University of Glasgow

QUANTUM TECHNOLOGY SCHOOL 2021

ACADEMIC PACK

PART 4: CRYPTOGRAPHY

Prof. Alice Miller and Prof. Simon Gay, School of Computing Science, University of Glasgow[1]

Task 3.1: The remote coin flip - Cryptographic protocols

This activity shows how to accomplish a simple, but seemingly impossible, task: making a fair random choice by flipping a coin, between two people who don't necessarily trust each other, and are connected only by a telephone.

Technical terms: Distributed coin-flipping, computer security, cryptography, cryptographic protocol, and-gate, or-gate, combinatorial circuit.

The captains (Alicia and Benito) of two football teams have to decide who gets to be the home team for a championship game. The simplest way would be to flip a coin, but the cities are far apart, and this is not possible. Can they do it over the telephone? Alicia could flip and Benito could call heads or tails. But this won't work because if Benito called heads, Alicia could simply say "sorry it was tails" and Benito would be none the wiser. Even if Alicia is telling the truth, would Benito believe that he had lost?

This is what they decide to do. Working together, they design a circuit made up of and-gates and or-gates (as explained below). In principle they can do this over the phone (how? See later). During the construction process, each has an interest in ensuring that the circuit is complex enough that the other won't be able to cheat. The final circuit is public knowledge.

The rules of and-gates and or-gates are simple. Each "gate" has two inputs and one output (see Figure 2.1A). Each of the inputs can be either 0 or 1, which can be interpreted as false

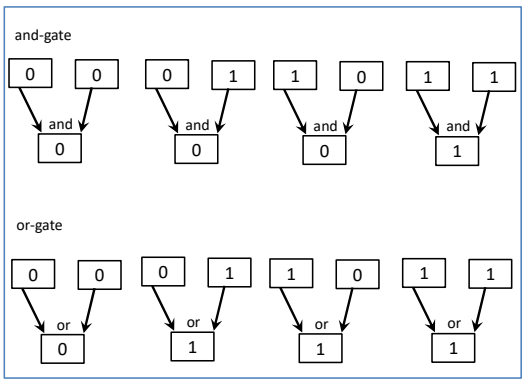
[1] The material in this section was sourced from "Computer Science Unplugged" (classic.csunplugged.org)

and true respectively. The output of an and-gate is 1 (true) only if both inputs are 1 (true), and 0 (false) otherwise. The output of an or-gate is 1(true) only if at least one of its inputs is true (1).

The output of one gate can be connected to the input of another (or several others) to produce a more complicated effect (see Figure 2.1B).

For the remote coin flip, we need even more complicated circuits. The circuit in Figure 2.2A has six inputs and six outputs. Figure 2.2B shows a worked example for one particular set of input values.

A



B

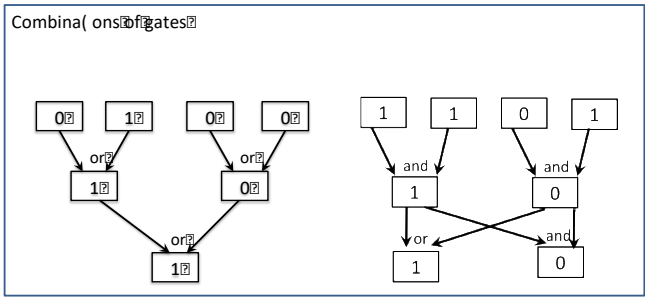
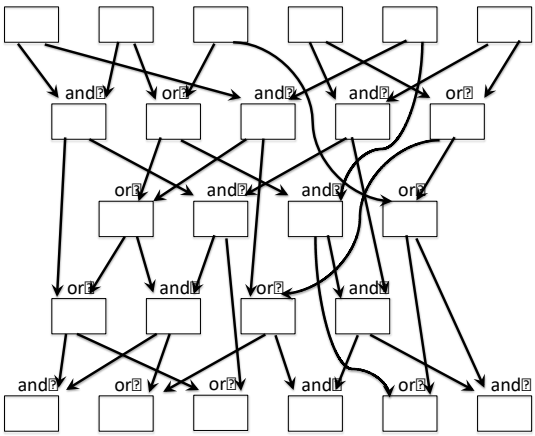


Figure 3.2: A. and-gate and or-gate. B. combinations of gates

A



B

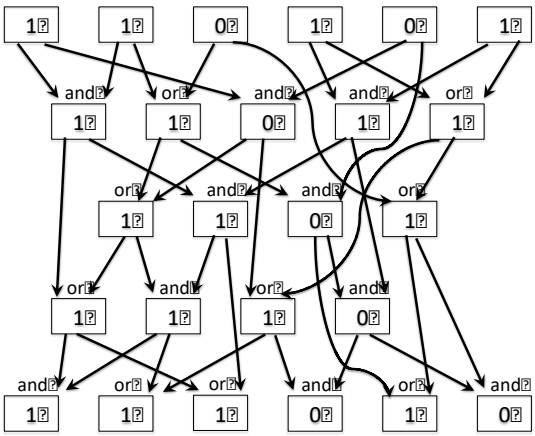


Figure 3.1: A. Complex circuit. B. Filled for example input

The way this circuit can be used to flip a coin by telephone is as follows.

Alicia selects a random input to the circuit, consisting of six binary digits (0s or 1s), which she keeps secret. She puts the digits through the circuit and sends Benito the six bits of output. Once Benito has the output, he must try to guess whether Alicia's input has an even or an odd number of 1s, i.e. he must guess the *parity* of Alicia's input. If the circuit is complex enough then Benito won't be able to work out the answer, and his guess will have to be a random choice. Benito wins if his guess is correct. Alicia wins if Benito guesses incorrectly. Once Benito has guessed, Alicia sends him her secret input so that Benito can confirm that it produces the claimed output.

Exercises

1. Divide your group into two pairs, each containing person A and person B. Take it in turns for A to play the part of Alicia (by putting in random inputs to the circuit shown in Figures 3 to 5 – there are 3 copies per worksheet so that should be enough!), and B to play the part of Benito. How often is the parity guessed correctly?

2. Try to think of how Alicia could cheat (assuming she had a computer). How easy would it be, and how much harder would it be if the circuit took inputs of length 10? 100? 1000?

3. Would it be possible for Benito to cheat? Or even to apply common sense in some cases?

4. Devise your own circuits and think about how you could make it easier for Alicia to cheat – or for Benito to cheat.

5. How would this protocol be implemented in practice? (How would the circuit be created?)

What's it all about?

One aspect of cryptography is how to place controls on information that limit what others can find out, and about establishing trust between people who are geographically separated. Formal rules or "protocols" for cryptographic transactions have been devised to allow such seemingly impossible things as un-forgeable digital signatures and the ability to tell others that you possess a secret without revealing what it is. Flipping a coin over the telephone is a simpler but analogous problem, which also seems, on the face of it, to be impossible.

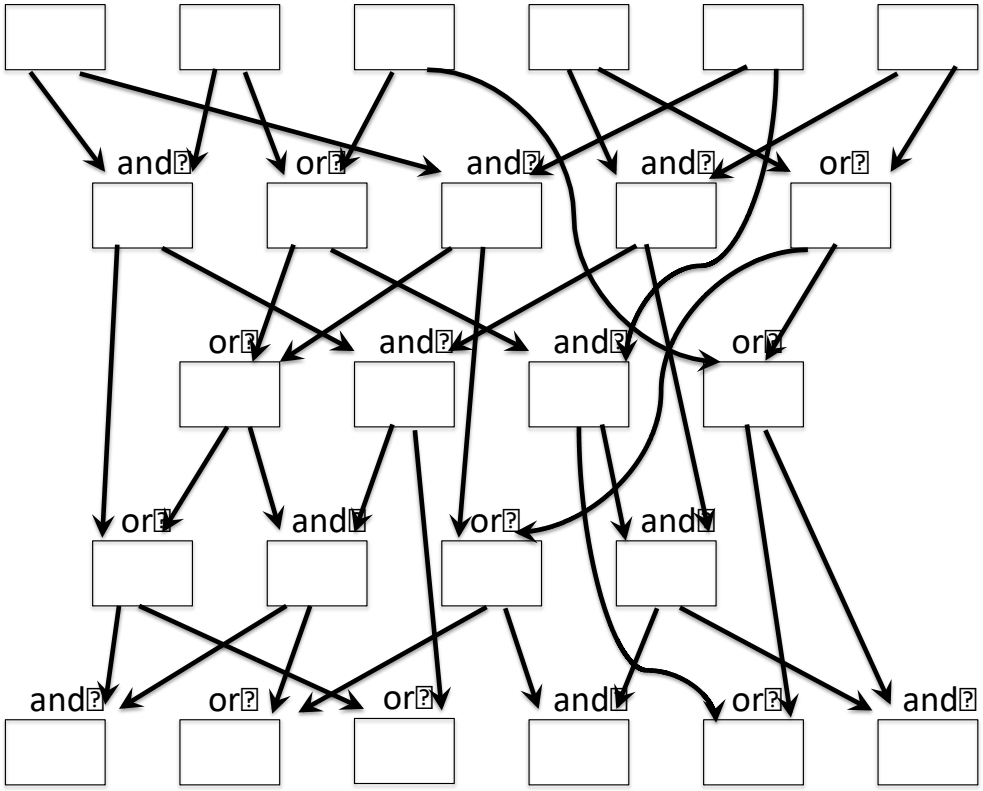


Figure 3.3: Original circuit (copy 1)

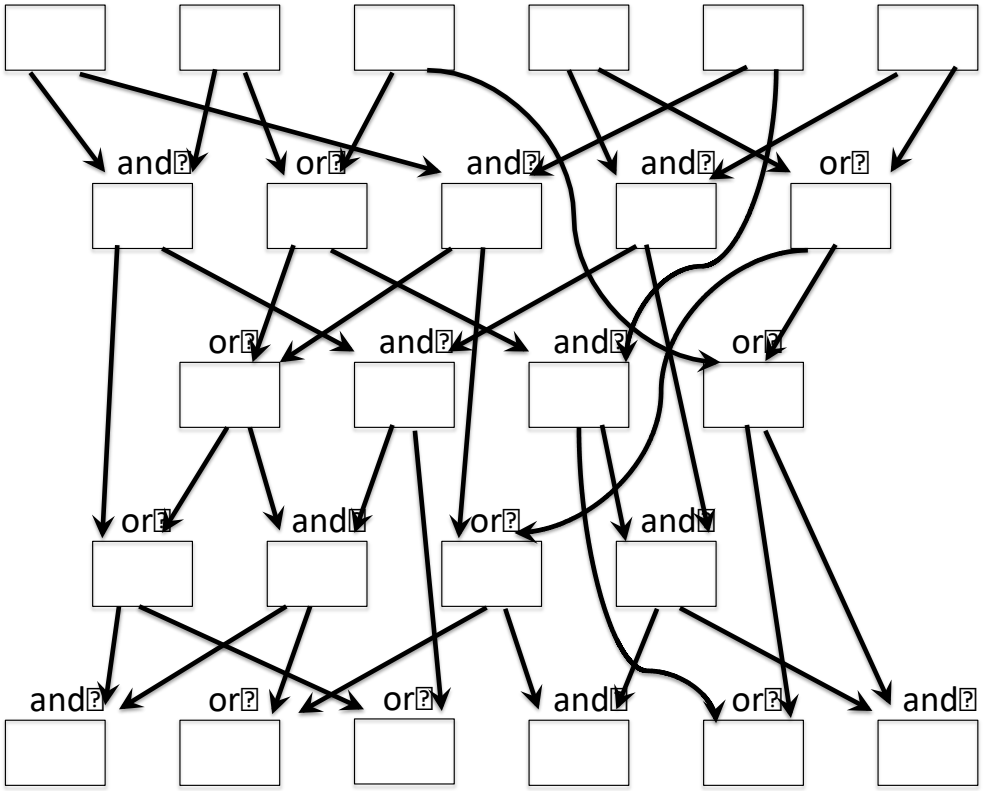


Figure 3.4: Original circuit (copy 2)

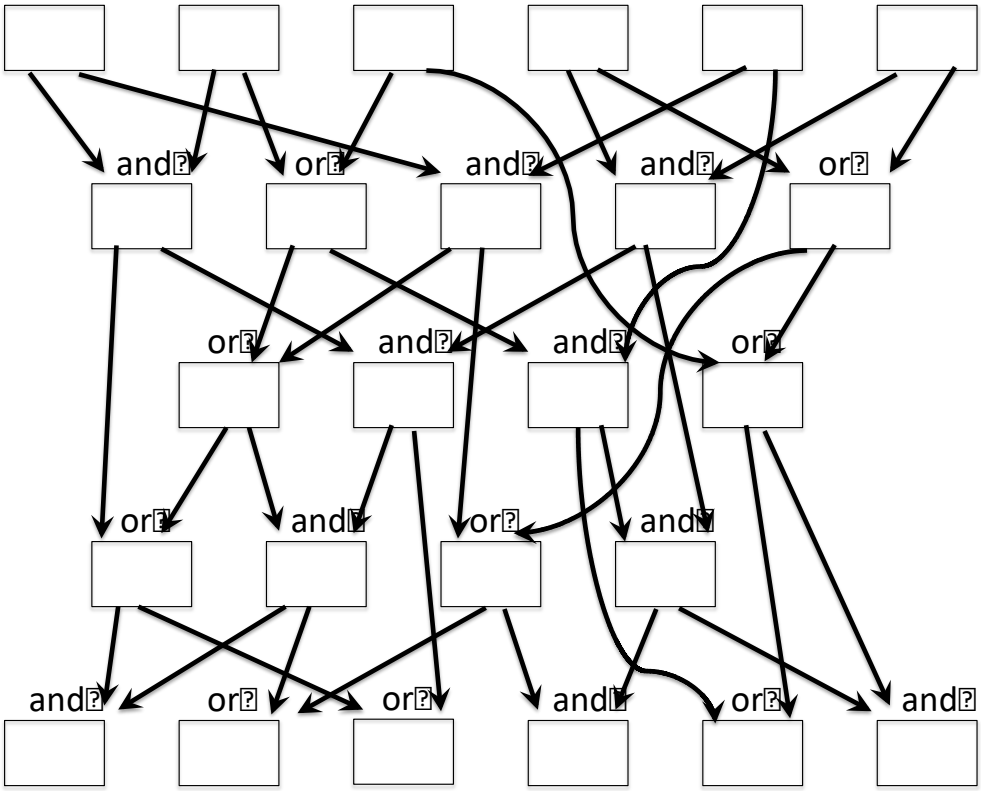


Figure 3.5: Original circuit (copy 3)

University of Glasgow

Task 3.2: Tourist town – Dominating Sets

Many real-life situations can be abstracted into the form of a network or "graph". Networks present many opportunities for the development of algorithms that are practically useful. In this activity, for a given graph, we want to mark some of the junctions (nodes) in such a way that all other nodes are at most one step away from one of the marked ones. The question is, how few marked nodes can we get away with? This turns out to be a surprisingly difficult problem.

In Figure 3.6 there is a map of Tourist Town. The lines are streets and the dots are street corners. On hot days ice-cream vans park at the street corners and sell ice-creams to tourists. We want to place the vans on some of the street corners so that any street corner is at most one step away from a van and no two vans are on adjacent corners. How many vans are needed and where should they be placed?

Exercises

1. Working in pairs, using counters on the intersections to mark an ice-cream van, find a solution to the problem, i.e. ensure that all other intersections are one step from a counter.

2. Compare your solution with the rest of the group. What is the minimum number of counters that were needed?

3. The minimum number of counters is 6. Can you find a solution with this number of counters?

4. This is an example of a problem that is easy to construct but hard to solve (a one-way function). Your tutor will tell you how this graph was constructed so that the solution is already known.

5. Design your own difficult map using this strategy and try them out on each other.

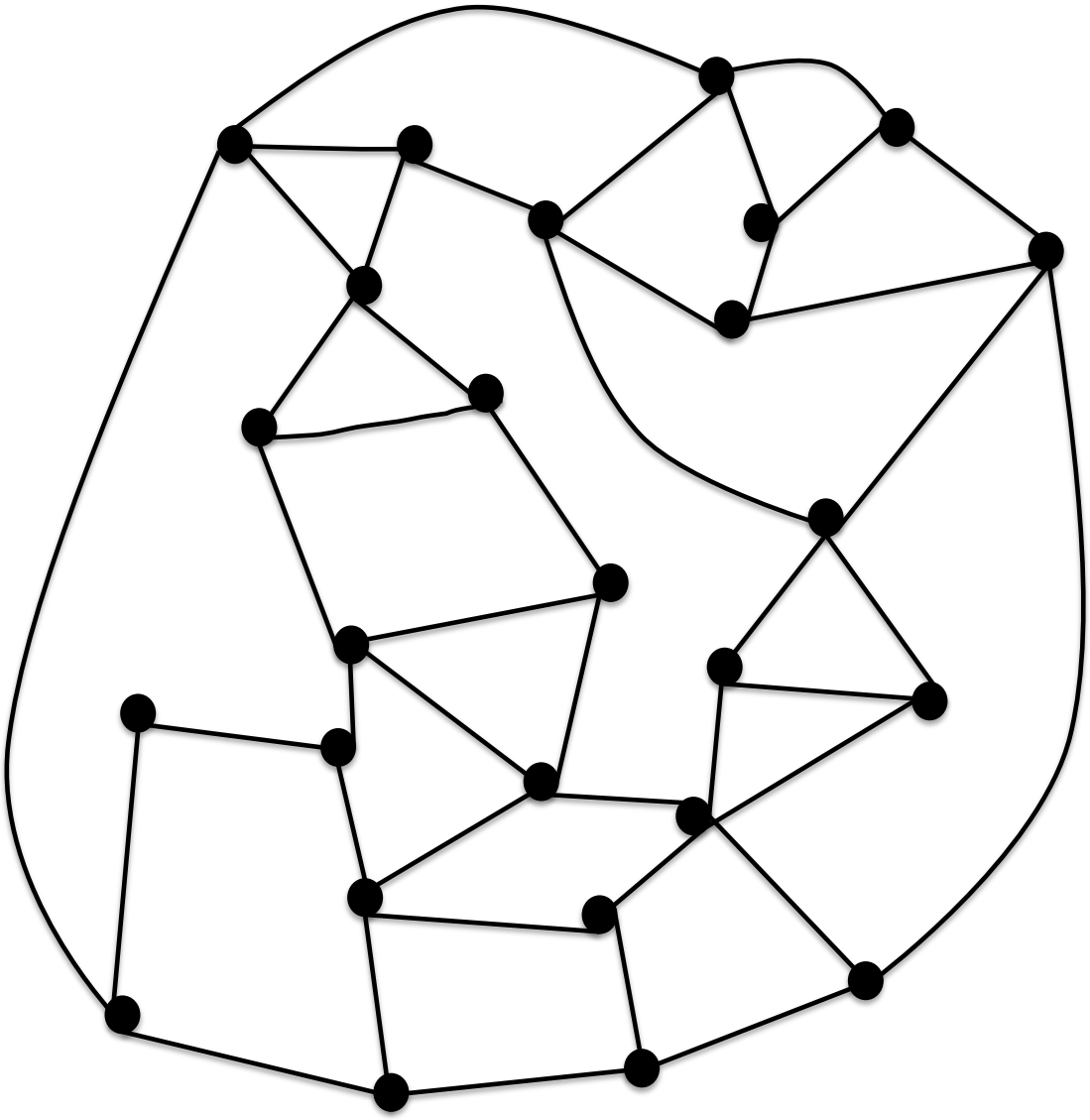


Figure 3.6: Map of Tourist Town

What's it all about?

One of the interesting things about the ice-cream problem is that no-one knows whether there is an algorithm for finding a minimum set of locations that is significantly faster than the brute-force method! The time taken by the brute-force method grows exponentially with the number of intersections (it's called an exponential-time algorithm). A polynomial-time (PT) algorithm is one whose running time grows with the square, cube, or any other power, of the number of intersections. A PT algorithm will always be faster for sufficiently large maps, since an exponentially-growing function outweighs a polynomially-growing one once its argument becomes large enough. (If you work it out, whenever n is bigger than 17 then n^{17} is smaller than 2^n).

The ice-cream van question is officially called the "minimum dominating set" problem. It is one of a large number of problems for which it is not known if a PT algorithm exists, in domains ranging from logic, through jigsaw-like arrangement problems to map colouring, finding optimal routes on maps, and scheduling processes. Astonishingly, all of these problems (which are called NP-complete) have been shown to be equivalent in the sense that if a PT algorithm is found for one of them, it can be converted into a PT algorithm for all the others.

Task 3.3: Public key encryption

Encryption is key to information security. And the key to modern encryption is that using only public information, a sender can lock up their message in such a way that it can only be unlocked (privately) by the intended recipient.

Imagine that everyone buys a padlock and writes their name on it, and puts them all on a table for others to use. They keep the key of course – the padlocks are the kind where you just click them shut. If I want to send you a secure message, I put it in a box, pick up your padlock, lock the box and send it to you. Even if it falls into the wrong hands, no one else can unlock it. With this scheme there is no need for any prior communication to arrange secret codes.

This activity shows how this can be done digitally. In the digital world, instead of picking up your padlock and using it, I copy it and use the copy, leaving the original lock on the table. If I were to make a copy of a physical padlock, I could only do so by taking it apart. In doing so I would inevitably see how it worked. But in the digital world we can copy locks without being able to discover the key.

Amy is planning to send Bill a secret message. Normally we might think of secret messages as a sentence or paragraph, but in the following exercise Amy will just send one character. In fact she will just send one number (e.g. 5, 37 or 101) that represents a character.

We will see how to embed Amy's number in an encrypted message using Bill's public lock so that if anyone intercepts it, they will not be able to decode it. Only Bill can do that, because only he has the key to the lock.

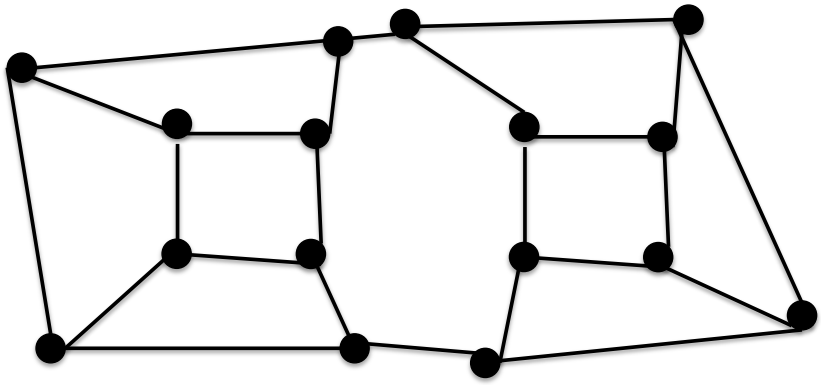
We will lock up messages using maps (depicted as graphs, as used in Part 2). Each map has a public version – the lock – and a private version – the key.

Shown in Figure 3.7 (A) is Bill's public map. It's not a secret. He gives it to anyone who wants to send him a message. Fig 3.7 (B) shows Bill's private map. It's the same as his public map

except some nodes are marked as special by enlarging them. He keeps this version of the map secret.

Suppose that Amy wants to send the message 66 to Bill. Your tutor will explain how she does this, and how Bill retrieves the original message, using the maps in Figures 3.8 and 3.9.

A



B

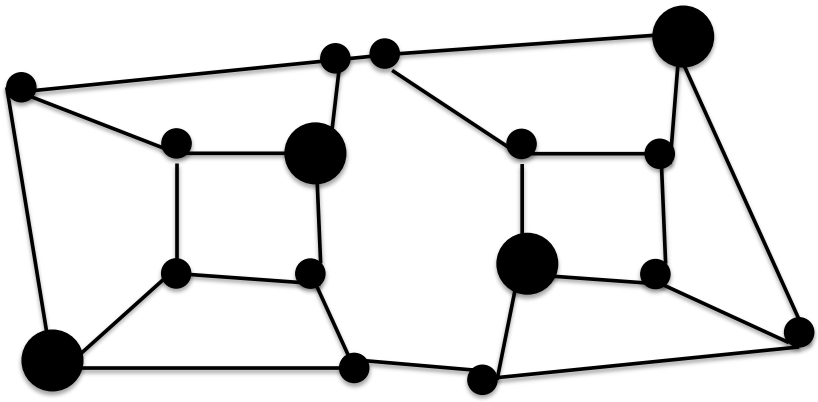


Figure 3.7: A. Bill's public map B. Bill's private map

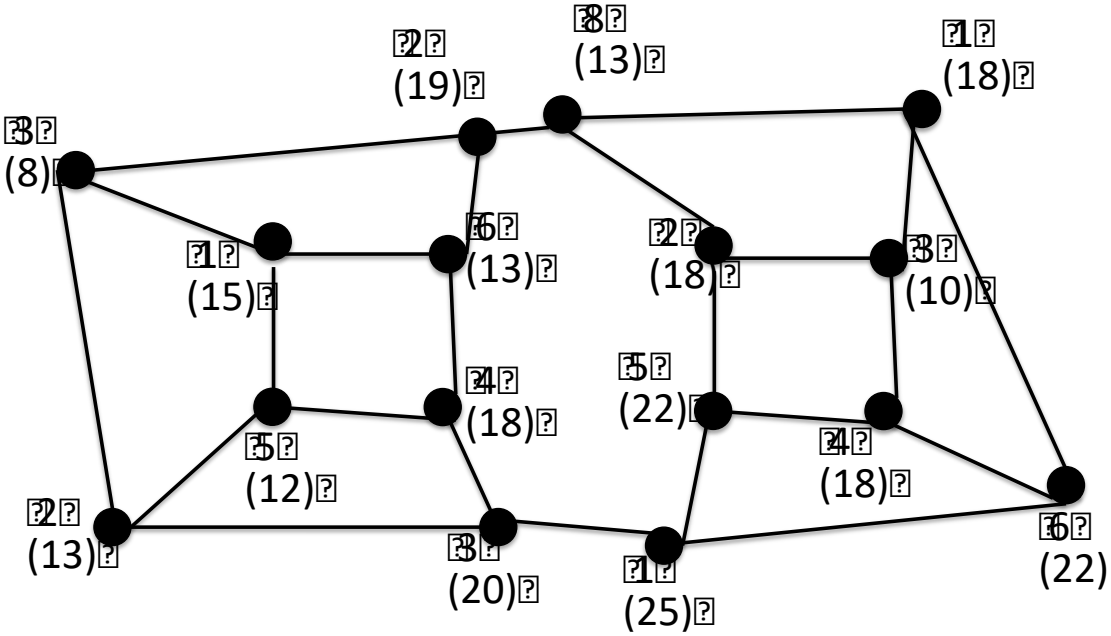


Figure 3.8: Amy's calculations, using Bill's public map

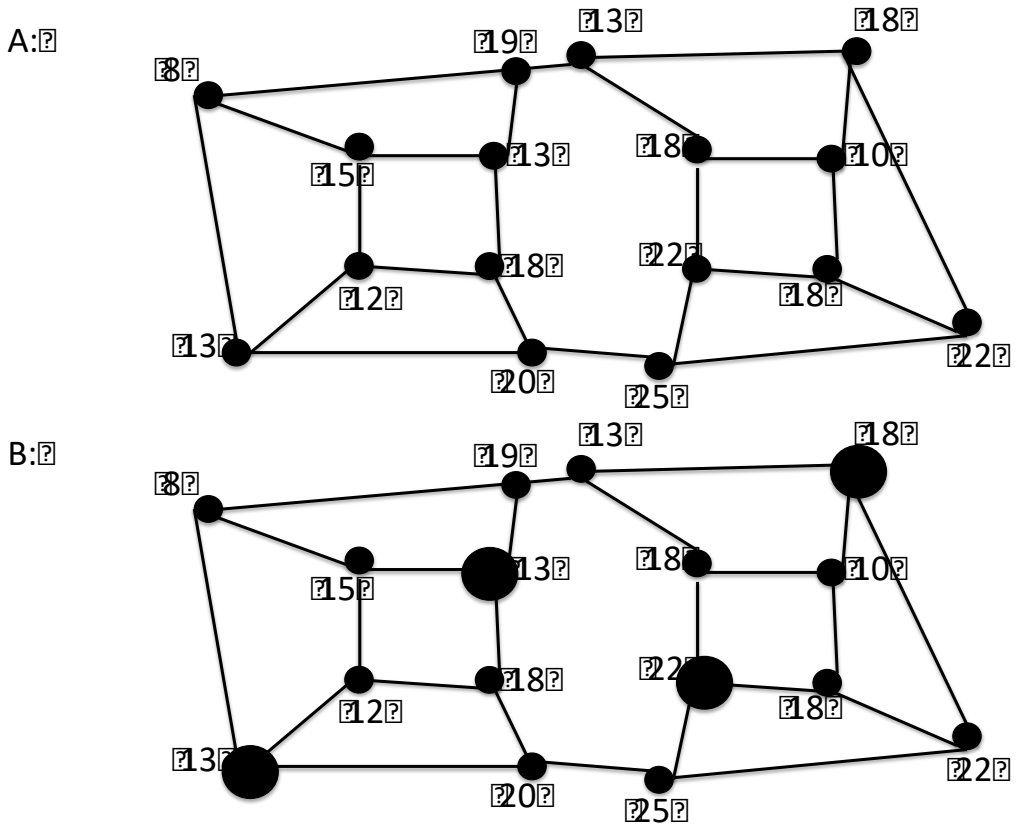


Figure 3.9: A. What Amy sends to Bill B. information transcribed onto Bill's private map

Exercises

1. Divide your group into two groups A and B (Amy and Bill). Group A should choose a message and send it to B using the method described above, using the public graph shown in Figure 3.10. Group B should try to decode it. This will be difficult without the private key!

2. The private key is Figure 3.7. With this, group B can now decode the message.

   Determining a private key for any map is a special case of the Tourist Town example seen in Part 2. In this case the large nodes are chosen such that every other (small) node is adjacent to *exactly one* large node, and *no two large nodes are adjacent*. We call this a *special dominating set*. As for a dominating set, finding a special dominating set for a given graph a very difficult task. However, if we start from a special dominating set to create the graph in the first place, it is much simpler! Your tutor will now show you how to do this.

3. Design your own graphs for which you know a special dominating set, and use them to send messages to each other.

4. How might you program this into a computer? You can find an example at https://blairarchibald.github.io/csunplugged-public-key-encryption/app/src/

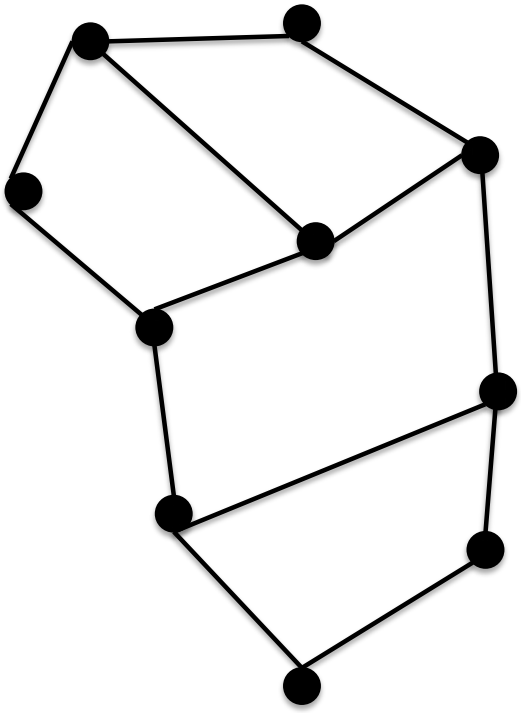


Figure 3.10: Graph for exercise (copy 1)

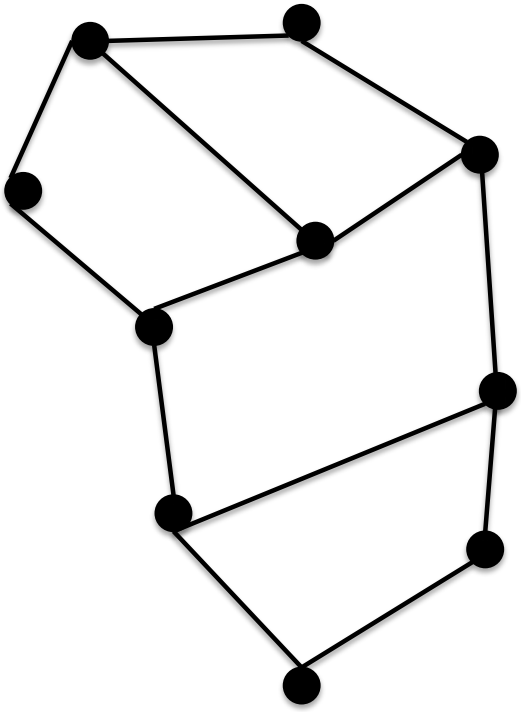


Figure 3.11: Graph for exercise (copy 2)

What's it all about?

It's clear why you might want to send secret messages over computer networks that no-one but the intended recipient could decode, no matter how clever they were or how hard they tried. And, of course, there are all sorts of ways in which this can be done if the sender and receiver share a secret code (like a cypher). But the clever part of public-key encryption is that Amy can send Bill a secure message without any secret prior arrangement, just by picking up his lock from a public place like a web page.

However…

Although the scheme illustrated in this activity is very similar to an industrial-strength public-key encryption system, it is not in fact a secure one – even if a large map is used. Although there is no known way of finding the minimal special dominating set, there happens to be a completely different way of solving it.

The processes involved in real public-key cryptosystems are virtually identical to what we have seen, except that the techniques they use for encoding are different – and really are infeasible to do by hand. The original public-key method, and still one of the most secure, is based on the difficulty of factoring large numbers.