Public Key Cryptography Using Graph Theory

Nicholas Drain, John Villalpando, Ph.D.

California Lutheran University

Definitions

Graph

A graph is a collection of points called vertices and connections called edges that signify some relationship between points.

Vertex

A vertex is a point on a graph. It can be assigned a color or a value known as a weight. A vertex can also be connected by an edge to any vertex in the graph.

Edge

An edge is a line between 2 vertices connecting them and signifying some relationship.

Maximum Independent Set

The largest set of vertices where no vertices in the set are connected.

Public Key Cryptography

The practice of encrypting data using a combination of public and private information to keep information secure. It is also sometimes referred to as asymmetric.

XOR

An operator where two bits are compared and the resulting bit is 1 if exactly one of the two bits is one and 0 if both bits are identical. This operator is sometimes also called "exclusive or", or referred to with the symbol $\bigoplus$.

Brute Force Attack

Trying every possible answer until a suitable result is found.

Man in the middle attack

When a transmission is intercepted in transit to the intended recipient.

Objectives

The Primary objective of this study was to create a public key encryption algorithm using graph theory. The secondary goal is to ensure reasonable efficiency and security from attacks by conventional and quantum computers.

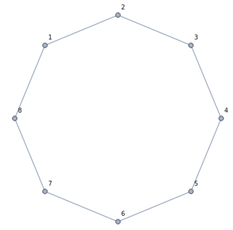
Algorithm

Create some number of graphs that can be defined as G. Each of these graphs should have some number of vertices n. Assign each vertex a random weight expressed as a binary string of pre-determined size. Select exactly half of the vertices to be the key and then connect the graph in a cycle such that no vertex in the key touches another vertex in the key. Add one edge between 2 vertices that aren't in the set. Add a random number of edges to the graph without any of these new edges connecting two vertexes that are in the maximum independent set. Send all graphs to the receiver. The receiver will select one graph at random and find the correct set by brute force. The key generated by solving this will be used in all future communications. The receiver will then send an encoded message using the key that they generate. The original sender will then decode this message using the keys that they stockpiled earlier until one works.

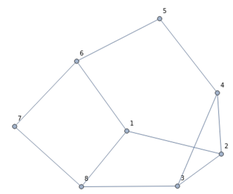
Example

Two individuals, Alice and Bob, wish to communicate with each other securely.

1. Alice creates an 8 vertex graph that is a cycle. she then assigns each vertex both a 4 bit binary string and a number between 1 and the number of vertices.



2. Alice decides a maximum independent set for this graph. In this case it is all odd vertices, however normally it would not follow a pattern.
3. Alice adds at least one edge between 2 vertices not in the set as well as a random number of extra edges between 2 vertices where both of these vertices cant be in the maximum independent set.



4. Steps one through 3 are then repeated until a very large number of graphs are made.
5. These graphs are then sent to Bob
6. Bob selects one graph at random then checks every set of half of the vertices until he finds one that is independent.
7. Bob then uses the $\bigoplus$ operator to combine the binary strings in the set. This generates the string 1001.
8. Bob then takes his message (the string 10101) and uses the $\bigoplus$ operator to combine his message and the key looping the key as needed until the whole message is encoded. since his message is one bit longer than the key the last bit of his message will be $\bigoplus$'ed with the first bit of the key. this generates the string 00110.
9. This message is then sent to Alice who attempts to decode the message using the maximum independent set of each graph. Once one of the sets generates a valid solution Alice is now aware of the key and knows the message.

Applications and Implications

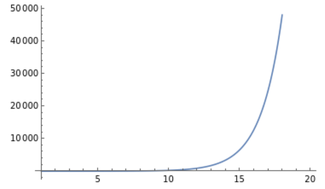
This algorithm would be usable in similar situations to any other encryption algorithm. Specifically though it would be well suited to sending secure transmissions between two individuals that have had no prior communications. Some common instances of this case would be when you shop online or connect to a website. The algorithm also might be secure in post quantum cryptography, however; more research would need to be done to prove this.

Vulnerability Analysis and Computational Complexity

The computational complexity of solving a single graph can be expressed as

$$\frac{n!}{(.5n)!(.5n)!}$$

where n is the number of vertices in the given graph. This means that assuming the worst possible case a brute force attack would need to try that many different permutations to get the answer. The number of possible solutions that an attacker would try in this worst possible case can be modeled by the graph below.



This complexity is more than enough to ensure that using a sufficiently high value of n would make finding a solution time consuming. Since the attacker would also most likely need to solve more than one graph the time spent would be considerably longer than the time it would take to solve a single graph. The other possible attack would be to ignore the graphs and simply attack the key. Since the key length can be set to anything the number of possible keys can be expressed as

$$2^x$$

where x is the length of the key. This means that given a long enough key it would also be impossible to crack it from this side. For context, with a 64 digit long key there would be roughly $1.84e19$ possible keys. This means that with a 64 bit key there are more possible keys then there are grains of sand on earth.

Moving Forward

Moving forward there is work to be done in terms of verifying the exact security involved in the encryption scheme. One possible way in which the algorithm could be attacked is by looking at how many edges are connected to each vertex. This frequency analysis might give potential attackers a way to penetrate the algorithm, however all current tests have been unable to exploit this. Furthermore, it is unknown if this algorithm would be able to remain secure if attacked by a sufficiently strong quantum computer. Further testing could be conducted to verify its security when matched against a computer of this type. Further research could also be done in terms of optimizing the efficiency of using the algorithm.

Acknowledgements

I would like to thank my mentor Dr. Villalpando for his help in developing the algorithm as well as his guidance in the research process. I would also like to acknowledge that Swenson Research Fellowship for funding my research. Furthermore, I would like to thank the OURCS team as well as California Lutheran University for providing the opportunity to conduct this research.