Scientia Iranica, Vol. 6, No. 1, pp 39–42
© Sharif University of Technology, January 1999

SCIENTIA
IRANICA

Research Note

A Zero-Knowledge Proof for Vertex Cover Problems

J. Mohajeri[1]

In this paper, a new zero-knowledge proof for vertex cover problem is proposed which is efficient and practical. This proof can be modified for independent set and clique problems. These problems are all NP-C.

INTRODUCTION

Many researchers have studied zero-knowledge proofs and the class of problems which have such zero-knowledge proofs. Little attention, however, has been paid to the practicality of these proofs. It is known, for example, that under certain cryptographic assumptions, all problems in NP have zero-knowledge proofs [1,2]. Although these proofs can be performed by probabilistic polynomial time provers who have the appropriate trapdoor information, they may involve a transformation to a circuit or to an NP-complete problem, therefore, they are often quite inefficient. The first zero-knowledge proofs, for quadratic residuosity and nonresiduosity [3], were practical as well as efficient; however, probabilistic polynomial-time proving could be performed if appropriate trapdoor knowledge was available. One of these kinds of proofs is demonstrated in later sections.

In this paper, a practical zero-knowledge proof for a vertex cover problem is presented in the following order.

First, polynomial time algorithms and intractable problems are described and NP-complete problems are defined. Then, minimum and maximum disclosure proofs are explained. Finally, the main idea for zero-knowledge proofs and the new zero-knowledge proof for vertex cover problems are presented.

POLYNOMIAL TIME ALGORITHMS AND INTRACTABLE PROBLEMS

Different algorithms possess a wide variety of time complexity functions. The characterization of efficiency defined as "efficient enough" and "too inefficient" always depends on the actual situation. However, computer scientists recognize a simple distinction that offers considerable insight into this matters, which is the difference between polynomial and exponential time algorithms.

Definition 1

A function $f(n)$ is $O(g(n))$ whenever there exist a constant c and an integer N such that $\mid f(n) \mid \leq c.\mid g(n) \mid \forall n \geq N$.

Definition 2

A polynomial time algorithm is defined as one whose time complexity function is $O(p(n))$ for some polynomial function p, where n is used to denote the input length.

Based on the above definitions, any algorithm whose time complexity function cannot be bounded as defined above is called an exponential time algorithm [4].

The distinction between these two types of algorithm has particular significance when the solution of large problem instances is considered.

The fundamental nature of this distinction was first discussed by Cobham [5] and Edmonds [6]. Edmonds, in particular, equated polynomial time algorithms with "good" algorithms and conjectured that certain integer programing problems might not be solvable by such "good" algorithms. This reflects the viewpoint that exponential time algorithms should not be considered as "good" algorithms, which, indeed, is usually the case. Hence, a problem is referred to as intractable if it is so hard that no polynomial time algorithm can possibly solve it.

1. *Electronic Research Center, Sharif University of Technology, P.O. Box 11365-8639, Tehran, I.R. Iran.*

NP-COMPLETE PROBLEMS

Problems are classified into two important classes, P and NP.

Definition 3

The notation P is used for all problems that can be solved using a polynomial time algorithm on a deterministic Turing machine, whereas NP consists of all problems that a polynomial time algorithm on a nondeterministic Turing machine can solve.

From the above definition, it is clear that class P is a subclass of NP, but it is a celebrated open problem where P $\neq$ NP.

Another particular subset of NP is NP-C. A specific problem is NP-complete if and only if it is one of the "hardest" members of NP, that is , every problem in NP can be reduced in polynomial time to this specific problem. It follows that P = NP if and only if an NP-complete problem is in P. One of the most famous members of NP-C is satisfiability problem.

Since it is generally believed that P $\neq$ NP, NP-complete problems are considered to be intractable. The foundations for the theory of NP-completeness were presented in [7].

MINIMUM AND MAXIMUM DISCLOSURE PROOFS

Assume that P (the prover) knows some information, and wants to convince V (verifier) of it. P can convince V in two ways: maximum disclosure proof and minimum disclosure proof.

In a maximum disclosure proof, P simply discloses his information, so that V is capable of verifying and, therefore, actually learns the information.

Example 1

Let P know the prime factorization of n, this could be proved to V as the following:

P: I know prime factorization of n. V: No you don't. P: Yes I do. V: Do not. P: Do too. V: Prove it. P: All right. I will tell you. He whispers in V's ear. V: That is interesting. Now I know it, too. I'm going to tell Bob. P: Oh no.

This is an instance of maximum disclosure proofs. Unfortunately, in this usual way, in order for P to prove something to V, P has to tell him. But then he knows it too and can consequently tell anyone else he wants to and P can do nothing about it.

Using One-way functions, P could perform a minimum disclosure proof. In a minimum disclosure proof, P convinces V that he has the information, but does not reveal even one bit of it and, consequently, does not help V in any way to determine the information he has.

This protocol proves to V that P does have a piece of information, but it does not provide the content of the information. These proofs take the form of interactive protocols. V asks P a series of questions. If P knows the secret, he can answer all the questions correctly. If he does not, he has just some chance of answering correctly. After about 10 questions, V will be convinced that P knows the secret, yet none of the questions or answers reveal the content of the information.

Definition 4

In a minimum disclosure proof, the following properties hold:

1. P probably cannot cheat V. If P does not know the proof, his chances of convincing V that he knows the proof are negligible.
2. V cannot cheat P. He does not get the slightest hint of the proof, apart from the fact that P knows the proof. In particular, V cannot demonstrate the proof to anyone else without proving it himself from scratch. Zero-knowledge proofs have an additional condition.
3. V learns nothing from P that he could not learn by himself without P, apart from the fact that P knows the proof. In other words, V is able to simulate the protocol as if P were participating although he, in fact, is not. A very simple minimum disclosure proof about knowledge of the factor of n is shown in the following example.

Example 2

Recall Example 1; P knows the factorization of n and wants to convince V of it. He can do so through using the following protocol in three steps:

Step 1: V chooses a random integer x and tells that $z = x^4 \pmod{n}$ to P.

Step 2: P has to compute $y = z^{\frac{1}{2}} = x^2 \pmod{n}$ with his information and tell it to V.

Step 3: V himself computes $x^2 \pmod{n}$ and compares it to y, if they are equal he accepts, otherwise rejects.

V obtains no new information because he can square x himself. On the other hand, it is known that extracting square roots is equivalent to factoring n. In Step 2, P has to extract not only a square root of x^4, but the particular one of the four square roots that is a quadratic residue (mode n). Determining quadratic residuosity is also intractable without knowing the factor of n. Of course, the possibility of P succeeding without knowing the factor of n can be made still smaller by iterating the protocol.

Let now repeat the basic requirements. It is assumed that the information is the proof of a theorem.

1. The prover probably cannot cheat the verifier.
2. The verifier cannot cheat the prover.

MAIN IDEA FOR ZERO-KNOWLEDGE PROOFS

The protocol of Example 2 was constructed in an ad hoc manner, based on the special interconnection between factoring and extracting square roots. Some general ideas are needed if one wants to construct protocols satisfying the defined requirements for a large class, such as problems in NP. The crucial idea in construction will be that of a lockable box. The verifier cannot open it because the prover has the key. On the other hand, the prover has to commit himself to the contents of the box, that is, he cannot change the contents when he opens the box. In fact, the verifier may watch when the prover opens the boxes. Locking information in a box means applying a one-way function to it. The prover knows the inverse function and applies it when opening the box. His commitment to the box can be verified by applying the one-way function to the plain text information. In the next section, this method will be employed in the zero-knowledge proof of the vertex cover problem.

ZERO-KNOWLEDGE PROOF FOR VERTEX COVER PROBLEM

Definition 5

Let G(V,E) be a graph, a subset $V' \subseteq V$ is called a vertex cover if and only if for each edge $\{u,v\} \in E$, at least, either u or v belongs to V'.

Vertex Cover Problem

Let k be a positive integer, with $k \leq \mid V \mid$ and G(V,E) a graph; now, is there a vertex cover of size k or less for G?.

Lemma 1

For any graph G(V,E) and subset $V' \subseteq V$, the following statements are equivalent:

a) V' is a vertex cover for G.

b) $V - V'$ is an independent set for G.

c) $V - V'$ is a clique in the complement G^c of G, where $G^c = (V, E^c)$ with $E^c = \{\{u,v\} : u,v \in V, \{u,v\} \notin E\}$.

This implies that the NP-completeness of all three problems will follow as an immediate consequence of proving that any one of them is NP-C. Karp has proved the following theorem in [8].

Theorem 1

Vertex cover is an NP-Complete problem. Every zero-knowledge proof based on the above problem can be modified for the other two. A zero-knowledge for independent set problems is presented in [9]. That protocol can also be used for the two other problems. In this paper, a new zero-knowledge proof protocol based on vertex cover problems will be proposed.

P wants to convince V that he knows a vertex cover of a graph with n vertices. In the following, a protocol, with m round is described, each round consists of 4 steps.

Step 1: P prepares and presents to V the following locked boxes; B_i, $1 \leq i \leq n$, B'_i, $1 \leq i \leq k$, B_{ij}, $1 \leq i < j \leq n$. Each of B_i, B'_i and B_{ij} are respectively called vertex box, cover box and edge box. Each of B_i boxes contains one of the nodes, each of the B'_i boxes contains one of the B_i boxes which has one vertex of V'. Each of B_{ij} boxes consists of 0 or 1. If B_i and B_j include v_{ki} and v_{kj}, respectively, then 1 appears in B_{ij} if and only if $\{v_{ki}, v_{kj}\}$ belongs to E, and 0 appears otherwise.

Step 2: V flips a fair coin and tells P the outcome.

Step 3: a) If the outcome is "heads", P opens B_i and B_{ij} boxes.

b) If the outcome is "tails", V randomly selects one of the B_{ij} boxes, i.e., an edge box, then: (i) if it contains 1 then P has to open one of the B'_i, which contains B_i or B_j, otherwise (ii) if 0 appears, V reselects another edge box. This procedure will be repeated until 1 appears, then (i) will be conducted.

Step 4: a) V verifies that he has got a copy of G. If so, he accepts, otherwise rejects. The examination is easy because the opened vertex boxes reveal the proper vertex labels, so no problem concerning graph isomorphism has to be settled.

b) V verifies that the opened box contains B_i or B_j, if so, he accepts, otherwise rejects. Note that the boxes have to be reconstructed for each round of protocol.

P may try to cheat V in two ways:

1. He does not put a description of G in the boxes, but, rather, a description of some other graphs that he knows their vertex cover. Then he succeeds if event b occurs and gets caught if event a happens.
2. P uses a false vertex cover of size k. Then he succeeds if event a or b occurs and V chooses an edge box which at least one of its vertex belongs to V', otherwise he gets caught.

Probability of success for P in state (1) is $\frac{1}{2}$ and this probability in state (2) is $\frac{1}{2}\frac{m}{e}$, where $e = \mid E \mid$ and m is the number of edges in G covered by V' Thus, probability of a fail prover passing L round is equal to: $\frac{1}{2}^L$ or $(\frac{m}{2e})^L$ or $\frac{1}{2}^r(\frac{m}{2e})^{L-r} = \frac{1}{2}^L(\frac{m}{e})^{L-r}$, and each of them approaches zero as L increases.

The above protocol satisfies three conditions of a zero-knowledge proof, because:

1. The prover probably cannot cheat the verifier. If the prover does not know a vertex cover of the graph, his chance of convincing the verifier that he knows this information is negligible, because the possibility of P succeeding without knowing a vertex cover of the graph can be made arbitrarily small by iterating the protocol.
2. The verifier cannot cheat the prover. He does not get the slightest hint of the proof, apart from the fact that the prover knows a proof. In particular, the verifier cannot prove that he knows a vertex cover of size k for G to anyone else without proving it himself from scratch.
3. Assume that V has an algorithm A (running in random polynomial time) to extract some information from his conversation with P. In the following way, V can use A to extract the same information even in the absence of P. V first plays the role of P. He flips a coin and, according to the outcome, either applies an isomorphism to G and locks the result in the vertex and edge boxes and puts arbitrary k boxes in the cover boxes, or else locks an arbitrary k vertex box in cover boxes and puts some numbers in other boxes such that the chosen vertex performs a vertex cover for G.

Now having received the boxes, V plays the role of V. He applies his algorithm A to decide the choice between (a) and (b) lines. He either gets the same information as in the presence of a true prover P or learns that P is a false prover. V can do everything in polynomial time. Hence, the following result is obtained.

Theorem 2

The given protocol for the vertex cover is zero-knowledge proof.

ACKNOWLEDGEMENT

The author would like to thank Dr. Mahmood Salmasizadeh for his comments concerning this work.

REFERENCES

1. Chaum, D. "Demonstrating that a public predicate can be satisfied without revealing any information about how", *Advances in Cryptology - Crypto 86 Proceedings*, pp 195-199 (1987).
2. Goldreich, O., Micali, S. and Widgerson, A. "How to prove all NP-statements in Zero-Knowledge, and a methodology of Cryptographic protocol design", *Lecture Notes in Computer Science*, **263**, Springer, Berlin Heidelberg, New York, pp 171-185 (1987).
3. Goldwasser, S., Micali, S. and Rackoff, C. "The knowledge complexity of interactive proof systems", *Proceeding of the 17th ACM Symposium on the Theory of Computing*, pp 291-304 (1985).
4. Garey, M.R. and Johnson, D.S., *Computers and Intractability*, W.H. Freeman and Company (1979).
5. Cobham, A. "The intrinsic computational difficulty of functions", *Proc 1964 International Congress for Logic Methodology and Philosophy of Science*, North Holland, Amesterdam (1964).
6. Edmonds, J. "Paths, trees, and flowers", *Canad. J. Math.*, **17**, pp 449-467 (1965).
7. Cook, S.A. "The complexity of theorem-proving procedures", *Proc. 3rd Ann. ACM Symp. On Theory of Computing*, New York, pp 151-158 (1971).
8. Karp, R.M., *Reducibility Among Combinatorial Problems*, R.E. Miller and J.W. Thatcher, Eds., Complexity of Computer Computations, Plenum Press, New York, pp 85-103 (1972).
9. Mohajeri, J., *Zero-Knowledge Proofs for Independent Set and Dominating Set Problems*, C.J. Colbourn and E.S. Mahmoodian, Eds., Combinatorics Advances, Kluwer Academic Publishers, pp 251-254 (1995).