Authenticating Grid Using Graph Isomorphism Based Zero Knowledge Proof

Worku B. Gebeyehu*, Lubak M. Ambaw*, M.A. Eswar Reddy, and P.S. Avadhani

Dept. of Computer Science & Systems Engineering, Andhra University, India
{workubrhn,eswarreddy143}@gmail.com,
{rosert2007,psavadhani}@yahoo.com

Abstract. A zero-Knowledge proof is a method by which the prover can prove to the verifier that the given statement is valid, without revealing any additional information apart from the veracity of the statement. Zero knowledge protocols have numerous applications in the domain of cryptography; they are commonly used in identification schemes by forcing adversaries to behave according to a predetermined protocol. In this paper, we propose an approach using graph isomorphism based zero knowledge proof to construct an efficient grid authentication mechanism. We demonstrate that using graph isomorphism based zero knowledge proof provide a much higher level of security when compared to other authentication schemes. The betterment of security arises from the fact that the proposed method hides the secret during the entire authentication process. Moreover, it enables one identity to be used for various accounts.

Keywords: Zero Knowledge proof, zero knowledge protocol, graph isomorphism, grid authentication, grid security, graph, prover, verifier.

1 Introduction

The intention of authentication and authorization is to deploy the policies which organizations have devised to administer the utilization of computing resources in the grid environment. According to Foster in grid technology is described as a "resource-sharing technology with software and services that let people access computing power, databases, and other tools securely online across corporate, institutional, and geographic boundaries without sacrificing local autonomy"[1]. For example a scientist in research institute might need to use regional, national, or international resources within grid-based projects in addition to using the major network of the campus. Each grid-project mainly requires its own authentication mechanisms, commonly in the form of GSI based or kerberos based certificates so as to authenticate the scientist to access and utilize grid based resources.

* Corresponding authors.

S.C. Satapathy et al. (eds.), *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of CSI - Volume I*, Advances in Intelligent Systems and Computing 248,
DOI: 10.1007/978-3-319-03107-1_2,

These days, computer scientists are exerting lots of efforts to develop different kinds of authentication mechanisms that provide strong Grid security. The currently existing grid authentication mechanisms are usually bound with only one, in most cases based on public key infrastructure (PKI). Such system unnecessarily limits users since they are required to use only the one mechanism, which may not be flexible or convenient.

Moreover, regardless of particular type of certificates or PKI, one of the key drawbacks of the PKI is that current tools and producers for certificate management are too complicated for users. This leads either to rejection of the PKI or to insecure private-key management, which dis-empowers all the Grid infrastructure [2]. This paper proposes a novel methodology to overcome the aforementioned limitations by implementing ZKP based grid authentication.

2 Basic Concepts

2.1 What Is Zero-Knowledge Proof?

A zero-knowledge proof (ZKP) is a proof of some statement which reveals nothing other than the veracity of the statement. Zero-Knowledge proof is a much popular concept used in many cryptography systems. In this concept, two parties are involved, the prover A and the verifier B. Using this technique, it allows prover A to show that he has a credential, without having to give B the exact number.

The reason for the use of a Zero-Knowledge Proof in this situation for an authentication system is because it has the following properties:

- Completeness: If an honest verifier will always be convinced of a true statement by an honest prover
- Soundness: If a cheating prover can convince an honest verifier that some false statement is actually true with only a small probability.
- Zero-knowledge: if the statement is true, the verifier will not know anything other than that the statement is true.

Information about the details of the statement will not be revealed. [3]. A common application for zero-knowledge protocols is in identification schemes, due to Feige, Fiat, and Shamir[4].

2.2 Graph

A graph is a set of nodes or vertices V, connected by a set of edges E. The sets of vertices and edges are finite. A graph with n vertices will have: V = {1, 2, 3,..., n} and E a 2-element subsets of V. Let u and v be two vertices of a graph. If (u,v)ЄE, then u and v are said to be adjacent or neighbors.

A graph is represented by its adjacency matrix. For instance, a graph with n vertices, is represented by a nxn matrix M=[$m_{i,j}$] , 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.

2.3 Graph Isomorphism

Two graphs G1 and G2 are said to be isomorphic, if a one-to-one permutation or mapping exists between the set of vertices of G1 and the set of vertices of G2, with the property that if two nodes of G1 are adjacent, so are their images in G2. The graph isomorphism problem is therefore the problem of determining whether two given graphs are isomorphic. In other words, it is the problem of determining if two graphs with different structures are the same. Figure 1 gives an example of isomorphic graphs, with their corresponding adjacency matrices. Notice that the entries of the matrices, where m_{ij} =0 are left blank.

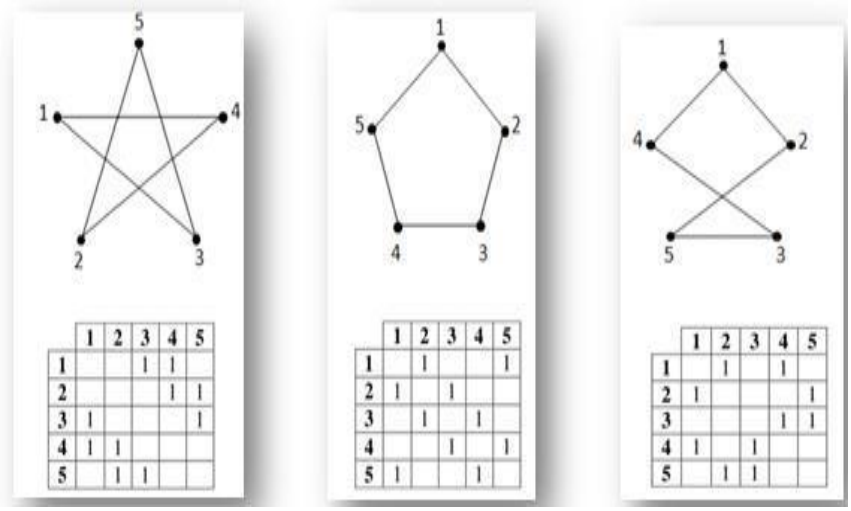


Fig. 1. Example of Isomorphic Graphs with their corresponding adjacency matrices

For example, Graph (b) is found, by relabeling the vertices of Graph (a) according to the following permutation: (3, 5, 2, 4, 1). This means that Node 1 in Graph (a) becomes Node 3 in Graph (b), Node 5 becomes Node 1 and so on. Following in Fig.2 is an illustration of how the permutation is applied to Graph (a).

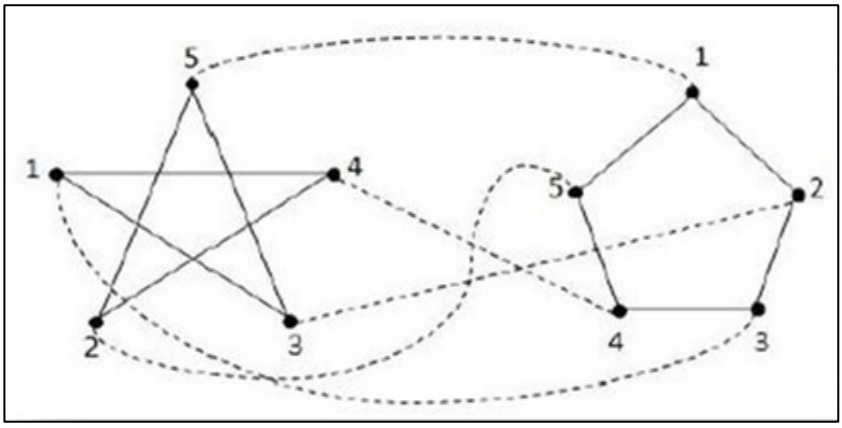


Fig. 2. Relabeling the vertices of Graph (a) using the permutation (3,5,2,4,1)

3 Proposed Method

3.1 Graph Isomorphism Based Zero-Knowledge Proofs

The purpose of Zero-Knowledge Proof (ZKP) protocols is to enable the prover to prove an assertion to the verifier that he holds some secret knowledge ,without leaking any information about the knowledge during the verification process (zero-knowledge).To demonstrate this concept ,we chose to implement ZKP protocol based on Graph Isomorphism (GI).In this protocol the input to the prover and the verifier is a pair of graphs G0 , G1 , and the goal of the prover is to convince the verifier that the graphs are isomorphic, but without revealing any information. If G0 and G1 are isomorphic, then the prover also has a permutation π such that $\pi(G0) = G1$ (i.e., π is an isomorphism).

Suppose we have two graphs G1 and G2 where G2 is generated from G1 using a secret permutation named π. G2 is obtained by relabeling the vertices of G1 according to secret permutation π by preserving the edges. The pair of graphs G1 and G2 forms the public key pair, and the permutation π serves as the private key. A third graph Q is either obtained from G1 or G2 using another random permutation ρ. Once the graph Q is found, the prover (represents the new node seeking entrance to the grid)sends it to the verifier who will challenge him to provide the permutation σ which can map Q to either G1 or G2.

For example, if Q is found from G1 and the verifier puts a challenge to the prover to map Q to G1, then $\sigma = \rho\text{-}1$. In the same fashion, if Q is obtained from G2 and the verifier challenges the prover to map Q to G2, then $\sigma = \rho\text{-}1$. Otherwise, if Q is obtained from G1 and the verifier puts a challenge to the prover to provide the permutation that maps Q to G2, then $\sigma = \rho\ \text{-}1 \circ \pi$, which is a combination of ρ-1and π. Indeed, ρ-1will be applied to Q to obtain G1 then the vertices of G1 will be modified according to the secret permutation π to get G2. Eventually, if Q is obtained from G2 and the verifier challenges the prover to map Q to G1, then $\sigma = \rho\text{-}1 \circ \pi\text{-}1$.

One can observe that in the first two cases, the secret permutation π is not even used. Thus, a verifier could only be certain of a node's identity after many interactions. Moreover, we can also observe that during the whole interaction process, no clue was given about the secret itself this makes it strong grid authentication mechanism.

3.2 Pseudo Code

Given G1 and G2 such that G2 = π (G1), the interactions constituting iterations of the graph isomorphism based ZKP protocol are demonstrated below:

Step 1: Prover randomly selects x Є {1,2}
Step 2: Prover selects a random permutation ρ, and generates Q=ρ(Gx)
Step 3: Prover communicates the adjacency matrix of Q to the verifier
Step 4: Verifier communicates y Є {1,2}to prover and challenges for σ that maps Q to Gy
Step 5: If x=y the prover sends $\sigma = \rho^{-1}$ to the verifier
Step 6: If x=1 and y=2 the prover sends $\sigma = \rho^{-1} \circ \pi$ to the verifier
Step 7: If x=2 and y=1 the prover sends $\sigma = \rho^{-1} \circ \pi^{-1}$ to the verifier
Step 8: Verifier checks if σ (Q)=Gy and access is granted to the prover accordingly

A number of iterations of these interactions are needed for the verifier to be totally convinced of the prover's identity, as the prover can be lucky and guess the value of y before sending Q.

3.3 Prototype

The researchers have implemented a prototype version of graph isomorphism based ZKP protocol for authenticating a grid. Graph isomorphism has been chosen because of its ease of implementation. The JAVA-implementation of the ZKP protocol in authentication of the grid is presented below. The advantage of this implementation is that it allows us to verify the adjacency matrices at each level of the simulation and to evaluate the correctness of the algorithm. The JAVA implementations are mainly based on the pseudo-code provided in Pseudo code section above. So as to demonstrate the logic used in the program, we considered a simple graph with 4(four) nodes. In reality, however, the graphs need to be very large, consisting of several hundreds of nodes (in our case the number of users requesting for grid resources), and present a GI complexity in the order of NP-complete.

There are three functions that are used in the implementation, namely, "Generate_Isomorphic", "Permutuate", and "Compare_Graphs" also one main function.

a) "Generate_Isomorphic" function

Generate_Isomorphic function enables us to apply a random permutation (π or ρ) to a specific graph and to get another graph that is isomorphic to the first graph. For example, the declaration G2=Generate_Isomorphic(G1,pi,n)" is used to apply π to the

graph G1 so as to get the graph G2; where the variable „n‟ represents the number of nodes in the graph(the number of users requesting for grid resources).

b) "Permutuate" function

This function is used after receiving the challenge response from the prover. Indeed, once the verifier receives σ from the prover, his objective is to apply that permutation to the graph Q to check if he will get the expected response G1 or G2. This function is declared as follows, "Permutuate (Q,sigma,n)" where σ is being applied to a graph Q.

c) "Compare_Graphs" function

Once the verifier receives the response σ and applies it to Q to get another graph that we will refer it as R for explanation purpose, the "Compare_Graphs" function is used to check whether or not R = G1 or R = G2 depending on the case.

The function is declared as "Compare_Graphs(G2,R,n)." Here, for instance, the function is used to compare both graphs G2 and R.

4 Simulations and Results

In this section we will depict how the implemented algorithm works through simulation. At first, the simulator requests for the value of "n" which represents the size of the graphs (In practise the number of grid users requesting for resource at one point of time). Once the size is set(the value of „n‟ is known), the user has the choice to either launch the interaction or to stop the simulation. The user is then prompted to provide the values of the graph G1. After all the values are entered, the user is asked for the value of the secret π and then for the value of the random permutation ρ. The user is then requested to choose the graph from which to create the graph Q and the graph that the verifier could ask the prover to generate when challenged. After these graphs are specified, the user is eventually asked for the value of σ. Once the value of σ entered, the simulator verifies all the inputs and declares if the prover is correct or incorrect. In other words, the simulator verifies if the prover knows the secret or not. In the real environment it is similar to proving the veracity of the grid user before allowing access to the grid. At last, the user is requested to repeat the process if he/she is willing. In order to test the correctness of the algorithm we simulate the code with some input examples and validate the results. In fact, the simulations were ran with the values of n, π, ρ and the graph G1 illustrated below.

n=4

π={2,3,1,4}

ρ={1,3,4,2}

	1	**2**	**3**	**4**
1	0	1	0	0
2	1	0	1	1
3	0	1	0	1
4	0	1	1	0

Fig. 3. Graph G1 and its Adjacency Matrix representation

If Q is obtained from G1 (Q = G1* ρ) and the verifier challenges for G1, we will have the equation:

$$\sigma = \rho^{-1} \tag{1}$$

From equation 1 we have $\sigma = \rho^{-1}$ = {1, 4, 2, 3} and to get the inverse of ρ = {1, 3, 4, 2}, we assign indices running from 1 to 4 to its vertices, and then process as follows. The first index "1" is at position 1, so its position remains the same. Index 2 is at the position 4; so the second vertex of ρ^{-1} is 4. Index 3 is at the second position; therefore the next vertex in ρ^{-1} is 2. Finally, index 4 is at the third position; therefore, the last element of ρ^{-1} is 3. Following this logic, the vertices of ρ^{-1} are obtained. If Q is obtained from G2 (Q = G2* ρ) and the verifier challenges for G2, we will have σ = ρ = {1, 4, 2, 3}.If Q is obtained from G1 (Q = G1* ρ) and the verifier challenges for G2, we will get the equation

$$\sigma = \rho^{-1} \circ \pi \tag{2}$$

From equation 2 we have $\sigma = \rho^{-1} \circ \pi$ = {1, 4, 2, 3} ○ {2, 3, 1, 4} = {2, 4, 3, 1}.To equate and get the value of σ, we consider the vertices of ρ^{-1} as indices in π and observe which values they are associated with. The whole process consists of the following steps namely the first vertex of π is 2, the fourth vertex of π is 4. and the second vertex of π is 3 and the third vertex of π is 1. If Q is obtained from G2 (Q = G2* ρ) and the verifier challenges for G1, we will have the equation:

$$\sigma = \rho^{-1} \circ \pi^{-1} \tag{3}$$

From equation 3 we have $\sigma = \rho^{-1} \circ \pi^{-1}$ = {1, 4, 2,3} ○ {3, 1, 2, 4} = {3, 4, 1, 2}.Now that all the parameters are set, we can run the actual simulation. One can note that the permutations are applied to the rows as well as the columns of the adjacency matrix in the above process. Simulation results will be depicted in the following section so as to demonstrate the three different cases (i.e. $\sigma \rho^{-1}$, $\sigma = \rho^{-1} \circ \pi$, and $\sigma = \rho^{-1} \circ \pi^{-1}$).

```
The value of n is: 4
Start the interaction?(1:Yes,0:No):1
The value of the original graph G1 is: G1=[0 1 0 0;1 0 1 1;0 1 0 1;0 1 1 0]
The value of the secret pi is :[2 3 1 4]
The value of ro is :[1 3 4 2]
Choose the graph from which to create the graph H to give the verifier[0:G1;1:G2]:0
The verifier randomly chooses a graph for the prover to generate [1:G1;2:G2]:0
The value of sigma is :[ 1 4 2 3]
The prover has proved himself!

Does the  verifier want to repeat the process? (1:Yes;0:No): 0
```

Fig. 4. Demonstration of the case in which Q = G1 *ρ and when the verifier challenges the prover to map Q to G1

After the first simulations is over we obtained G2 = G1*π and Q = G1* ρ as depicted in figure 3 and figure 4 below respectively.

	1	2	3	4
1	0	0	1	1
2	0	0	1	0
3	1	1	0	1
4	1	0	1	0

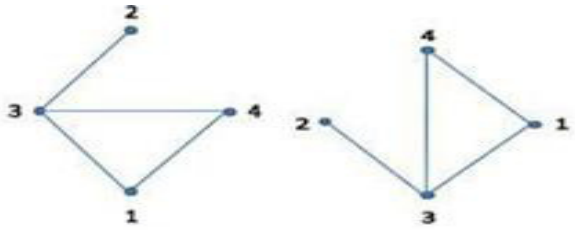
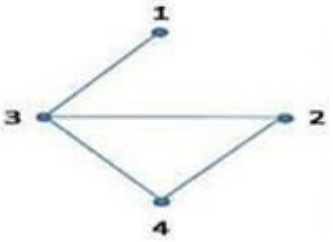


Fig. 5. Graph G2 in two distinct forms and its adjacency matrix



	1	2	3	4
1	0	0	1	0
2	0	0	1	1
3	1	1	0	1
4	0	1	1	0

Fig. 6. Demonstration of graph Q = G1 * and its adjacency matrix

At last, let us see the situation where the prover enters incorrect responses to the challenges. The following figure depicts this case.

```
The value of n is:4
Start the interaction? (1: Yes, 0: No):1
The value of the original graph G1 is: G1=[0 1 0 0;1 0 1 1;0 1 0 1;0 1 1 0]
The value of the secret pi is:[2 3 1 4]
The value of ro is:[1 3 4 2]
Choose the graph from which to create the graph ϕ to give the verifier (0:G1,1:G2):0
The verifier randomly chooses a graph for the prover to generate(1:G1,2:G2):1
The value of sigma is:[1 2 4 3]
The prover is proved wrong! He does not know the secret!

Does the verifier want to repeat the process?(1:Yes,0:No):1
Choose the graph from which to create the graph ϕ to give the verifier (0:G1,1:G2):0
The verifier randomly chooses a graph for the prover to generate(1:G1,2:G2):0
The value of sigma is:[2 1 3 4]
The prover is proved wrong! He does not know the secret!

Does the verifier want to repeat the process?(1:Yes,0:No):1
Choose the graph from which to create the graph ϕ to give the verifier (0:G1,1:G2):1
The verifier randomly chooses a graph for the prover to generate(1:G1,2:G2):0
The value of sigma is:[1 4 3 2]
The prover is proved wrong! He does not know the secret!

Does the verifier want to repeat the process?(1:Yes,0:No):
```

Fig. 7. Demonstration of the case where a user provides wrong answer for the challenge

5 Conclusion and Future Work

A new approach for authenticating grid using ZKP protocol based on Graph Isomorphism (GI) is proposed. Our simulation results show that the proposed method provides a much higher level of security for authenticating users in the grid by hiding the secret during the entire authentication process. Besides, it enables one identity to be used for various accounts. The implementation could be modified further so that the system is interactive and user friendly. Moreover, one can integrate the implementation of graph isomorphism based zero knowledge proof within a grid portal.

References

[1] Foster, I.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Wiley, Argonne Illinois (2002)
[2] Daniel, K., Ludek, M., Michal, P.: Survey of Authentication Mechanisms for Grids. In: CESNET Conference, pp. 200–204. Czech Academy of Science Press, Prague (2008)
[3] Lum, J.: Implementing Zero Knowledge Authentications with Zero Knowledge. In: The Python Papers Monograph Proceedings of PyCon Asia-Pacific, Melbourne (2010)
[4] Lior, M.: A Study of Perfect Zero-Knowledge Proofs, PhD Dissertation, University of Victoria, British Colombia, Victoria (2008)