Public-Key Cryptography

Suppose...

...that your friend's birthday is coming up. They've gone off to college on the other side of the country, and so that you don't start to drift apart, you want to send them a cake. They're living in a not-so-great part of town and you want to make sure nobody who isn't your friend opens the vulnerable package on their doorstep.

You have some padlocks and their corresponding keys. Why not try locking the box and sending it off to them then? Then no unwanted party can get to the cake. Well unfortunately neither can the wanted party – your friend. However, your friend has padlocks of their own.

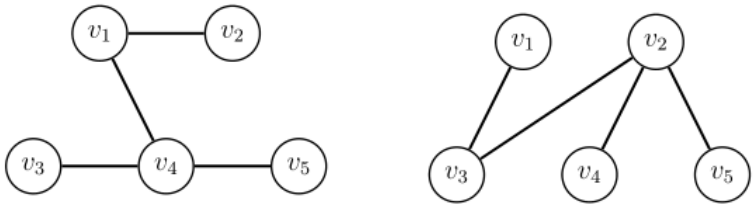
Question 1

What can you and your friend do to ensure that your friend *and only your friend* can open the box with the cake in it?

Graphs

A **graph** is made of **vertices** (the plural of **vertex**) and **edges** between them. We can think of a vertex as just being a single point. An edge joining vertex v_1 to v_2 is represented by the pair (v_1, v_2). Here we say that v_1 is **adjacent** to v_2 or that v_1 and v_2 are **neighbors**. We can visualize a graph by representing its vertices by points and its edges as line segments or curves connecting these points.

Now say we have a set of five vertices, $V = \{v_1, v_2, \ldots, v_5\}$. We can form several graphs with these vertices, a couple of which are shown in Figure 1.



(a) (V, E_1) (b) (V, E_2)

Figure 1: Two graphs with the same set of vertices

Graph (V, E_1) has the edges $E_1 = \{(v_1, v_2), (v_1, v_4), (v_3, v_4), (v_4, v_5)\}$.

Question 2

a Write the edge set for the graph (V, E_2).

b For each vertex in the above graphs, list its neighbors.

c If we insist that no vertex can be adjacent to itself and any two distinct vertices can be joined by at most one edge, how many possible graphs can we make with five vertices? What about with n vertices?

The Plan[1]

The plan is to use a graph to encode a message in a way that only the intended recipient can easily decode it. Here our messages are going to be numbers (this isn't really that much of a limitation. Why not?).

[1]Teaching Fundamentals Concepts of Informatics: 4th International Conference on Informatics in Secondary Schools - Evolution and Perspectives, ISSEP 2010, Zurich, Switzerland, January 13-15, 2010. Proceedings (pp.112-123)

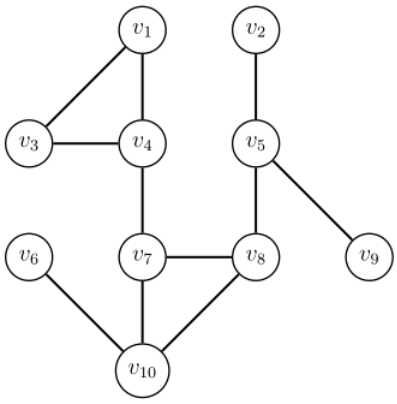
Suppose we want to disguise the number 113 with the graph given in Figure 2a. We start by splitting the number 113 into ten pieces – one per vertex. These pieces can be whatever we want them to be as long as they sum to 113. We assign each piece to a vertex as in Figure 2b. Clearly our original number, 113, is easily reconstructed from these pieces. Just add them up.
Now let's scramble our message. To do this, we add to each vertex the sum of its neighbors. So for example, if a vertex contains the number 6 and its three neighbors have numbers 11, 5, and 17, then we replace the 6 with $6 + 11 + 5 + 17 = 39$. This process of scrambling our message is called **encryption** and the resulting numbered graph is called the **ciphertext**.

While we could just add up all the numbers in our original graph to reconstruct our message, there doesn't seem to be an obvious way to get 113 from the ciphertext. But maybe you can figure one out.

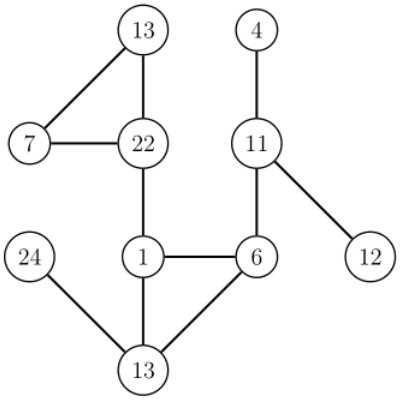
Cryptanalysis

Let's try some **cryptanalysis**: decrypting a ciphertext without knowing any decryption algorithm in advance. You and a partner will be given two copies of the same graph. Choose a number and repeat the process we walked through above. Assign a number to each vertex so that all vertices sum to your chosen number. Then walk through the encryption process and record the ciphertext on the second graph.

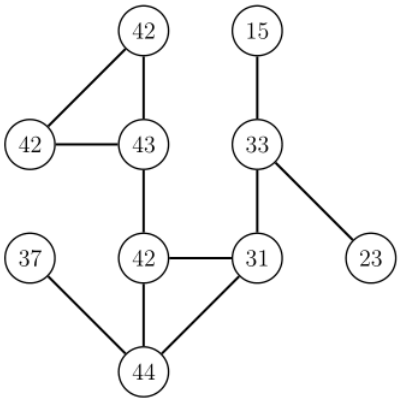
Once every team has encrypted their secret number, we'll exchange ciphertexts. Try to see if you can recover the original message from the ciphertext you're handed.



(a) Starting Graph



(b) Message



(c) Ciphertext

Figure 2: Using a graph to encrypt a message.

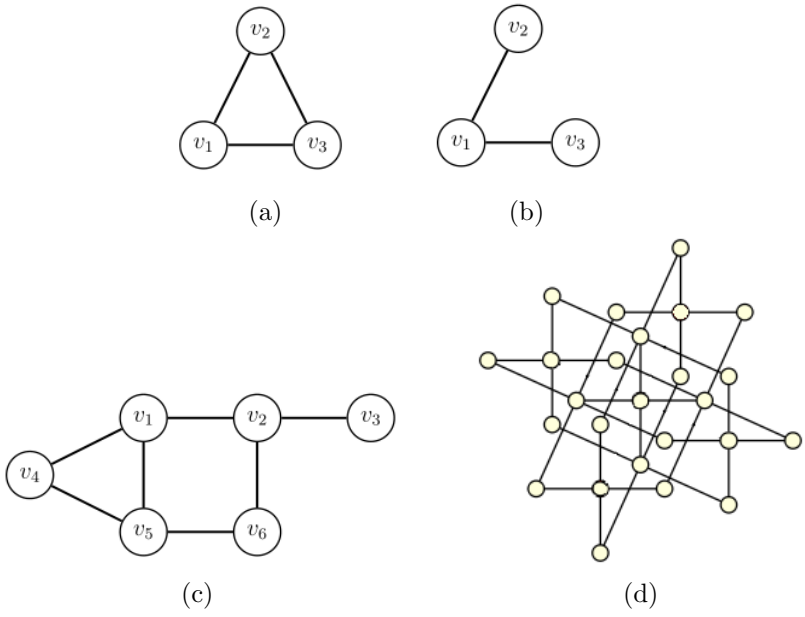
Decryption and Exact Dominating Sets

The graphs you were handed were carefully designed so that the encryption process can be quickly reversed if some secret information is known. Returning to the message and ciphertext in Figure 2, we see that the numbers in vertices v_4, v_5, and v_6 sum to the secret number 113: $37 + 43 + 33 = 113$.

The vertices v_4, v_5, and v_6 form what we'll call an **exact dominating set**. An exact dominating set, D, of a graph (V, E) is a subset of vertices so that each vertex in V that isn't in D is adjacent to *exactly one* vertex in D and no two vertices in D are adjacent to one another.

Question 3

(a) Determine if each graph has an exact dominating set.



(b) Draw a graph with five vertices that has an exact dominating set. What are the possible sizes for an exact dominating set in a graph with five vertices?

(c) Given integers m and n, can you come up with a way to draw a graph with $m + n$ vertices that has an exact dominating set of size m?

(d) Suppose you're given a ciphertext made with a graph that has an exact dominating set. Explain *why* summing the values on the exact dominating set reveals the message.

(e) A **dominating set**, D', of a graph is a subset of its vertices such that each vertex not in D' is adjacent to at least one vertex in D'. Explain how this is different from an *exact* dominating set, maybe by drawing a graph that has a dominating set that isn't an exact dominating set. Why is it important that we use exact dominating sets for our encryption scheme?

Putting it Together

Public-key cryptography is the process of encrypting and decrypting messages in such a way that *anybody* can encrypt a given message but only the recipient can decrypt it. To accomplish this, the recipient, Alice, publishes her *public key* for the world to see – it isn't kept secret. If Bob has Alice's

public key, he can use it to encrypt a message and send it to Alice, who decrypts it using her secret *private key*.

The advantage to this system is that no sensitive information is ever transfered unmasked. Alice doesn't care if an eavesdropper, Eve, sees her public key. If the underlying encryption algorithm is any good, if Eve intercepts Bob's encrypted message to Alice, she won't be able to decipher it if she doesn't have Alice's private key.

Question 4

(a) Consider the graph-based encryption scheme we discussed earlier. How is this a public-key system? What part constitutes the public key and what part is the private key? Is it hard to find the private key if you know only the public key? How can Alice and Bob use this system to communicate in the presence of an eavesdropper, Eve?

(b) Suppose Alice and Bob have access to many different colors of paint. Alice and Bob *publicly* agree on the color yellow. Alice *secretly* chooses the color red and Bob *secretly* chooses the color green. How can Alice and Bob publicly communicate and use the paints they've chosen to establish a shared secret color, even if Eve listens in on their conversation?

Bonus Material: Number Theory

Let a, b, and n be positive integers. We say that a is **congruent to b mod n** if a and b have the same remainder when divided by n. We write this as $a \equiv b \pmod{n}$. Here's a theorem about prime numbers.

Theorem. Let p be a prime number and let a be an integer not divisible by p. Then there exists a number b such that $ab \equiv 1 \pmod{p}$. □

Here's another theorem often attributed to the French mathematician Fermat:

Theorem. (Fermat's little theorem) Let p be a prime number and let a be an integer not divisible by p. Then $a^{p-1} \equiv a \pmod{p}$. □

Question 5

(a) (A bit hard) Prove the first theorem. Hint: think about greatest common divisors and the Euclidean algorithm.

(b) (Harder) Prove Fermat's little theorem. Hint: consider the sets $S = \{1, 2, \ldots, p-1\}$ and $aS = \{a, a \cdot 2, \ldots, a \cdot (p-1)\}$.

(c) Suppose Bob wants to send a number m to Alice. Use the two theorems above to construct a public-key system where Bob can securely send m to Alice.

Here's a good start. Alice and Bob publicly agree on a prime number p. Alice and Bob then secretly choose two numbers each. What properties should these numbers satisfy? How can Bob use his numbers to encrypt his message and how can Alice use her to decipher it?