

## Using Subgraph Isomorphism as a Zero Knowledge Proof Authentication in Timed Wireless Mobile Networks

JOSEPH M. KIZZA, LINDSAY BRAMLETT AND ELIZABETH MORGAN<sup>8</sup>,

Computer Science and Engineering Department  
University of Tennessee at Chattanooga  
Chattanooga, Tennessee

---

### Abstract:

In networks with sensitive resources and time critical missions, user identification and authentication is essential and critical for all secure access communications originating from outside into the networks. However, there are networks and times when entity identification is not required and indeed in some networks, for the security of the outside entity seeking authentication, identity must not be revealed thus preserving the secrecy and privacy of such entities. Zero knowledge protocol (ZKP) using subgraph isomorphism to authenticate new entities requiring entry into existing mobile network while preserving entity privacy and identity works well in these instances.

### Categories and Subject Descriptors: C.2.2

**General Terms:** Computer Networks, Protocols

**Key Words:** Zero-knowledge, PKI

---

### IJCIR Reference Format:

Joseph M. Kizza, Lindsay Bramlett and Elizabeth Morgan. Using Subgraph Isomorphism as a Zero Knowledge Proof Authentication in Timed Wireless Mobile Networks. *International Journal of Computing and ICT Research*, Special Issue Vol. 4, No. 1, pp. 81 -9. <http://www.ijcir.org/Special-Issuevolume4-number1/article9.pdf>.

---

## 1. INTRODUCTION

Communication allows for the transfer of data between two or more entities. This transfer of data and knowledge allows for the growth and prosperity of our society. The challenge faced is how to determine the data used to make decisions is valid and trustworthy. Each side in the exchange must be able to trust the other side and the data being exchanged.

In networking, there are many protocols that allow for the verification of nodes attempting to join the selected network. Different protocols require varying amounts of time to authenticate a new entity. As network technology continues to change it is important to minimize the latency of authentication protocols while still protecting the resources and data of the network.

The development of wireless mobile networks has driven research into network authentication protocols that may exclude a trusted third party to verify the identity of entities before allowing communication between the newcomer into an already existent network. This paper describes a zero knowledge protocol (ZKP), one such authentication protocols. The ZKP protocol makes use of subgraph isomorphism to authenticate new entities into the secure mobile network. The paper is divided into the following sections: related works, research of mobile networks, existing authentication protocols, zero knowledge protocols, graph and subgraph isomorphism, an algorithm for implementing a zero knowledge protocol using subgraph isomorphism, a comparison of PKI to the proposed, and the future of this work.

---

<sup>8</sup> Author's Address: Joseph M. Kizza, Lindsay Bramlett and Elizabeth Morgan, Computer Science and Engineering Department, University of Tennessee at Chattanooga, Chattanooga, Tennessee

"Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than IJCIR must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee."

© International Journal of Computing and ICT Research 2010.

International Journal of Computing and ICT Research, ISSN 1818-1139 (Print), ISSN 1996-1065 (Online), Special Issue, Vol.4, No.1, pp. 81 - 91, October 2010.

## 2. RELATED WORK

Several interesting zero knowledge proofs are of note here starting with Hannu Aronsson [1995] work which explains zero knowledge proofs starting with the basics including a summary of the entire major zero knowledge studies. Li Lu et al [2006] discuss zero knowledge authentication in P2P systems. However, they use a modified ZKPI scheme that uses a key exchange but with no third party. Stawomir Grozonkowski et al [2008] propose a zero knowledge authentication scheme similar to ours using graph isomorphism. However their application is based on web applications. None of these and other works is using zero knowledge in Mobile Networks with their stringent constraints.

## 3. MOBILE NETWORKS

A Wireless Mobile Network (WMN) is an infrastructure that provides communication services through at least one wireless node that is capable of motion. This type of network is transportable and flexible in allowing asynchronous communication between nodes in an area.

We consider a timed wireless mobile network in which communication between entities is very time sensitive. Entities requesting access to the network need a fast authentication protocol, because they will not exist in the network's area for long periods of time. Two entities are essential in the authentication process. The first entity/node, the Verifier is within the mobile network. It is responsible for the authentication of the second entity, the Prover, which is outside the network but seeking the use of the network resources, thus entry into the network. The Verifier, therefore, must be able to determine beyond a doubt that the entity seeking network resources is trustworthy before it is granted permission. The network's security is another challenge to the development of an mobile network. Malicious users cannot be allowed access to the data being exchanged within this network. Any security protocol used must allow only those who can be trusted [Cryptosys, 2010].

## 4. ZERO KNOWLEDGE PROOFS

A Zero Knowledge Proof is a method of proving one's identity to another without revealing anything except that a statement is true. The statement is usually a mathematical one, and also it revolves around a secret. The "Prover" is the one who is trying to validate to a "Verifier"[Graph Isomorphism, 2010, Aronsson, 1995]. For example:

Once upon a time in the land of Barkfest the noble King Robert sent down a letter to dear farmer John requesting a basket of his favorite purple plums. The king made this request of John, because only John knew where to find the king's favorite fruit.

So John set out for the river George to cross the only bridge onto the island Marigold where he knew he would find the purple plums in his secret plum patch. As he approached the bridge, John prepared to greet his long time friend, the Ogre Jax. But, to John's surprise, a new Ogre stood between him and the only means of getting to the island with the purple plums.

John said, "Where is Jax?"

The Ogre answered, "Oh Jax is on vacation. I'm his cousin Ralph."

John replied, "Good to meet you Ralph. I've come to pick purple plums for King Robert, so I'll be on my way."

Ralph moved into John's path blocking the crossing over the bridge, "No man crosses this bridge by decree of the Elder Ogres!"

John was shocked because he had been over the bridge so often to get the king's favorite fruit. He said, "I've crossed this bridge many times. Jax knows me well and I'm the only person who knows where to find the plums."

Ralph asked, "So you say you've been on the island... are there any rose bushes along your way to the plums?"

John replied, "No."

Very good, thought Ralph. There weren't any rose bushes on the entire island. Ralph asked, "Do bluebirds nest near the plums?"

John replied, "Yes."

Ralph thought before asking his next question. If there are bluebirds near the plums then this patch must be on the north side of the island. There are many red daisy plants up there too. John would have seen them if he had actually been on the island. "What color are the daisies where your plums are?"

John replied, "Red."

Ralph is beginning to trust that John has been allowed to cross over to the island before today to pick these plums. He decided to ask a final question before fully trusting farmer John. “Will you be heading to the north or south end of the island?”

John said, “North.”

Ogre Ralph was convinced that John had been allowed on the island before. He stepped out of John’s way and let the farmer go about his business though never knowing where exactly John was headed to pick those delicious purple plums.

A Zero Knowledge Proof must satisfy these three properties: Completeness, Soundness, and Zero-Knowledge. Completeness is that if the statement is true, the good Verifier will be convinced if the Prover is honest. Soundness is that if the statement is false, a deceitful Prover cannot convince an honest Verifier that it is true. There is a very small probability that they can. Zero-knowledge is that if the statement is true, a deceitful Verifier cannot learn anything about the honest Prover [Graph Isomorphism, 2010].

## 5. ALTERNATE AUTHENTICATION METHODS

The public key infrastructure (PKI) was created as a means of verifying users on the Internet. This authentication protocol uses digital certificates stored by a trusted third party, known as the Certificate Authority. The digital certificates are publically available and often known as public keys. Each digital certificate uniquely identifies an individual or organization. A Registration Authority is used as a verifier to guarantee the Certificate Authority can be trusted.

The Certificate Authority creates a public and private key using the same algorithm. This allows for a sender to encrypt a message with the receiver’s public key. When the message is received the receiver will decrypt using his private key. In return, a user can authenticate to another user by encrypting a message with his private key. The receiver, in this case, will decrypt the message using the sender’s public key retrieved from the Certificate Authority.

Public key infrastructure has the benefit of a trusted third party. The existence of a public key with a Certificate Authority for a sender verifies to the receiver the message can be trusted and how the message can be decrypted. The security of public key infrastructure breaks down if the private key is discovered or intercepted by a malicious person. This intruder or man-in-the-middle can then intercept and decrypt messages. The intruder is capable of impersonating the user as well.

The public key infrastructure authentication method doesn’t work well in zero knowledge protocols in an mobile network because of its use of a third party. The delay accrued waiting for the third party to check the directories for a user’s public key is too great to make the protocol a viable solution. Also, the parties must be able to verify between themselves that the other user can be trusted since there isn’t a third party available to ask.[Aronsson, 1995]

## 6. GRAPH ISOMORPHISM

Isomorphism is a bijection between the vertex sets of graph  $G_1$  and graph  $G_2$  such that any two vertices  $u$  and  $v$  of  $G_1$  are adjacent if and only if  $f(u)$  and  $f(v)$  are adjacent in  $G_2$  [Li et al, 2006]. The graph isomorphism problem attempts to find an isomorphism between two graphs. The problem is classified as NP.

Graph isomorphism was not considered for use in this project because algorithms such as *Nauty*, *VF* and *Saucy2* exist that throw doubt to the hardness of finding an isomorphism between two graphs. This casts doubt of the security of the use of graph isomorphism protocol in Mobile Networks. Instead, we used subgraph isomorphism which is known to be an NP-hard problem and yet may have a solution.

## 7. SUBGRAPH ISOMORPHISM

Subgraph isomorphism is an improvement over the use of graph isomorphism in the zero knowledge protocol. The improvement comes from subgraph isomorphism being an NP-hard problem and therefore more difficult for a malicious user to solve. The graph isomorphism problem has been solved and therefore is vulnerable to attacks by malicious users. In both methods, the more nodes present in each graph, the more difficult the isomorphism is to break.

Any protocol using subgraph isomorphism will start with two public graphs,  $G_1$  and  $G_2$ . The Prover has a secret that he keeps from all parties. The Prover knows an isomorphism between graph  $G_1$  and a subgraph,  $H$ , of graph  $G_2$ .

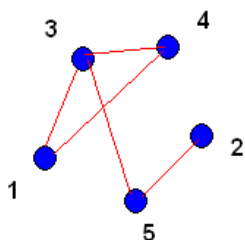


Figure 1: Graph  $G_1$

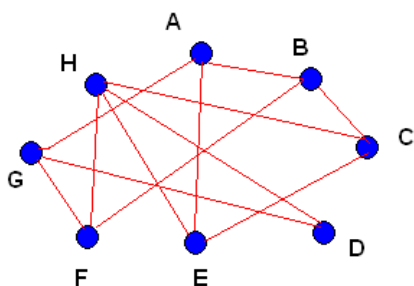


Figure 2: Graph  $G_2$

In the beginning, the Prover will create an isomorphism function and apply it to graph  $G_2$ . This will create an isomorphic graph of  $G_2$ , known as  $G_3$ . To start the authentication process the Prover will send the new graph  $G_3$  to the Verifier.

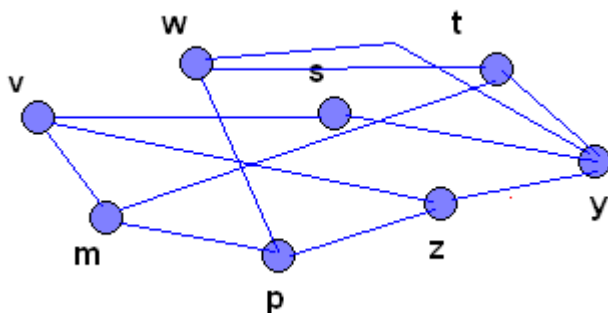


Figure 3: Graph  $G_3$

In response, the Verifier sends the Prover a bit,  $c$ , which is either 0 or 1. If 0, the Prover will send graph  $G_2$  and the isomorphism function used to create graph  $G_3$ . If 1, the Prover will send graph  $G_1$  and the isomorphism that proves  $G_1$  is isomorphic to a subgraph of  $G_3$ .

The Prover's secret is never compromised during this authentication exchange. Each round will prove either the Prover sent a graph that is isomorphic to graph  $G_2$  or the Prover sent a graph containing a subgraph that is isomorphic to graph  $G_1$ . There is no method available for the Verifier to extract the Prover's secret in this protocol.

The verifier will always have doubt that the Prover is a valid user, because the Verifier can never test the Prover's secret. This doubt is part of the risk of using zero knowledge protocols. The verifier's doubt is reduced through the subgraph isomorphism protocol by repeating the above verification steps. As long as the Prover doesn't anticipate the Verifier's response the Prover must be prepared to answer for both 0 and 1. Multiple verification rounds reduce the chance the Prover guessed the Verifier's response. The Prover will gradually gain the Verifier's trust.

Trust is equal to  $1-(\frac{1}{2})^n$  where  $n$  is the number of times the verification process has been run. With each run the Verifier's trust of the Prover moves closer, but never to, 1. The protocol can be set to completely trust a Prover once the trust value has reached a specific threshold such as 95%.

## 6. THE ALGORITHM

Multiple sections of code are needed to create, test, and verify the isomorphism. The Prover must first be capable of creating two public graphs known as ORG and LILORG. The algorithm will allow for a random size for graph ORG, but this isn't implemented in the program. The size of graph ORG is known to the Prover. The Prover creates a random graph and populates the ORG matrix. Next, the Prover creates a random SECRET array. This array is kept by the Prover and never shared. It is the isomorphism between a subgraph H1 of ORG and LILORG. Once the ORG graph and the SECRET array are created the LILORG graph may be created from the known values in ORG. Random generation of the ORG graph and the isomorphism is done using C#'s rand() function and a time element to ensure the values are always random.

Now that the public graphs and the Prover's secret have been created the Prover must create an isomorphism, known as MORPH, which will be used to create a graph, MOR. MOR is isomorphic to the public graph ORG according to the isomorphism MORPH. For each verification round between the Prover and the Verifier the Prover will create a new MORPH array and therefore a new MOR graph.

To use the subgraph isomorphism problem in this protocol, the public graph LILORG must be isomorphic to a subgraph, known as H, of MOR. At the beginning of a verification round the Prover will create an array called LILMORPH. This array is the isomorphism of the subgraph H of graph MOR and the LILORG public graph. Creation of LILMORPH is accomplished using the knowledge of the SECRET and the MORPH isomorphisms that already exist to the Prover.

Now that the Prover has created his public graphs, ORG and LILORG, and has made an isomorphic graph MOR to graph ORG, he is prepared to respond to the Verifier's request whether it is a zero or a one. The Verifier must be able to verify the Prover is being truthful during each round. The Verifier will do this by testing that either MORPH is an isomorphism between graphs ORG and MOR or by testing for a subgraph of MOR, called H, is isomorphic to LILORG according to the isomorphism array LILMORPH the Prover provides. The Verifier will alter his trust level of the Prover and continue verification rounds until either a threshold of trust is met or the Prover has proved to be untrustworthy. (Note: Figures within this document are for demonstration and explanation purposes only.)

The Prover creates an isomorphism by randomly populating an array known as MORPH. The MORPH array is as long as the number of nodes in the public graph, ORG. Each index of MORPH relates to a corresponding vertex of ORG. The random number stored at index 0 will be the node of the isomorphic graph MOR that is isomorphic to vertex 0 of ORG. The random population of the MORPH array is accomplished using C#'s rand() function, adding to that the current system time, and using the modulo operator. This results in a different array each time the program is run.

0	1	2	3	4	5	6	7	ORG
0	0	1	1	0	1	0	0	0

Figure 4: Extract from ORG's matrix for vertex 0 showing edges between vertex 0 to the eight vertexes of ORG

0	1	2	3	4	5	6	7	Vertex of ORG
3	0	7	5	2	6	1	4	MORPH array

Figure 5: Randomly created MORPH array. The index of the array corresponds to the vertex of the ORG public graph. Values in the MORPH array correspond to vertexes of the MOR graph.

By applying the MORPH array to the matrix of the ORG public graph the matrix for a new MOR graph is created. The MOR graph is isomorphic to the ORG public graph according to the MORPH isomorphism function. Figure 3 shows an extract from the isomorphic graph MOR's matrix.

0	1	2	3	4	5	6	7	MOR
3	1	0	1	0	0	0	0	1

Figure 6: Extract from MOR's matrix for vertex 3. Vertex 3 is isomorphic to vertex 0 of the public graph ORG. Vertex 0 of ORG has edges to vertexes 1, 2, and 4. These vertexes are isomorphic to MOR's vertexes 0, 7, and 2 according to MORPH.

The MOR graph is not the only object the Prover must be capable of generating. The Prover possesses the SECRET array. The SECRET array defines the isomorphism between a subgraph, known as  $H_1$ , of the ORG public graph and the LILORG public graph. The Prover will never share the SECRET with the Verifier, but must still prove knowledge of the SECRET.

To prove knowledge of the SECRET without giving it away, the Prover must know the subgraph, known as  $H$ , of the isomorphic graph MOR that is isomorphic to LILORG. This isomorphism, called LILMORPH, is generated after the MORPH isomorphism has been created. The Prover's knowledge of the SECRET and MORPH isomorphisms allow for the creation of the LILMORPH array. The index of LILMORPH is the vertex of LILORG. The value stored at this index is the vertex of MOR isomorphic to this corresponding vertex of LILORG.

0	1	2	3	4	Vertex of LILORG	
7	3	0	5	1	Vertex of ORG	SECRET array

Figure 7: The Prover's SECRET. Stores the vertexes of a subgraph of ORG and the isomorphic mapping to the vertexes of LILORG

0	1	2	3	4	Vertex of LILORG	
4	5	3	6	0	Vertex of MOR	LILMORPH array

Figure 8: LILMORPH array is created after applying the MORPH array to the Prover's SECRET array.

At this point the Prover has created all objects he may need for one verification round with the Verifier. Subsequent verification rounds will require the Prover to create another random MORPH isomorphism, MOR graph, and LILMORPH array.

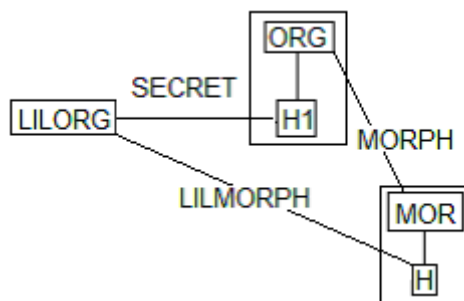


Figure 9: Components of the algorithm and how they relate to each other. Items in boxes are graphs. Lines represent the isomorphism between the two graphs.

MOR is a graph that is isomorphic to the public graph ORG. It is sent to the Verifier at the start of each verification round. The Verifier will store the matrix for this graph for later use during the verification process. The Verifier will now request data from the Prover by randomly choosing between 0 and 1. This is accomplished by using C#'s `rand()` function, adding to that the current system time, and using the modulo operator to get a zero or one.

The Verifier will send the random value to the Prover. If a zero, the Prover will send the matrix for the ORG public graph and the MORPH array. If a one, the Prover will send the matrix for the LILORG

public graph and the LILMORPH array. The Verifier must now be able to confirm the Prover created a graph isomorphic to ORG or the public graph LILORG is isomorphic to a subgraph, H, of MOR.

The Verifier already has the MOR matrix and has sent a zero back to the Prover. The Prover now provides the ORG matrix and the isomorphism array MORPH to the Verifier. The Verifier uses the MORPH array on the isomorphic graph MOR's matrix to create what he calls the UNMOR matrix. Once populated, the UNMOR matrix is compared to the ORG matrix. If any value of UNMOR's matrix is not equal to the value in ORG's matrix the Prover will be proven to be a liar and communication will terminate. If all values in UNMOR's matrix match those of ORG's matrix the Verifier will increase his trust of Prover using this equation:

$$\text{TRUST} = 1 - (\frac{1}{2})^n$$

There is a fifty percent chance the Verifier will send a one to the Prover. In this case the Prover will send the LILORG public graph's matrix and the LILMORPH array to the Verifier. The LILMORPH array will allow the Verifier to determine if the LILORG public graph is isomorphic to a subgraph, H, of MOR which the Verifier already has.

Using the LILMORPH array, the Verifier can extract a subgraph of MOR into a new matrix called LILUNMOR. This LILUNMOR matrix will be compared to LILORG's matrix. If there are any values that are not equal between the two matrices the Prover will be considered untrustworthy and communication will terminate. If all values in LILUNMOR's matrix match those of LILORG's matrix the Verifier will increase his trust of Prover using this equation:

$$\text{TRUST} = 1 - (\frac{1}{2})^n$$

The value of n is understood to mean the verification round number. This number starts at one and is incremented after each round. Verification ends when TRUST is greater than 0.99. At this time the Prover is permitted to communicate with the Verifier.

## 8. SIMULATION AND RESULTS

The simulation was coded in C# using Visual Studio 2005. It uses a GUI window to display the results, a Graph class to hold all the variables used and threads to simulate the authentication. The algorithm was tested in several different ways. One way uses hard coded graphs. Another uses randomly created graphs for ORG and LILORG, with the user being able to select the number of nodes.

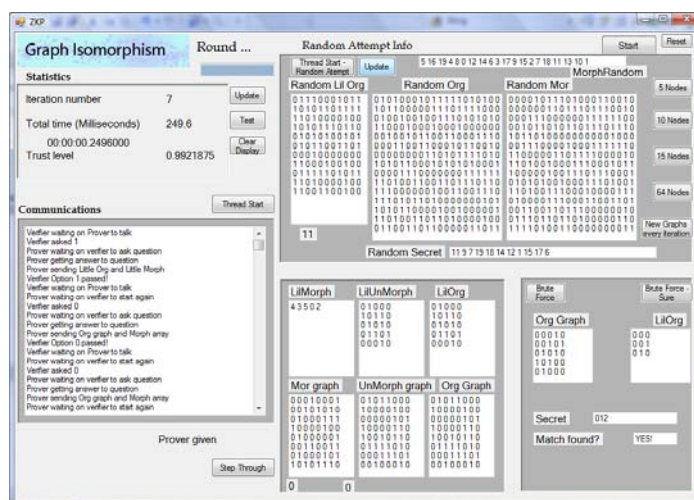


Figure 10

At the top left of Figure 10, there is the iteration number, the total time in milliseconds, and the trust level. Below that is the Communications window. It shows the log of actions between the Prover and Verifier

threads. On the top-right is where we find the area for Randomly Created ORG and LILORG graphs. You can choose the size of your graph by clicking on the corresponding button on the right. Clicking the Update button will show the results of the simulation. Below the Random area on the right is the smaller simulation using hard coded ORG and LILORG graphs. These graphs are 8 and 5 nodes respectively. To the bottom right is where we randomly create 2 graphs, an ORG and LILORG. This section looks for a subgraph of ORG that is isomorphic to LILORG. If a subgraph of ORG is found, the display shows the list of ORG's nodes that make up the subgraph in the textbox labeled Secret.

Below the Communications window is the "step through" button that goes step-by-step through the algorithm. It shows the graphs used in each step, the current trust level and the iteration count.

To collect results the simulation was run seven times and the elapsed time was recorded for each run. The results of several runs based on node size were plotted in several graphs. As you can see in Figure 11, the number of nodes in a graph do not appear to have a bearing on the runtime of the algorithm.

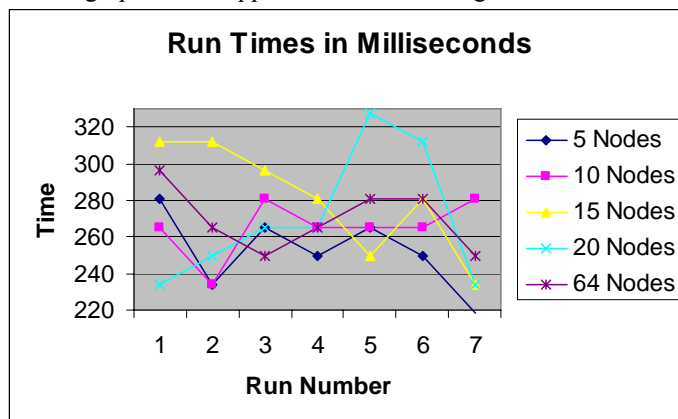


Figure 11

In Figure 12 the average runtime required for the 5, 10, 15, 20, and 64 node graphs to complete is displayed.

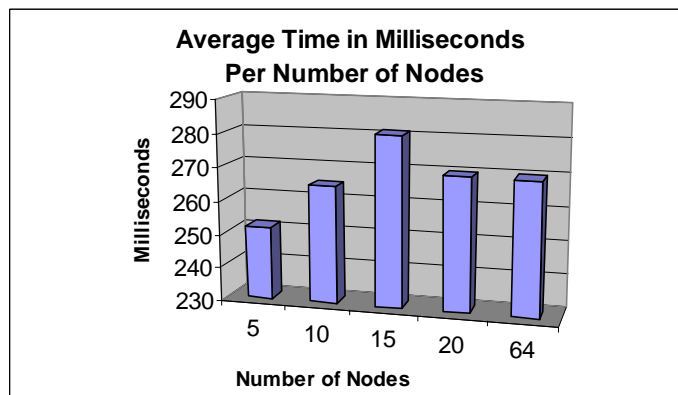


Figure 12

It is important to note that this timing function does not measure the latency time on transferring the graphs from the Prover to the Verifier. The graphs are stored in a static class that the two share. This timer feature is only measuring the time it took to complete the verification process.

### 8.1 ZKP Compared to PKI

We also compared the algorithm to a commonly used authentication algorithm, PKI. The code for the PKI authentication was created by CryptoSys[8]. A trial was used in this experiment. The only

modification that was added was a timer function. Figure 13 is a display of the interface produced by the program.

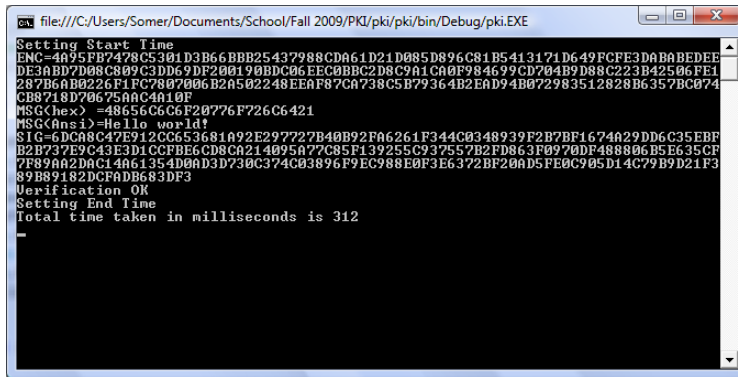


Figure 13: Exchange Keys

Figure 14 shows the run times for seven authentication runs using PKI. The average of these run times is 280.29 milliseconds. By adding together the runtime results for all graph sizes in all trial runs, the average time for our authentication algorithm is 267.43 milliseconds. These results suggest our subgraph isomorphism algorithm actually runs faster than the PKI authentication protocol.

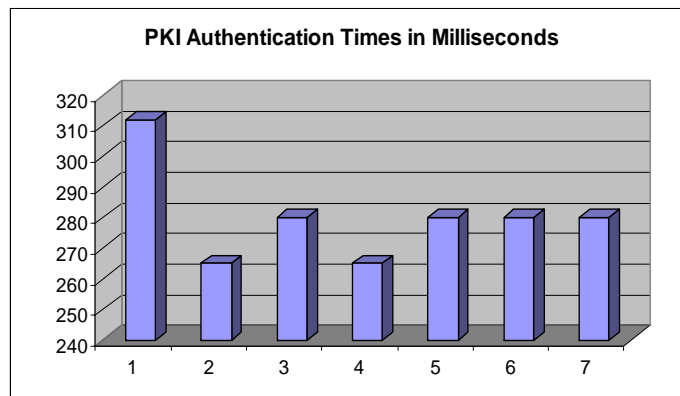


Figure 14

In Figure 15, the averages of the runtimes of each authentication protocol are compared. It can be seen that the zero knowledge protocol proposed by this paper performed better than the PKI protocol.

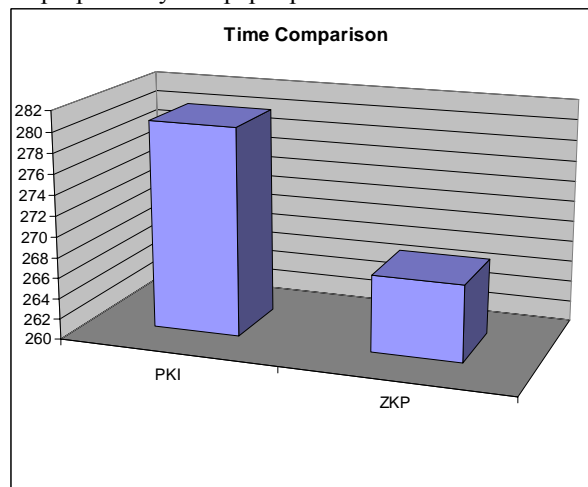


Figure 15

## 8.2 Theories on time variances

As seen in Figure 11, there doesn't seem to be a correlation between the node size of the graphs and the run time it takes to authenticate. One theory for this is the computer that the simulation is running on. There may be other processes that use more of the computers resources at given runs compared to other runs. For example, the laptop that ran the simulations also had Norton 360, which is known to be a resource hog at varying times. Also, there were 2 instances of Visual Studio 2005 processing as the simulation is being run. These may consume resources and cause variance in the runtimes.

Another possible cause of the varying runtimes is the authentication protocol's program itself. The algorithm is designed to create graphs randomly and on the fly. Part of the initialization of the Prover requires random SECRET and MORPH arrays to be created. To populate the MORPH or the SECRET array a random number is chosen between zero and the number of nodes in the ORG graph. One constraint on the MORPH and SECRET arrays is no duplicates. Due to the completely random number generation in the program, it is possible for a number to appear more than once or for a number not to appear for many random number generations. This aspect of the program will take a different amount of time to run each time it is requested. The Prover creates one SECRET and a new isomorphism for every verification round, so the seven creations of the MORPH array throughout verification is another explanation of the variance in runtimes during testing.

## 9. FUTURE WORKS

Zero Knowledge Protocols are designed to work between two parties, the Prover and the Verifier. Through verification rounds the Prover attempts to convince the Verifier he possesses a secret. Over time the Verifier may trust the Prover has a secret and allow the Prover to communication with the Verifier.

The trust the Verifier has for the Prover does not transfer to other nodes of the network. Future work and continuation of this project needs to look into how the Prover earns the trust of the entire network once accepted by the Verifier.

The program used to test the algorithm may be furthered to place the Prover's and Verifier's code on separate nodes of a network. This will allow testing of signal delays and give a better estimation of time to run this protocol.

The subgraph isomorphism problem is a difficult problem without a good known solution. While we tried this solution, it was run for a short time and using very small graphs.

The program randomly populates two matrices which represent the two graphs G1 and G2. In the current form, graph G1 is a five node graph and graph G2 is a three node graph. The matrices are populated the same way as ORG was in the algorithm above using C#'s rand() function and a time element.

Once the graphs are created, the program uses three for loops to iterate through all possible subgraphs of graph G1. The three for loops provide three different nodes of G1 from which to create a subgraph. A temporary matrix is populated with the data from the G1 graph for these three nodes.

Once generated, the program checks if the subgraph is isomorphic to graph G2 by comparing the values in the two matrices. If any values are found to be different between the two matrices the subgraph is considered a failure and the program moves to the next subgraph. If at any point a subgraph matrix matches the values of G2's matrix the program prompts that it was successful. If the program finishes without finding a subgraph the user is prompted that no subgraph of G1 is isomorphic to G2. The program is very specific and works only where graph G2 contains three nodes.

There is room to expand on the proposed solution for the subgraph isomorphism problem. The solution created during this project is very specific to the size of graph G2. More time and dedication to the problem may result in solving for larger graphs and a better thought out, overall solution.

## 10. CONCLUSION

This project produced a new authentication protocol for networks and wireless mobile networks in particular. The subgraph isomorphism problem was used in combination with a zero knowledge protocol to provide the means of verifying a new entity to an existing network.

The wireless mobile network we are working with is extremely time sensitive and data passed through it must remain secure. Without the use of a third party, the Verifier works to ensure the Prover can be trusted. The proposed protocol remains secure as the Verifier is 99% sure the Prover can be trusted before allowing communication to occur. Through development, simulation, and testing the proposed

authentication protocol has shown to be comparable and an improvement to the runtime of the existing PKI authentication protocol. Since graph size didn't have a noticeable effect on runtime, this protocol could be used for much larger, and therefore more secure, graphs in future implementations.

## 11. REFERENCES

- CRYPTOSYS PKI, (2010). <http://www.cryptosys.net/pki/pkicsharp.html>
- GRAPH ISOMORPHISM PROBLEM, (2010). [http://en.wikipedia.org/wiki/Graph\\_isomorphism\\_problem](http://en.wikipedia.org/wiki/Graph_isomorphism_problem).
- GRAPH ISOMORPHISM,(2010). [http://en.wikipedia.org/wiki/Graph\\_isomorphism](http://en.wikipedia.org/wiki/Graph_isomorphism).
- GRIGORIEV AND SHPILRAIN, (2010).“Zero-Knowledge Authentication Schemes from Actions on Graphs, Groups, or Rings”. Preprint submitted to Annals of Pure and Applied Logic.
- HANNU A. ARONSSON.(1999). “Zero Knowledge Protocols and Small Systems”.  
<http://www.tml.tkk.fi/Opinnot/Tik-110.501/1995/zeroknowledge.html>
- [HTTP://EN.WIKIPEDIA.ORG/WIKI/ZERO-KNOWLEDGE\\_PROOF](HTTP://EN.WIKIPEDIA.ORG/WIKI/ZERO-KNOWLEDGE_PROOF) (2010).
- LI LU, JINSONG HAN, YUNHAO LIU, LEI HU, JINPENG HUAI LIONEL NI AND JIAN MA.(2006). “Pseudo Trust: Zero-knowledge Authentication in Anonymous P2Ps”. *IEEE Transactions on Parallel and Distributed Systems*. Vol. 19, No. 10.
- MOBILE NETWORKING, (2010). [http://en.wikipedia.org/w/index.php?title=mobile\\_networking](http://en.wikipedia.org/w/index.php?title=mobile_networking)
- NAMUDURI, KAMESH. (2009). “An Active Trust Model based on Zero Knowledge Proofs for Mobile Networks”.
- PKI, [http://searchsecurity.techtargget.com/sDefinition/0,,sid14\\_gci214299,00.html](http://searchsecurity.techtargget.com/sDefinition/0,,sid14_gci214299,00.html)
- STAWOMIR GRAZONKOSKI, WOJCIECH ZAREMBA, MACIEJ ZAREMBA AND BIL MCDANIEL. (2008).“Extending Wenb Applications with a Lightweight Zero Knowledge Proof Authentication”. CSTST 2008, October 27-31, Cegy-Pontoise, France