A precise formulation of the new measure relies on many technical details and is unsuitable for such a short abstract. We will, however, present the motivation behind it and mention the notions necessary to its formalization. This is best done by discussing an example.

1.2 An Example

Assume that a crime x has happened, that B is a reporter and A a police officer. A understands the rights of the press but, for obvious reasons, also tries not to release too much knowledge about x. Should reporter B call the police officer A to know more about x? It depends. If he has probability essentially equal to 1 of efficiently generating, alone at home, in front of his typewriter, the "same" conversations (about this specific crime x) that he might have with A, he should not bother to call: A will give him zero knowledge about x. Assume, instead, that B may, efficiently and by himself (i.e. without A's intervention), generate four conversations about x, one of which (but is not clear which one) is guaranteed to be the one actually occuring if talking to A. In this case, the knowledge that B may receive from A about x cannot exceed two bits. Moreover, this is an absolute upper bound for any kind of knowledge B may receive. Possibly, the knowledge B is interested in is even less. Still, it may pay off to call. If, finally, B has only chance 1 in 2^{100} of generating the possible conversations about x with the police officer by an efficient procedure, then the officer is a real gossiper and B should rush to the telephone!

Let us stress once more that it is crucial that B should be able to "simulate" A efficiently. The ability of generating the conversations in question with probability 1 but in exponential time would be totally useless. Essentially, because B would not live long enough to see the result of his computations! It is the probability of quickly imitating A that essentially measures the "amount of knowledge that A may give to our specific reporter B".

Much more important, however, is the notion of an agent (police officer) that only gives away *at most* a certain amount of knowledge, *no matter with whom it talks*. An honest reporter will not try to get out of A knowledge that is

(2) This will in general make the analysis particularly delicate as polynomial time is a notion easy to define but difficult to use. In fact though progress has been made towards estimating the computational difficulty of problems still very fundamental questions remain open.

not supposed to receive. However, nobody guarantees the officer that he is talking to an honest reporter. Indeed a dishonest and news-hungry B', though still bound to feasible computation, may be able to find out more about x. Despite this capability, if the officer is so skillful to be one who communicates, say, at most 2 bits of knowledge, no matter how tricky questions B' asks and how much he cheats, he will not get out of him more than two bits about x.

Proving that some communicator A, that is programmed to answer certain type of questions, releases at most two bits of knowledge is certainly more difficult. Essentially because a proof must consider all possible polynomial-time strategies for B'. Still, as we point out is section 2 and 3, such proofs have been found in some cases. First, we need to refine the concepts of the previous example.

1.3 Our Example Revisited

In the example above, we talked about the probability with which B, by himself and efficiently, may generate the discussion that it and A may have on input x. However, everyone knows that some randomness is present in every conversation. Let our communicator be the most predictable police officer you can imagine. Still, for no crime the set of his relative possible comments will consist of a single element. Rather, to any possible crime will be associated an *ensemble* of possible trivialities. Guessing exactly which comment will be actually said will be almost impossible, but nevertheless inessential as these comments are all "equivalent". Insisting in predicting individual strings would mislead us toward a wrong definition of knowledge, one in which essentially all communicators would appear releasing enormous amounts of knowledge. Consider a communicator T that, no matter what the input is, always transmits a randomly chosen 100-bit string. It will be pretty hard to correctly guess which string it will send us, but it will be extremely easy to dispose of T and replace it by flipping a 100-bit string ourselves whenever we want its opinion. This leads to the following point of view.

The probabilistic programs (agents) A and B, together with the input x specify a *probability distribution* AB_x, namely the set of all possible conversations of A and B about x. These probability distributions may be extremely complex. What is relevant to our analysis is the computational difficulty of *sampling* AB_x, i.e. picking elements with exactly the probabilities they are assigned in AB_x. Let us revise in this light some of the cases discussed in our example of section 1.3.

Reaching a more appropriate level of generality, we will say that A gives zero knowledge to B if, on input x, B will be able, by himself, to sample AB_x in polynomial time. That is, if B can efficiently select conversations c with exactly the probability distribution with which A and B talking together would select it.

Let us consider the next case. One way in which A gives B at most 2 bits of knowledge is the following: on input x, B can efficiently select 4 conversations, one of which will be selected exactly according to AB_x, though B does not necessarily know which one.

1.4 A Pinch of Operationism (or: One More Visit To Our Example)

To reach a powerful level of generality, we have to further refine the notion of "efficient samplability" of AB_x given in the previous sub-section. There we insisted that B, alone and reasonably quickly, is able to select elements with exactly the same probability they are assigned in AB_x. Why do we need these probabilities to be *exact*? There is no compelling reason. For example, let D_1 be the uniform probability distribution over the set of the 1000-bit strings

and let the distribution D_2 assign equal probability to all 1000-bit string except the string 000...0 (one thousand times) which will be assigned probability 0. Then it will be humanly impossible to distinguish D_1 from D_2 by randomly drawing elements from them. Only after the Universe has ended will we find out that the string 000...0 does not come up with the same frequency when sampling D_2 as when sampling D_1. For all practical considerations, D_1 is equal to D_2.

Above we only discussed a particular way in which two probability distributions appear equal: the two probability distributions assign equal probability to equal strings except for a set whose total probability is negligibly small. However, as the following example suggests, this may not be the only way in which two probability distributions may appear equal. Consider a particular Turing machine M. let S_1^k be the set of the k-bit inputs on which M halts after $2^{\sqrt{k}}$ steps and S_2^k the set of k-bit inputs on which M does not halt. Now choose a particular k with the constraints that it is bigger than a million and that S_1^k and S_2^k contain, say, more than 2^{100} elements. Then, let us consider the uniform probability distribution for S_1^k and for S_2^k. This time the two distributions cannot be close as they are defined on totally disjoint sets. Nevertheless, it is conceivable that, by cleverly choosing M, the two distributions may appear equal from all practical points of view!

The notion that naturally includes all "plausible" ways of two distributions appearing equal (including of course the ones suggested above) is that of *Polynomial Time Undistinguishable Probability Distributions.* The precise definition is given in [GMR]. Here we only intend to outline the point of view behind the definition.

In essence, two objects X and Y can be called distinct only if there is an explicit procedure that tells them apart. At a second glance, what we really need is a procedure that is reasonably fast. If all polynomial time procedures fail to distinguish X from Y, then X and Y are either equal or distinguishable only by "angels". But, for us humans, it is an act of intellectual honesty to consider them equal.

We should rivisit once more our example armed with this new level of generality. In essence, the reporter receives 0 knowledge from the police officer if can efficiently generate a set of conversations that cannot be feasibly distinguished from the text of the conversations that he might have with the officer. In this case, experiencing the "real" conversations with the officer is useless as no other human (somebody who is bound to polynomial time) can find any difference with the fake ones. Whatever the reporter may find in the real conversations or UL "whatever" he can succeed doing with the real conversations, he may also find or succeed in doing with the fake ones.

One may ask at this point why further generalizations are not necessary. The best reassurance that the right formalization has been achieved can be derived from the successful application of these concepts to other fields of interest. Indeed, knowledge complexity enabled us to study some fundamental questions relative to the proving process (see next section) and to prove the correctness of cryptographic protocols, an extremely puzzling and difficult task.

2. The Knowledge Complexity of Theorem Proving Procedures

How much knowledge should be communicated for proving a theorem T?

Certainly enough to verify that T is true. Usually, much more. For example, to prove that a certain a is a quadratic residue mod m (i.e. a square mod m), it is sufficient to communicate an x such that $a \equiv x^2 \bmod m$. This communication, however, contains more knowledge than just the fact that a is a quadratic residue. It communicates a square root of a. We intend to measure the additional knowledge that a prover gives to a verifier during a proof, and investigate whether this additional knowledge may be essentially 0.

To be able to contain the amount of knowledge released during a proof we need to consider a natural generalization of efficient theorem-proving procedures.

2.1 Interactive Proof Systems

Much effort has been previously devoted to make precise the notion of an efficient theorem-proving procedure. *NP* constitutes a very successful formalization of this notion. Loosely speaking, a theorem is in provable in NP if its proof is easy to verify once it has been found. Let us recall Cook's [C] (and independently Levin's [L]) influential definition of *NP* in this light.

The *NP* proof-system consists of two communicating Turing machines A and B: respectively, the *prover* and the *verifier*. The prover is exponential-time, the verifier is polynomial-time. Both A and B are deterministic, read a common input and interact in a very elementary way. On input a string x, belonging to an *NP* language L, A computes a string y (whose length is bounded by a polynomial in the length of x) and writes y on a special tape that B can read. B then checks that $f_L(y)=x$ (where f_L is a polynomial-time computable function relative to the language L) and, if so, halts and accepts. This process is illustrated in figure 1.

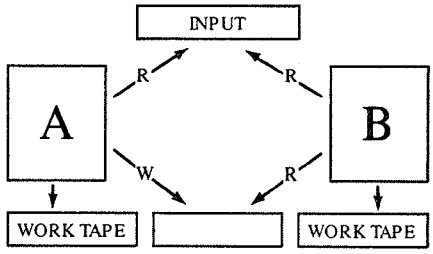

Fig. 1: The *NP* proof-system(*)

The notion of a proof, like the notion of a computation, is an intuitive one. Intuition, however, may and must be formalized. Computability by (deterministic) Turing machines is an elegant example of formalization of the intuitive concept of a computation. Each formalization, however, cannot entirely capture our original and intuitive notions,

(*) (By $----\!>$ we denote a read/write head, by $---R\!->$ a read-only head and by $---W\!->$ a write-only head)

exactly because they are intuitive. Following our intuition, probabilistic algorithms [R] [SS] [GK] are means of computing, though they are not in the previous formal model. Similarly, *NP* is an elegant formalization of the intuitive notion of a theorem-proving procedure. However, *NP* only captures a particular way of communicating a proof. It deals with those proofs that can be "written down in a book".

We want to introduce interactive proof-systems to capture a more general way of communicating a proof. We deal with those proofs that can be "explained in class". Informally, in a classroom, the lecturer can take full advantage of the possibility of interacting with the "recipients" of the proof. They may ask questions at crucial points of the argument and receive answers. This makes life much easier. Writing down a proof that can be checked by everybody without interaction is a much harder task. In some sense, because one has to answer in advance all possible questions.

Before arguing that our interactive proof systems capture the intuitive notion of a proof we need to ask: what is intuitively required from a theorem-proving procedure? First, that it is possible to "prove" a true theorem. Second, that it is impossible to "prove" a false theorem. Third, that communicating a proof should be efficient in the following sense. It does not matter how long must the prover compute during the proving process, but it is essential that the computation required from the verifier is easy.

The theorem-proving procedures we consider are

1) *efficient*: the "recipient" of a proof must be quickly convinced of its correctness.
2) *probabilistic*: on input a false n-bit long statement, the recipient may erroneously be convinced that it is correct with very small probability, say $\frac{1}{2^n}$. On input a true n-bit long statement, the recipient should rightfully be convinced of its correctness with very high probability, say $1-\frac{1}{2^n}$.
3) *interactive*: to verify the correctness of a statement, the "recipient" of the proof may actively ask questions and receive answers from the "prover".

Rather than giving a formal definition of our interactive proof-systems let us present an example.

Example 1: Let Z_m^* denote the set of integers between 1 and m that are relatively prime with m. An element $a \in Z_m^*$ is a *quadratic residue* mod n if $a = x^2$ mod m for some $x \in Z_m^*$, else it is a *quadratic nonresidue*. Now let $L = \{(m,x) \mid x \in Z_m^*$ is a quadratic nonresidue $\}$. Notice that $L \in$ NP: a prover needs only to compute the factorization of m and send it to the verifier without any further interaction. But looking ahead to zero knowledge proof-systems, we will consider a more interesting interactive proof-system for L. The verifier B begins by choosing $n = |m|$ random members of Z_m^*, $\{r_1, r_2, \ldots, r_n\}$. For each i, $1 \leq i \leq n$, he flips a coin, and if it comes up heads he forms $t_i = r_i^2$ mod m, and if it comes up tails he forms $t_i = x \cdot r_i^2$ mod m. Then B sends $t_1, t_2, \ldots, t_n$ to A. The prover, having unrestricted computing power, finds which of the t_i are quadratic residues, and uses this information to tell B the results of his last n coin tosses. If this information is correct, B accepts.

Why does this work? If $(m,x) \in L$, then A correctly predicts all last n coin tosses of B who will definitely accept. If (m,x) not in L, then the $\{t_i\}$ are just random quadratic residues, and the prover will respond correctly in the last part of the computation with probability $1/2^n$. In fact, for each of the last n coin tosses of B, A has probability exactly 1/2 of guessing it correctly.

We may view our interactive proof-systems as a special service offered by ATT. "Dial 144 for mathematical theorem proving". Some user may want to know whether x belongs to language L and dials 144. Immediatly he is connected with a Super Expert able to solve all problems. The trouble is that the user does not trust him. He may not be as expert as he claims or he may be a cheater ready to tell the user false information. Therefore the user will not take his word about what is true. He trusts however the coin he flips and, if at the end of the conversation he has been convinced, than he believes that $x \in L$. In fact he knows that if this was not the case, the probability of being convinced (taken over his own coin tosses) was exponentially vanishing.

We now address our next question.

2.2 The Knowledge Complexity of a Language

To prove that a formula is satisfiable it suffices to exhibit a satisfying assignment. This proof, however, appears to contain more knowledge than the single bit "satisfiable/non-satisfiable" ! Using knowledge complexity, we give a measure for the amount of additional knowledge that must be transfered during a theorem-proving procedure and show that, in some cases it may essentially be 0.

Let us view the case in which the additional knowledge is 0 in our ATT scenario. ATT charges a dollar for each call to 144. During a call the expert can be asked about only one theorem. Assume that the question is whether $x \in L$, i.e. whether x belongs to the set of true theorems. (Here we will assume that x does belong to L.) It is the purpose of the caller checking the validity of the received proof and, it must be admitted, getting at least two theorems at the price of one. ATT does not know whether the caller is a cheater and wants to insure that he only gets that $x \in L$ is true theorem. Can ATT succeed in this? Yes if membership in L can be shown releasing 0 additional knowledge. Rather than dealing with the general notion, we confine ourselves to discuss a restricted scenario.

The particular prover-recipient pair of example 1 possesses the following interesting property. When $x \in L$, A can show to B that indeed x belongs to L and *nothing else*. What is a way of expressing this? When $x \in L$, what B possesses at the end of the proof. First, knowledge of the fact that indeed x is in L, which was the main goal of the proof anyway. Second the actual *text* of the proof, that is an element e randomly selected from AB_x. However, in this example; once B has been convinced that $x \in L$, e becomes totally useless to it. In fact B can generate such e's alone and with the right probability distribution; he can efficiently sample AB_x without A's help. B may perfectly imitate A by looking at his own coin tosses. When B sends t_i computed by squaring r_i, it will imitate A by answering "quadratic residue". When B sends t_i computed by squaring r_i and then multiplying it by x, it will imitate A by answering "quadratic nonresidue". In other words, once B sees that A is able to predict its secret coin tosses, which "prooves" that x is a non-square mod n, it can predict its own coin tosses by himself as they are not secret to him!

Notice however, that, in example 1, A is not proving quadratic non-residuosity without releasing 0 additional knowledge. Rather, A is releasing 0 additional knowledge to the specific B of example 1. In fact, some other B' that interacts with A may decide to create the t_i's in a different way. For instance, such a B may send the sequence of integers $t_i = i$ and therefore receive an answer about their quadratic residuosity that it may not be able to compute by itself if deciding quadratic residuosity is not in probabilistic polynomial time.

However, a (more complex!) interactive proof system for proving quadratic non-residuosity that releases 0 knowledge (to anybody!) can be found in [GMR]. Recently we found a similar proof system for proving quadratic

residuosity as well. This is surprising as no efficient algorithm for deciding quadratic residuosity mod m is known when m's factorization is not given. Moreover, all known NP proofs for this problem exhibit the prime factorization of m. This indicates that adding interaction to the proving process, may decrease the amount of knowledge that must be communicated in order to prove a theorem.

Zero-knowledge proof systems are a surprise and we believe that they cannot prove membership in every language. We actually intend to classify languages according to the amount of additional knowledge that must be released for proving membership in them.

We believe that knowledge complexity is one of the fundamental parameters of a language or, equivalently, of a theorem-proving procedure. Theorem-proving procedures are in fact intended to communicate knowledge and it is very natural to classify them according to the amount of knowledge they must communicate.

3. Applications of Knowledge Complexity to Cryptographic Protocols

In traditional computational complexity or communication complexity, the goal is to communicate as much knowledge as possible as efficiently as possible. Since all participants are considered good friends, no one cares if more knowledge than necessary is communicated. The situation with respect to cryptographic protocols is very different. In this case there is generally no problem at all communicating the knowledge efficiently, but the whole problem is making sure not *too much* knowledge has been communicated.

Model theoretic knowledge has been used to analyze protocols. For example, in [HR] it has been used to prove Rabin's "Oblivious Transfer" correct in some setting. However, as pointed out in [FMR], Rabin's oblivious transfer still lacks a proof of correctness in a complexity theoretic framework.

We believe that knowledge complexity provides the right framework to discuss the correctness of crytographic protocols. For example, applying these ideas [FMR] modified Rabin's oblivious transfer so that it can be proved correct.

Knowledge complexity helps in proving or disproving the correctness of cryptographic protocols as these are based on the secrecy of some private information and should preserve this secrecy. The privacy of some information is what gives us an advantage over our adversaries. Let A(lice) possess the prime factorization of an integer n (say $n=p_1 \cdot p_2$), while B(ob) only knows n. During a protocol with B, A must protect the privacy of her information. Assume that A can perform each step of the protocol without having even to look at the value of p_1 and p_2. Then it is easy to show that the protocol did not compromise the privacy of n's factorization. It is also easy to see, however, that the protocol could not have accomplished any interesting task. In fact A has not made use of her "advantage"! The protocol may accomplish a non-trivial task if, in at least one step of it, A performs a computation c that depends on p_1 and p_2. This raises the question:

Will $c(p_1,p_2)$ betray to much information about p_1 and p_2?

Classical information theory does not provide an answer to this question. Knowledge complexity can. In particular,

1) We can quantify the amount of knowledge about p_1 and p_2 that c conveys and

2) We can design protocols so to minimize this amount of knowledge.

We use this to give an upper bound on the number of times a single protocol or a combination of protocols can be played, using a common secret key, without giving away too much information about the secret key. In addition, trying to measure the amount of knowledge revealed during the execution of a protocol about the secret, may pin point weaknesses in the design of the protocol.

A most important application of these ideas is that it allows us to prove correctness of protocols in a *modular* way. Complex protocols are usually composed of sub-protocols. For instance, many protocols use a sub-protocol for "coin tossing over a telephone" (Blum [Bl1]). However, it is not clear how to use a "normal" definition of correctness of "coin tossing" to prove the correctness of the main protocol. In general, it appears that much stronger definitions for these sub-protocols are needed in order to fit them modularly and cleanly inside larger protocols. Full details will be given in the forthcoming paper on the applications of Knowledge Complexity to the theory of cryptographical protocol.

Acknowledegement

Many thanks to Oded Goldreich for his many helpful suggestions.

References

[Bl1] M. Blum, *Coin flipping by telephone*, IEEE COMPCON 1982.

[C] S.Cook, *The Complexity of Theorem-Proving Procedures", Proc. of 3rd STOC, 1971.*

[FMR]M. Fischer, S. Micali and C. Rackoff, *A Secure Protocol for the Oblivious Transfer*, Lecture Eurocrypt 1984.

[GM]S. Goldwasser, and S. Micali, *Probabilistic Encryption*, JCSS Vol. 28, No. 2, April 1984.

[GM]S. Goldwasser, and S. MIcali ,*Proofs with Untrusted Oracles*, Unpublished Manuscript 1983.

[GGM]O. Goldreich, S. Goldwasser, and S. Micali, *How to Construct Random Function*, 25th FOCS, 1984.

[GMR]S. Goldwasser, S. Micali and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems* 17th Annual ACM Symp. on Theory of Computing, pp 291-304. Better version to appear on the Journal of ACM.

[GK] S. Goldwasser and J. Killian, *A provably Correct and Provably Fast Primality Test*, to appear 18th STOC

[HR] J. Halpern and M.O. Rabin, *A Logic to reason about likehood*, Proc. of 15th STOC, 1983.

[L] L.A.Levin, *Universal Sequential Search Problems*, Probl. Inform. Transm. 9/3 (1973), pp. 265-266.

[R] M. Rabin,

[SS] R. Solovay and V. Strassen *A fast Monte-Carlo test for primality*, SIAM J. on Comp., 1977, pp. 84-85.