T-79.511 Special Course on Cryptology / Zero Knowledge: Four Rounds Are Optimal for Zero Knowledge

Johan Wallén
johan.wallen@hut.fi

October 10, 2001

1 Introduction

Zero knowledge proofs [4] are interactive proofs with the remarkable property that the verifier learns nothing from the proof than the validity of the statement. One of the fundamental results on zero knowledge is the Zero knowledge theorem [2], which roughly asserts that assuming the existence of one-way functions, everything provable is provable in computational zero knowledge, that is, **CZK** = **IP** (= **PSPACE**). This results give rise to the natural question whether one-way functions also are *necessary* for zero knowledge. This question is fundamental due to two reasons. First, it improves our understanding of the relationship between zero knowledge and other fundamental cryptographic primitives. Second, it shows that zero knowledge proofs built on arbitrary one-way functions are *optimal* in the sense that they use the weakest possible assumptions.

In addition to their theoretical importance, zero knowledge proofs are fundamental building blocks for larger cryptographic protocols. In these applications, *efficient* zero knowledge proofs are important. Here, efficiency means the *round complexity* of the proofs: we prefer protocols with a *constant* number of rounds. We are especially interested in *round optimal* protocols that achieve the minimum number of rounds.

In this survey, we discuss these two optimality aspects. The main results discussed are the following.

- **One-way functions are necessary.** Without one-way functions, only trivial zero knowledge proofs are possible. If one-way functions does not exist, the set of languages having computational zero knowledge proofs collapses to **CZK** = **BPP**.
- **Four are rounds necessary.** Any three round computational zero knowledge proof or argument is trivial. This is shown by transforming any three round proof to an **BPP** algorithm deciding the same language.

Put another way, computational zero knowledge proofs for languages outside **BPP** must have at least four rounds and use one-way functions. In this survey, we focus on the second optimality aspect.

Given these results, the natural question is if there are four round **CZK** proofs built from any one-way function for languages outside **BPP**. A partial answer to this question is provided by an four round **CZK** argument for an **NP**-complete language built from any one-way function (of course, this assumes that **NP** $\not\subseteq$ **BPP**).

Throughout this survey, we assume some familiarity with interactive and Arthur-Merlin proofs, zero knowledge, and computational complexity. We refer to the other surveys in this seminar and standard textbooks like [6] for this background.

The rest of this survey is organized as follows. In the next section, we give definitions for some of the most fundamental concepts used in the survey. In Section 3, we show that at least 4 rounds are needed for non-trivial zero knowledge. In Section 4, we discuss the necessity of one-way functions. In Section 5, we discuss a round-optimal computational zero knowledge argument for an **NP**-complete language built from any one-way function. This is followed by some conclusions.

2 Definitions

2.1 One-way functions and indistinguishability

A function is *negligible*, if it is asymptotically smaller than the reciprocal of any polynomial.

Definition 1. A non-negative function $f\colon \mathbb{N} \to \mathbb{R}$ is said to be *negligible*, if for all $d \in \mathbb{N}$, there is an $N_d \in \mathbb{N}$ such that $f(n) < n^{-d}$, $\forall n > N_d$. It is said to be *non-negligible*, if for all $d \in \mathbb{N}$, there is an N_d such that $f(n) > n^{-d}$, $\forall n > N_d$.

Note that any negligible function multiplied by any fixed polynomial is a negligible function. We warn the reader that negligible and non-negligible are *not* complements.

A *one-way function* is a function that is easy to compute, but hard to invert. The standard formalization of this concept is the following.

Definition 2. A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is said to be *one-way* if the following two conditions hold.

- There is an efficient (deterministic) algorithm that given x outputs $f(x)$.
- Given $y = f(x)$, where x is selected at random, it is infeasible to find, with non-negligible probability, a pre-image of y. That is, there is no polynomial time algorithm that succeeds to invert f with non-negligible probability.

Two probability ensembles, $X = \{X_i\}_{i\in I}$ and $Y = \{Y_i\}_{i\in I}$ can be *similar* in three ways: They can be identical $X = Y$, statistically indistinguishable $X \overset{s}{\sim} Y$, or computationally indistinguishable $X \overset{c}{\sim} Y$. We will be mostly concerned with computational indistinguishability, so we only give a definition for that case.

Definition 3. Let $X = \{X_n\}_{n\in\mathbb{N}}$, and $Y = \{Y_n\}_{n\in\mathbb{N}}$ be probability ensembles, where each X_i, Y_i range over strings of length n. The ensembles are said to be *computationally indistinguishable*, written as $X \overset{c}{\sim} Y$, if for every probabilistic polynomial time algorithm A,

$$d_A(n) = |\Pr(A(X_n) = \top) - \Pr(A(Y_n) = \top)|$$

is a negligible function of n.

2.2 Interactive proofs

Recall that an interactive proof (V, P) for a language L is a two-party game between a probabilistic polynomial time verifier V and an unrestricted prover P. For the shared input x, V and P can exchange at most polynomially many messages of polynomial length. The protocol is ended by V either accepting or rejecting x. The following two conditions are required to hold for some negligible function ϵ.

Completeness. $\Pr(V \text{ accepts } x) > 1 - \epsilon(|x|)$ for all sufficiently large $x \in L$.

Soundness. For every prover P' and all sufficiently large $x \notin L$, $\Pr(V \text{ accepts } x) < \epsilon(|x|)$.

The number of rounds is the total number of messages exchanged. The *transcript* $\langle V, R\rangle(x)$ consists of x an all exchanged messages. For probabilistic algorithms, we usually consider equivalent deterministic versions that are given random strings as additional inputs.

We can also consider a weaker version of interactive proofs, where also the prover is restricted to probabilistic polynomial time. These protocols are known as interactive *arguments*. In interactive arguments, the prover usually posses some additional information, like an **NP** witness.

In this survey, we will only consider *computational black-box zero knowledge*, as defined in [3].

Definition 4. An interactive proof (or argument) (V, P) is a *black-box computational zero knowledge proof (argument)* for $L \subset \Sigma^*$ if there is a generic oracle calling algorithm S (the *simulator*) running in expected polynomial time, such that for all verifiers V', the probability ensembles $\{\langle V', P\rangle(x)\}_{x\in L}$ and $\{S^{V'}(x)\}_{x\in L}$ are computationally indistinguishable.

If we in the definition above require that the probability ensembles are identical or statistically indistinguishable, we get *perfect* or *statistical* zero knowledge proofs (arguments), respectively.

In the sequel, zero knowledge will always refer to black-box computational zero knowledge.

3 Four rounds are necessary

The main result of this section is that four rounds are needed for any non-trivial zero knowledge proof.

Theorem 5 ([3]). A language L has a three round black-box computational zero knowledge proof if and only if $L \in \mathbf{BPP}$.

Proof. Let (P, V) be a three round black-box computational zero knowledge proof system for L and let S be the corresponding simulator. We will present a **BPP** algorithm deciding L. The algorithm will use S to distinguish between $x \in L$ and $x \notin L$.

Let x denote the input to the protocol, and let $n = |x|$. The first message α is sent from P to V. To this message, V responds with β. Finally, P sends a message γ to V, and the protocol is ended by V either accepting or rejecting. Note that β is a function of x, α, and the random string of V. Thus, there is a function β_V such that $\beta = \beta_V(x, r, \alpha)$. Let ρ_V denote the predicate computed by V in order to accept or reject. For convenience, we assume that ρ_V is a deterministic function $\rho_V(x, \alpha, \beta, \gamma)$.

Let V^* be any verifier. For simplicity, we write $S = S^{V^*}$. The simulation process consists of several tries. In each try, the simulator feeds V^* with the arguments y (the input) and α (the prover's message). Let $\beta = V^*(y, \alpha)$ be the response of the verifier in this try. Without loss of generality, we can assume the following.

- The simulator always outputs a conversation of the form $(x, \alpha, \beta, \gamma)$, where x and α are fed to the verifier in some try and $\beta = V^*(x, \alpha)$ is the response of V^* in this try.

- The simulator runs in polynomial (worst-case) time. Let $t(n)$ be a polynomial bound on the number of tries by S for inputs of length n. The proof can be generalized to polynomial average case time by increasing the running time and error probability. The resulting algorithm is still in **BPP**.

A crucial observation is that the simulation process is completely determined by x, the random string of S, and the responses of the verifier.

The **BPP** algorithm A operates as follows. First, A will pick a random string R_S for S, and $t = t(n)$ random strings $r^1, \ldots, r^t$. The simulator S is used as a subroutine with x as input and R_S as random string. Whenever the simulator invokes the verifier with arguments y and α, we find the first unused r^i and respond with $\beta^i = \beta_V(x, r^i, \alpha)$. If S invokes the verifier with the same α more than once, we return the same β^i as the previous time regardless of the value of y.

Note that the conversation is uniquely specified by x, R_S, and $r^1, \ldots, r^t$. Thus, we can define the function $\mathrm{conv}_S(x, R_S, r^1, \ldots, r^t)$ to be the resulting transcript. Let α^i the ith different α output by S during the simulation. Clearly, α^i is a function of x, R_S, and $r^1, \ldots, r^{i-1}$: $\alpha^i = \alpha_S(x, R_S, r^1, \ldots, r^{i-1})$. By our assumption above, if $\mathrm{conv}_S(x, R_S, r^1, \ldots, r^t) = (x, \alpha, \beta, \gamma)$, S explicitly tried the arguments x and α in one of the tries. Thus, there is an i, $1 \leq i \leq t$, such that $\beta = \beta_V(x, r^i, \alpha)$.

Definition 6. A vector $(x, R_S, r^1, \ldots, r^t)$ is *S-good*, if $\mathrm{conv}_S(x, R_S, r^1, \ldots, r^t) = (x, \alpha, \beta, \gamma)$ and $\rho_V(x, \alpha, \beta, \gamma) = \top$. We say the conversation is *(S, i)-good* (or simply i-good), when i is minimal for which $\beta = \beta_V(x, r^i, \alpha)$.

Finally, the algorithm A accepts x if and only if $(x, R_S, r^1, \ldots, r^t)$ is S-good.

Since both S and V runs in polynomial time, so does A. By the following two lemmas, the algorithm can distinguish between $x \in L$ and $x \notin L$ with small enough error probability. Thus, $L \in$ **BPP**. □

Lemma 7. For $x \notin L$, only a negligible portion of the vectors $(x, R_S, r^1, \ldots, r^t)$ are S-good.

Proof. Assume that the portion of S-good vectors $(x, R_S, r^1, \ldots, r^t)$ for $x \notin L$ is not negligible. It follows that there must be infinitely many $x \notin L$ such that the portion of S-good vectors is non-negligible. For each such x, there is an index i_0 such that a non-negligible portion of the vectors are i_0-good. Thus, there is a non-negligible number of prefixes $(x, R_S, r^1, \ldots, r^{i_0-1})$, each with a non-negligible number of i_0-good continuations $(r^{i_0}, \ldots, r^t)$. Fix one such prefix and let $\alpha^{i_0} = \alpha_S(x, R_S, r^1, \ldots, r^{i_0-1})$. For each i_0-good continuation, S outputs $(x, \alpha^{i_0}, \beta^{i_0}, \gamma)$ and $\rho_V(x, \alpha^{i_0}, \beta^{i_0}, \gamma) = \top$. Thus, there must be a non-negligible number of β^{i_0} for which this happens.

We conclude that there for each x as above is an $\alpha_x = \alpha^{i_0}$ such that the set $B(x, \alpha_x) = \{\beta \mid \exists\gamma\colon \rho_V(x, \alpha_x, \beta, \gamma) = \top\}$ is of non-negligible size. Consider a prover that sends α_x as its first message. If V responds with $\beta \in B(x, \alpha_x)$, the prover responds with one of the γ that causes V to accept. This gives a non-negligible acceptance probability for infinitely many $x \notin L$, contradicting the soundness property of (V, P). $\square$

Lemma 8. For $x \in L$, a non-negligible portion of the vectors $(x, R_S, r^1, \ldots, r^t)$ are S-good.

Proof. Let $x \in L$ and $|x| = n$. Without loss of generality, we can assume that there is an ℓ such that $|\alpha| = |\beta| = \ell = \ell(n)$. Let H_n be a set of $t(n)$-independent hash functions $\mathbb{Z}_2^\ell \to \mathbb{Z}_2^\ell$. For a randomly selected $h \in H_n$, the images of h are uniformly distributed, and t-wise independent.

For each $h \in H_n$, we define a verifier V_h^*, which responds to the prover's message α with $\beta = \beta_V(x, h(\alpha), \alpha)$. That is, V_h^* simulates V with $h(\alpha)$ as random string.

Consider the simulator S using the verifier V_h^*. For fixed x, R_S and $h \in H_n$, the simulation is deterministic. In particular, this determines a unique sequence of α tried by the simulator and responses β of V_h^*. Let $\alpha^1, \ldots, \alpha^s$ denote the different values of α used in the tries. Add $t - s$ strings α such that the strings $\alpha^1, \ldots, \alpha^s, \ldots, \alpha^t$ are all different. Let $r^i = h(\alpha^i)$ and define $v(x, R_S, h) = (x, R_S, r^1, \ldots, r^t)$.

Claim 9. For uniformly selected $h \in H_n$, there is a non-negligible portion of pairs (R_S, h) for which $v(x, R_s, h)$ is S-good.

To prove this claim, let p_x denote the probability that (V, P) accepts x. Since $x \in L$, p_x is non-negligible. Consider (V_h^*, P) for uniformly chosen $h \in H_n$. For all α, the random string $h(\alpha)$ used in the simulation of V is uniformly distributed. Thus, the responses of V_h^* on α are distributed in the same way as the responses of V on α. It follows that the probability that (V_h^*, P) accepts x is p_x. Since the simulator succeeds in simulating (V_h^*, P), the probability that S outputs an accepting conversation when interacting with V_h^* is almost the same (up to negligible difference) as the probability that (V_h^*, P) outputs an accepting conversation. The latter probability is p_x. We conclude that that probability that $v(x, R_S, h)$ is S-good is almost the same as p_x, and thus non-negligible.

Claim 10. For all x and R_S, and for uniformly chosen $h \in H_n$, the vector $v(x, R_S, h)$ is uniformly distributed over $\{(x, R_S, r^1, \ldots, r^t) \mid r^i \in \mathbb{Z}_2^\ell\}$.

To prove the claim, note that $v(x, R_S, h) = (x, R_s, r^1, \ldots, r^t)$ if and only if for all i, $1 \leq i \leq t$, $r^i = h(\alpha_S(x, R_S, r^1, \ldots, r^{i-1}))$, where α_S is as above. By the properties of H_n, for any t different $a_1, \ldots, a_t \in \mathbb{Z}_2^\ell$, the sequence $h(a_1), \ldots, h(a_t)$ is uniformly distributed over all the possible sequences $b_1, \ldots, b_t$ over $\mathbb{Z}_2^\ell$. Thus, for all fixed sequences $r^1, \ldots, r^t$, the probability that $v(x, R_S, h) = (x, R_S, r^1, \ldots, r^t)$ equals the probability that h maps $\alpha^i = \alpha_S(x, R_S, r^1, \ldots, r^{i-1}) \mapsto r^i$ for all i. Since the α^i are different, the properties of H_n gives that this latter probability is the same for every sequence $r^1, \ldots, r^t$. The claim follows.

For any R_S, Claim 10 shows that $v(x, R_S, h)$ is uniformly distributed over all possible vectors $(x, R_S, r^1, \ldots, r^t)$. By Claim 9, a non-negligible portion of the vectors $v(x, R_S, h)$ are S-good. It follows that a non-negligible portion of the vectors $(x, R_S, r^1, \ldots, r^t)$ are S-good. $\square$

Note that the proof of Theorem 5 holds also when all provers are restricted to probabilistic polynomial time. Thus, we have the following corollary.

Corollary 11. A language L has a three round black-box computational zero knowledge argument if and only if $L \in \mathbf{BPP}$.

4 One-way functions are necessary

In this section, we argue that one-way functions are essential for non-trivial zero knowledge. Before we state the precise result, we define a few concepts.

An *auxiliary input one-way function* f_x, where x is a bit string, is roughly speaking an ordinary one-way function for almost all values of x, when all algorithms are given x as an additional input.

Let A, B be probabilistic polynomial time algorithms, and let $A(x) = (y, z)$. B is said to *extrapolate* A, if for all x, $A(x) \overset{s}{\sim} (y, B(y, z))$. If (auxiliary input) one-way functions does *not* exist, a *universal extrapolation* result holds: for all probabilistic polynomial time algorithms A, there is a probabilistic polynomial time algorithm B that extrapolates A (see the references in [5]).

Theorem 12 ([5]). Under the assumption that auxiliary input one-way functions does not exists, if there is a computational zero knowledge proof system for L, then $L \in \mathbf{BPP}$.

Proof sketch. Let (V, P) be a computational zero-knowledge proof system for L with simulator S. We will describe a **BPP** algorithm A deciding L. On input x, the algorithm operates as follows. First, it picks a random string R_V for V. Then, it simulates moves of a prover P^* and the original verifier V with random string R_V. If it is V's turn, A simply simulates V to produce the next message. If it is P^*'s turn, A takes the transcript so far and extrapolates the behavior of S on this partial transcript to predict P's next move. This move is used for P^*. A accepts if and only if V accepts during this simulation.

Under the assumption that there are no one-way functions, both the simulation of the verifier and the extrapolation of P runs in probabilistic polynomial time. Since the prover P^* does not use the random tape of V, the soundness property of (V, P) guarantees that the error probability for $x \notin L$ is small enough. It remains to bound the error probability for $x \in L$.

The idea behind the proof is to show that for $x \in L$, the transcript produced by A is computationally indistinguishable from the transcript produced by (V, P). That is, $(V, P^*) \overset{c}{\sim} (V, P)$. This shows that the error probability of A is small enough also for $x \in L$, since otherwise (V, P^*) could be easily distinguished from (V, P). The proof that $(V, P^*) \overset{c}{\sim} (V, P)$ is long and technical, and can be found in [5].

If follows that $L \in \mathbf{BPP}$. □

5 An optimal protocol for NP

Given the results discussed so far, it is natural to ask if there are four round computational zero knowledge proofs or arguments built from arbitrary one-way functions for languages outside **BPP**. We give a partial answer by presenting a four round computational zero knowledge *argument* for the **NP**-complete (propositional) SAT. The protocol we discuss is from [1], and can be based on any one-way function. We warn the reader that we omit several important details of the protocol; please see [1] for the full protocol.

Throughout this section, n is a security parameter, and $f = f_n$ denotes a fixed but arbitrary one-way function. The input to the protocol is a formula φ, and the prover is supposed to know a satisfying truth assignment T.

One of the consequences of the **NP**-completeness of SAT is the following. There is a polynomial-time computable function $\mathsf{formula}_f$ such that for any $y \in \{0,1\}^*$, $\mathsf{formula}_f(y)$ is a boolean formula that is satisfiable if and only if there is an $x \in \{0,1\}^*$ such that $f(x) = y$. Furthermore, there are polynomial time functions $t_{f,1}$, $t_{f,2}$ such that the following holds. $T = t_{f,1}(x)$ is a satisfying truth assignment of $\mathsf{formula}_f(f(x))$. Conversely, if T is a satisfying truth assignment of $\mathsf{formula}_f(y)$, $x = t_{f,2}(T)$ satisfies $f(x) = y$. The mappings $t_{f,i}$ are called *witness transformations*. The important point is that the knowledge of a satisfying truth assignment of $\mathsf{formula}_f(y)$ is equivalent to knowing a pre-image x of y under f.

The protocol uses a *commitment scheme* based on f and an atomic four round zero knowledge proof with error probability 0.5 as building blocks.

5.1 The atomic protocol

The atomic protocol has the following steps (for input ϕ and truth assignment T).

1. V picks a random commitment setup string R and sends it to P.
2. P picks a random string ρ and computes an *encapsulated circuit*, $C = \mathsf{encCirc}_f(\phi, T, R, \rho)$. The formula is randomized using ρ, and annotated with the values of the truth assignment T. This gives the annotated circuit d. C is a commitment to d using R as commitment setup string. C is sent to V.
3. V picks a random challenge bit c and sends it to P.
4. P responds to this challenge with an answer $D = \mathsf{answer}_f(\phi, T, R, \rho, c)$. D consists of opening parts of the commitment C.
5. V accepts if and only if $\mathsf{check}_f(\phi, R, C, c, D) = \top$.

Clearly, the operations used in the protocol must satisfy certain properties so that this is a zero knowledge proof with error probability 0.5. The operations must also satisfy some additional properties for use in the full protocol. (Omitted.)

5.2 Putting it all together

The idea behind the full protocol is to run n copies of the atomic protocol in parallel. Naturally, this requires some extra care, since normal zero knowledge does *not* (in general) compose in parallel [3]. The structure of the full protocol is as follows.

1. For $i = 1, \ldots, n$ and $j = 0, 1$, V picks $x_{i,j} \in \{0,1\}^n$ randomly and sets $y_{i,j} = f(x_{i,j})$. Let $Y = (y_{i,j})$. V also picks n random commitment setup strings $R = (R_1, \ldots, R_n)$. The string R_i will be used in the ith parallel run of the atomic protocol. Y and R are sent to P. For $i = 1, \ldots, n$ and $j = 0, 1$, let $\phi_{i,j} = \mathsf{formula}_f(y_{i,j})$. These can be computed by both V and P.

2. P picks n random bits $b = (b_1, \ldots, b_n)$. The bit b_i selects the value y_{i,b_i}. In the next step, V will be asked to reveal the pre-images of these values. Set $\phi = \bigvee_{i=1}^{n} \phi_{i,1-b_i}$. This is the conjunction of the formulas corresponding to Y-values not selected by P. As f is one-way, the prover does not know a truth assignment for ϕ. The prover now sets $\psi = \phi \vee \varphi$ (φ is the input to the protocol). P now picks n random values $\rho_1, \ldots, \rho_n$ and computes the commitments $C_i = \mathsf{encCirc}(\psi, T, R_i, \rho_i)$ for $i = 1, \ldots, n$. P sends b and $C = (C_1, \ldots, C_n)$ to V.

3. V computes $x = (x_{1,b_1}, \ldots, x_{n,b_n})$ (the pre-images of the Y-values selected by P). Since V knows $b_1, \ldots, b_n$, he can compute ϕ and ψ. V picks n bits $c = (c_1, \ldots, c_n)$ as challenges, and sends (x, c) to P.

4. The prover checks that $f(x_i) = y_{i,b_i}$, $i = 1, \ldots, n$. If the check fails, the protocol is aborted. P computes the answers to the challenges, $D_i = \mathsf{answer}_f(\psi, T, R_i, \rho_i, c_i)$, and sends $D = (D_1, \ldots, D_n)$ to V.

5. V checks that $\mathsf{check}_f(\psi, R_i, C_i, c_i, D_i) = \top$ for $i = 1, \ldots, n$. V accepts $\iff$ all checks succeeds.

This protocol can be shown to be a computational zero knowledge argument with negligible error probability (see [1]).

6 Conclusions

In this survey, we have discussed two important optimality aspects of computational zero knowledge proofs, namely the necessity of one-way functions, and that at least four rounds are needed for non-trivial zero knowledge proofs and arguments. We also sketched a non-trivial zero knowledge *argument* achieving these lower bounds.

Since one-way functions are a necessity for most of modern cryptography, their necessity for non-trivial zero knowledge should come as no surprise. The optimal round complexity deserves some further comments.

In [3], it is also shown that $\mathbf{AM}[k] = \mathbf{BPP}$ for all constants $k \geq 3$. Although interactive and Arthur-Merlin proofs are equivalent for language recognition, this does no longer hold for zero knowledge! Thus, "private coins" are essential for non-trivial zero knowledge. We conclude that one-way functions, at least four rounds, and the verifier's randomness are all very essential for zero knowledge.

References

[1] M. Bellare, M. Jakobsson, and M. Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *Eurocrypt '97*, volume 1233 of *LNCS*, pages 280–305. Springer-Verlag, 1997.

[2] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero knowledge. In *Crypto '88*, volume 403 of *LNCS*, pages 37–56. Springer-Verlag, 1988.

[3] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal of Computing*, 25(1):169–192, 1996.

[4] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18:186–208, 1989.

[5] R. Ostrovsky and A. Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *ISTCS '93*, June 1993.

[6] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.