Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 3 Issue 10, October - 2016

Zero Knowledge Proof Protocol Based on Graph Isomorphism Problem

Mariana Durcheva
Department of Applied Mathematics and Informatics
Technical University of Sofia
Sofia, Bulgaria
E-mail: m_durcheva@tu-sofia.bg

***Abstract*—Cryptocurrency (Bitcoin) eliminates the need for a trusted third party to process payments. Zero-knowledge proof (ZKP) systems allow a Prover to convince a Verifier of the validity of a statement while revealing no additional information beyond the statement's validity. ZKP systems have applications as the building blocks in modern cryptography. They are considered to be very well suited to resource-limited systems. In this paper we suggest a ZKP protocol based on Graph Isomorphism Problem which is known to belong to the complexity class nondeterministic polynomial (NP).**

Keywords—ZKP protocol; Graph Isomorphism Problem.

I. INTRODUCTION

European Commission [3] reported that "The 2012 Eurobarometer poll on cyber security found that 38 % of EU internet users have changed their behavior because of these cybersecurity concerns: 18 % are less likely to buy goods online and 15 % are less likely to use online banking. It also shows that 74% of the respondents agreed that the risk of becoming a victim has increased, 12% have already experienced online fraud and 89% avoid disclosing personal information". This produce a need for increasingly greater development of cryptographic science as its development has made possible secure communications and electronic payments over insecure channels like Internet.

The design of protocols is not difficult if a Third Trusted Part (TTP) is available. In this case case, all input information is given by both parties to it, and then the TTP distributes corresponding outputs to each party. For instance, the use of a credit card is a form of electronic cash which relies on a trusted third party, preventing overspending or double spending. But cryptocurrency (Bitcoin) eliminates the need for a trusted third party to process payments and establish a pure peer-to-peer decentralized currency. Every peer in the Bitcoin network keeps the collection of all transactions. Transactions must be made public but Bitcoin addresses can be created without any personal data from the owner. ZKP can be used also in smart cards, contract signing, secret exchange, certified mail, coin flipping and two-sided comparison protocols.

In this paper we present a new protocol that combines zero-knowledge proofs and key exchange methods to provide secure and authenticated communication.

Zero-knowledge proof (ZKP) systems have been introduced by Goldwasser, Micali and Rackoff [5] in 1985. These protocols allow a Prover to convince a Verifier of the validity of a statement while revealing no additional information beyond the statement's validity. Different problems that are not known to be efficiently computable (such as the Graph Isomorphism, and Graph Non-Isomorphism problems) are shown to admit zero-knowledge proof systems [5],[6]. Since they were established, ZKP systems have applications as the building blocks in modern cryptography and in particular, in security protocols requiring authentication. During the authentication procedure, a Prover must respond to challenges issued by a Verifier over a number of accreditation rounds. The Prover must be able to answer all challenges successfully to prove his identity. Verifier's confidence in Prover's identity increases with every single round. In ZKP protocols, the Verifier cannot learn anything from the authentication procedure. Moreover, the Verifier is unable to cheat the Prover because he cannot calculate the Prover's secret. Furthermore, the Verifier cannot cheat the Prover because the protocol is repeated as long as the Verifier is not convinced. A challenge is selected at random, so the Verifier cannot pretend to be the Prover to a third party. For that reason, the computational overhead required for Prover to prove his identity is significantly less than that required for other ways of authentication without the need for a trusted third party such as that of RSA, while still remaining very difficult for an intruder to cheat (due to being based upon NP problems). This is the main reason why ZKP systems are considered to be very well suited to resource-limited systems [1],[2], [16].

II. PRELIMINARIES

In [7] some graph isomorphism problems are proved to be polynomial time equivalent:

- *Given two graphs with n vertices each, decide whether they are isomorphic*;

- *Given two labelled graphs, decide whether they are isomorphic*;
- *Given two graphs, decide whether they are isomorphic, and if so, construct an isomorphism from one to the other;*
- *Given a graph, determine a generating set for the Automorphism Group;*
- *Given a graph, determine the order of the Automorphism Group;*
- *Given two graphs with n vertices each, determine the number of isomorphisms from one to the other.*

The graph isomorphism problem is known to belong to the complexity class nondeterministic polynomial (NP) time but not known to be solvable in polynomial time nor NP-complete for the general case (see [15]). The problems that are polynomial-time equivalent to graph isomorphism are called *graph isomorphism complete* [7]. At present it is not known a polynomial time algorithm for solving the graph isomorphism complete problem in the worst case - all known algorithms have exponential time complexity. Most of the works available in the literature have been focused on finding practical isomorphism testing algorithms that solve the general problem in polynomial time. As a result, algorithms such as the *nauty* package of Brendan McKay [9], [10], *bliss* of Tommi Juntilla and Petteri Kashi [8], *saucy* of Martin Kutz and Pascal Schweitzer [12], *sinauto* and *conauto* of Jose Luis Lopez Presa [14], *Traces* of Aldofo Piperno [13], *nauty and Traces* [11] *Vsep* of Stoicho Stoichev [17] have been developed.

A. Graph Isomorphism and Graph Automorphisms

For the definitions of graph isomorphism and graph automorphism we follow [7]. A simple, undirected graph G is a pair (V, E), where V is a finite set of *graph vertices*, and E is a (finite) set of unordered pairs (v, w), where v and w are distinct vertices. The elements of E are called the *edges* of the graph. For a graph with n vertices the vertex set V is $\{1, \ldots, n\}$. When considering one-to-one maps from the vertex set of a graph G_1 onto the vertex set of a graph G_2, this convention allows us to understand as a permutation. Two vertices v and w of the graph $G = (V, E)$ are *adjacent* if there is an edge (v, w) in E.

Let $G = (V, E)$ be a graph, $V = \{1, \ldots, n\}$ and S_n be the symmetric group of G. A permutation $\pi \in S_n$ is an *automorphism* of G if $\pi(v)$, $\pi(w)$ is in E whenever (v, w) is in E. The set of all automorphisms of the graph G is a permutation group, called the *automorphism group*, $Aut(X)$, of the graph. If the automorphism group of a graph G is the trivial group, then G is said to be *rigged.*

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. G_1 is *isomorphic* to G_2 if there exists a one-to-one map ι from V_1 onto V_2 such that (ι (v), ι (w)) is an edge of G_2 whenever (v, w) is an edge of G_1.

When G and G_1 are isomorphic graphs, then the number of isomorphisms from G to G_1 is equal to the order of $Aut(G)$.

A graph is represented by its *adjacency matrix*. A graph with n vertices, is represented by a matrix, where the entry is "1" if there is an edge linking the vertex i to the vertex j and is "0" otherwise. For undirected graphs, the adjacency matrix is symmetric around the diagonal.

There are two main generalizations of the graph isomorphism: subgraph problem (given two graphs, determine if one of them is a subgraph to another) and largest common subgraph problem (given two graphs, determine the common subgraph to both that has the maximum number of vertices or edges).

B. Vsep Algorithm for Computing Graph Isomorphism

S. Stoichev published a number of papers that describe the algorithms he created, i.e. algorithms *Vsep*, exact (*Vsep-e*) and heuristic (*Vsep-h1, Vsep-h2*) for determining the generators, orbits and order of an undirected graph automorphism group (see [17]).

A basic tool of these algorithms is the adjacency *refinement procedure* that gives finer output partition on a given input partition of graph vertices. The refinement procedure is a simple iterative algorithm based on the criterion of relative degree of a vertex toward a basic cell in the partition. In the proposed algorithms is used a search tree (ST) in which each node is a partition. A non-singleton cell with maximum partitioning ability is selected in this partition. The partition of a given node of the tree is obtained from the parent-node partition by setting in a separate cell a vertex in the selected cell. This process is called *individualization*. For this vertex is determined that it is not similar to a previous vertex in the cell till the moment of the selection. Then the process of a refinement starts. These processes of individualization and refinement (denoted as IR) continue until a discrete partition is obtained. Then, a move back to the parent-partition follows. A move back takes place also after the whole selected cell of a given selection level has been traversed. That way a tower of finer partitions on every path of the search tree is obtained. The initial partition that is a result of a refinement of the input partition is at the top of the tower (root of the tree). The algorithm stops when the whole selected cell of the root of the tree has been traversed. All nonequivalent discreet partitions derivative of the selected vertices, called by the author “bouquet” are stored in a coded form in a hash table in order to reduce the necessary storage. The codes of two partitions are compared (instead of comparing the partitions) and if they are equal, then the partitions are compared to determine if they form an authomorphism. A disadvantage of the exact algorithm Vsep-e as it is noted in [17] is its higher requirements for memory, for instance in some worst cases several millions of numbers are stored. The worst cases for the algorithm *Vsep-e* are the graphs with smaller order $|Aut(G)|$, especially the rigid graphs. The heuristic algorithms

Vsep-h1, *Vsep-h2* are extremely fast (with some exceptions) compared with the exact one and are "almost exact". Their requirements for memory are very small.

III. Suggested Protocol

Zero Knowledge Proof Protocol Based on Graph Isomorphism Problem is as follows:

Given two isomorphic graphs G_1 and G_2 such that $G_2 = \iota(G_1)$, i.e. graphs G_1 and G_2 are public key and isomorphism ι is a secret key.

1. Prover randomly selects $a \in \{0,1\}$, i.e. tosses a coin.
2. Prover chooses a random permutation μ and generates new graph $G_3 = \mu(G_a)$.
3. Prover sends the adjacency matrix of graph G_3 to the Verifier.
4. Verifier sends $b \in \{0,1\}$, to the Prover and challenges for λ which maps G_3 to G_b.
5. If $a = b$, the Prover sends $\lambda = \mu^{-1}$ to the Verifier.
6. If $a = 0$ and $b = 1$, the Prover sends $\lambda = \mu^{-1} \cdot \iota$ to the Verifier.
7. If $a = 1$ and $b = 0$, the Prover sends $\lambda = \mu^{-1} \cdot \iota^{-1}$ to the Verifier.
8. Verifier checks if $\lambda(G_3) = G_b$ and grants access to the Prover accordingly.

Several rounds of these interactions are needed for the Verifier to be completely convinced of the Prover's identity, since the Prover can be lucky and guess the value of b before sending G_3. As is well known, the probability that this happens is $\frac{1}{2^n}$, with n being the number of rounds. Therefore, with several rounds, this probability is considerably low, and the level of confidence that the Verifier will gain about the identity of the Prover is $1 - \frac{1}{2^n}$. Of course, the presented protocol needs to satisfy the following properties: *soundness*, *completeness* and *zero-knowledge*

IV. Graphs that are appropriate for implementation of the ZKP protocols uggested Protocol

Before the actual implementation of the graph isomorphism based ZKP protocol, there are several important questions that one could ask. Even though the graph isomorphism problem is not known to be solvable in polynomial time for the general case, there are different types of graphs for which polynomial algorithms exist. Therefore, a question that one might ask is: what types of graphs are appropriate for the implementation of the ZKP protocols. There are some problems to be solved in this direction. Since the graphs G_1 and G_2 constitute the public keys, which all the parties involved in the interactions must have, envisage the case where the Verifier or any other party possess a powerful algorithm such as *Vsep.* Then he will in considerable low time (depending on the graph) to obtain the secret permutation μ which has been use to generate G_2. Knowing the secret μ will allow him to impersonate the Prover to the Verifier or to a third party. Another problem to be considered deals with the interception of graph G_3.

Taking into account the foregoing problems, it is clear how important is to find graphs that are very complex for the isomorphism problem. Since (see [17]), *Vsep* is one of the fastest available algorithms, we need to find the graphs that are hard for *Vsep.* Of course, it is good such graphs to be hard for other known and available algorithms as well. The search for "hard" graphs is not easy (see for instance [4] and [12]). It is well known that some graphs must not be used to ZKP protocols based on graph isomorphism problem. These are trees, random graphs, planar graphs, graphs with bounded eigenvalue multiplicity, and graphs with bounded genus, permutation graphs and convex graphs ([12]). On the other hand, the currently known hard instances for canonical labeling algorithms are (see [13]): *Projective Planes; Random Regular Graphs; Strongly Regular Graphs; Grid Graphs; Hadamard Matrix Graphs; Miyazaki Graphs.*

As it is reported in [17] for *Vsep-e* "hard" graphs are those with small order $|Aut(G)|$, especially the rigid graphs. For instance, the search tree for A50 rigid graph (benchmark graph from [11]) given by *Vsep-e* algorithm is shown in *Fig.1* [17]:

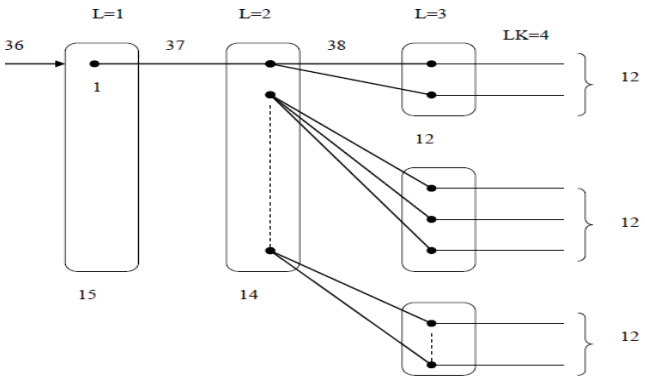

Fig.1. *Search tree for a rigid graph A50*

Our suggestion is to use these "hard" types of graphs for the implementation of the ZKP protocols based on graph isomorphism. Because the heuristic algorithms (like *Vsep-h1*, *Vsep-h2)* are very fast, another approach is to build the ZKP protocol is to limit the time of the Prover to provide the answers to the challenges of the Verifier.

V. Conclusion

In this paper we have presented the graph isomorphism based ZKP protocol. We suggested the *Vsep* algorithm for establishing the graph isomorphism. We considered some of the known hard graphs for the graph isomorphism problem and recommended the types of graphs that are suitable for the suggested protocol. As the future work, we want to

compare implementation of different algorithms for each types of hard graphs and find the most suitable one.

REFERENCES

[1] H. Aronsson, “Zero Knowledge Protocols and Small Systems”, Department of Computer Science, Helsinki University of Technology (1995).

[2] T. Beth, "Efficient zero-knowledge identification scheme for smart cards." Advances in Cryptology—EUROCRYPT’88. Springer Berlin Heidelberg, 1988.

[3] European Commission, “EU Cybersecurity plan to protect open internet and online freedom and opportunity”, Brussels, 7 February 2013.

[4] S. Fortin, “The graph isomorphism problem,” Technical Report TR 96-20, July 1996.

[5] S. Goldwasser, S. Micali and C. Rackoff, “The knowledge complexity of interactive proof systems” , in *Proc.* of STOC 1985, pp. 291-304.

[6] S. Goldwasser and S. Micali,” Probabilistic Encryprion”, JCSS Vol. 28. No. 2. April 1984.

[7] C. Hoffmann, “Group-Theoretic Algorithms and Graph Isomorphism”,Springer-Verlag, New York 1982.

[8] T. Junttila and P. Kaski, “Engineering an efficient canonical labeling tool for large and sparse graphs,” in Proc. of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX07), SIAM, 2007.

[9] B. D. McKay, “Practical graph isomorphism,” Congressus Numerantium, vol. 30, pp. 45-87, 1981.

[10] B. D. McKay, “*nauty* User’s Guide (Version 2.2)” Computer Science Department, Australian National University, 2002.

[11] B.D. McKay and A. Piperno, Practical Graph Isomorphism, II, Journal of Symbolic Computation, 60 (2014), pp. 94-112.

[12] M. Kutz and Pascal Schweitzer, “ScrewBox: a randomized certifying graph-non-isomorphism algorithm”, ALENEX 2007.

[13] A. Piperno, “Search space contraction in canonical labeling of graphs,” CoRR abs/0804.4881, 2008.

[14] J. Presa, “Efficient algorithms for graph isomorphism testing”, Doctoral Thesis, Madrid 2009.

[15] U. Schöning, “Graph isomorphism is in the low hierarchy,” in Proc. of the 4th Annual Symposium on Theoretical Aspects of Computer Science, pp. 114–124, 1987.

[16] M. Schukat and P. Flood, “Zero-Knowledge Proofs in M2M Communication”, 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014). 25th IET, Limerick, 2014, pp. 269-273.

[17] S. D. Stoichev, Vsep –New Heuristic and Exact Algorithms for Graph Automorphism Group Computation, arXiv:1007.1726.