Activity 18

Kid krypto—*Public-key encryption*

Age group	Junior high and up.
Abilities assumed	This is the most technically challenging activity in the book. While rewarding, it requires careful work and sustained concentration to complete successfully. Children should already have studied the example of one-way functions in Activity 14, Tourist Town, and it is helpful if they have completed the other activities in this section (Activity 16, Sharing Secrets, and Activity 17, the Peruvian coin flip). The activity also uses ideas covered in Activity 1, Count the dots, and Activity 5, Twenty guesses.
Time	About 30 minutes.
Size of group	Requires at least two people, can be done with a whole class.

Focus

Puzzle solving.

Secret codes.

Summary

Encryption is the 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*, of course) by the intended recipient.

It is as though everyone buys a padlock, writes their name on it, and puts them all on the same 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. And 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 arrange for people to copy locks without being able to discover the key!

Sounds impossible? Read on.

Technical terms

Public-key cryptosystems, encryption, decryption, NP-complete problems.

Materials

The children are divided into groups of about four, and within these groups they form two subgroups. Each subgroup is given a copy of the two maps on page 193. Thus for each group of children you will need:

two copies of the blackline master on page 193.

You will also need:

an overhead projector transparency of page 194, and

a way to annotate the diagram.

What to do

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 send just one character — in fact, she will send one number that represents a character. Although this might seem like a simplistic message, bear in mind that she could send a whole string of such "messages" to make up a sentence, and in practice the work would be done by a computer. And sometimes even small messages are important — one of the most celebrated messages in history, carried by Paul Revere, had only two possible values.

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.

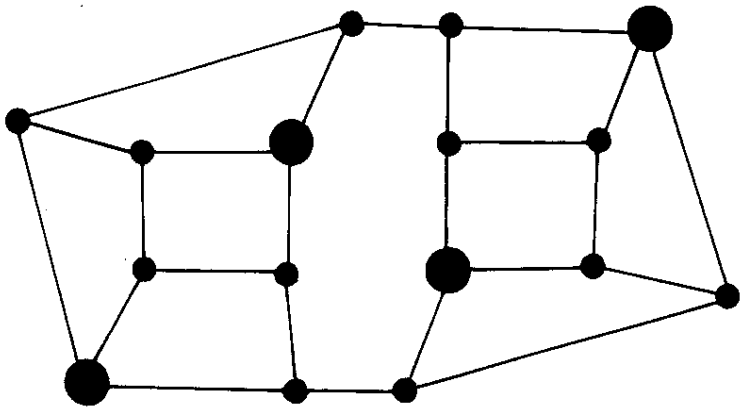


Figure 18.1: Bill's private map

We will lock up messages using *maps*. Not Treasure Island maps, where X marks the spot, but street maps like the one of Tourist Town on page 149, where the lines are streets and the dots are street corners. Each map has a public version—the lock—and a private version—the key.

Shown on page 194 is Bill's public map. It's not secret: Bill puts it on the table (or a web page) for everyone to see, or (equivalently) gives it to anyone who might want to send him a message. Amy has a copy; so has everyone else. Figure 18.1 shows Bill's private map. It's the same as his public map, except that some of the street corners are marked as special by enlarging them. He keeps this version of the map secret.

This activity is best done as a class, at least to begin with, because it involves a fair amount of work. Although not difficult, this must be done accurately, for errors will cause a lot of trouble. It is important that the children realize how surprising it is that this kind of encryption can be done at all—it seems impossible (doesn't it?)—because they will need this motivation to see them through the effort required. One point that we have found highly motivating for school children is that using this method they can pass secret notes in class, and even if their teacher knows how the note was encrypted, the teacher won't be able to decode it.

1. Put Bill's public map (page 194) on the overhead projector. Decide which number Amy is going to send. Now place random numbers on each intersection on the map, so that the random numbers add up to the number that Amy wishes to send. Figure 18.2 gives an example of such numbers as the upper (non-parenthesised) number beside each intersection. Here, Amy has chosen to send the number 66, so all the unbracketed numbers add up to 66. If necessary, you can use negative numbers to get the total down to the desired value.

2. Now Amy must calculate what to send to Bill. If she sent the map with the numbers on, that would be no good, because if it fell into the wrong hands anybody could add them up and get the message.

   Instead, choose any intersection, look at it and its three neighbors—four intersections in

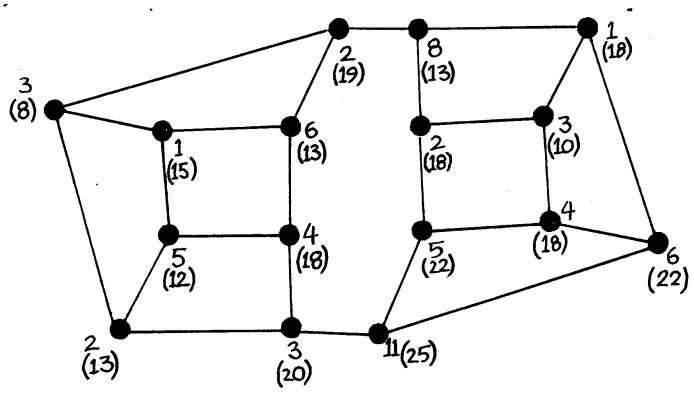


Figure 18.2: Amy's calculations, using Bill's public map

all—and total the numbers on them. Write this number at the intersection in parentheses or using a different color of pen. For example, the rightmost intersection in Figure 18.2 is connected to three others, labeled 1, 4, 11, and is itself labeled 6. Thus it has a total of 22. Now repeat this for all the other intersections in the map. This should give you the number in parentheses in Figure 18.2.

3. Amy will send to Bill his map, with only the parenthesised numbers on it.

   Erase the original numbers and the counts, leaving only the numbers that Amy sends; or write out a new map with just those numbers on it. See if any of the children can find a way to tell from this what the original message was. They won't be able to.

4. Only someone with Bill's private key can decode the message to find the message that Amy originally wanted to send. On the coded message mark the special enlarged nodes in Bill's private map (Figure 18.1).

   To decode the message, Bill looks at just the secret marked intersections and adds up the numbers on them. In the example, these intersections are labeled 13, 13, 22, 18, which add up to 66, Amy's original message.

5. How does it work? Well, the map is a special one. Suppose Bill were to choose one of the marked intersections and draw around the intersections one street distant from it, and repeat the procedure for each marked intersection. This would partition the map into non-overlapping pieces, as illustrated in Figure 18.3. Show these pieces to the children by drawing the boundaries on the map. The group of intersections in each partition is exactly the ones summed to give the transmitted numbers for the marked intersections, so the sum of the four transmitted numbers on those intersections will be the sum of all the original numbers in the original map; that it, it will be the original message!

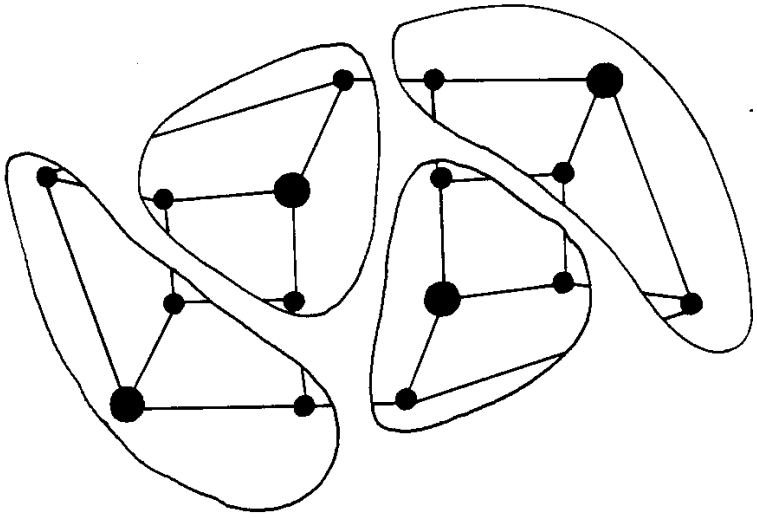


Figure 18.3: The regions that comprise Bill's private map

Phew! It seems a lot of work to send one letter. And it *is* a lot of work to send one letter—encryption is not an easy thing to do. But look at what has been accomplished: complete secrecy using a public key, with no need for any prior arrangement between the participants. You could publish your key on a noticeboard and *anyone* could send you a secret message, yet no-one could decrypt it without the private key. And in real life all the calculation will be done by a software package that you acquire, so it's only a computer that has to work hard.

Perhaps your class would like to know that they have joined the very select group of people who have actually worked through a public-key encryption example by hand—practising computer scientists would consider this to be an almost impossible task and virtually no-one has ever done it!

Now, what about eavesdropping? Bill's map is like the ones in the Tourist Town activity (Activity 14), where the marked intersections are a minimal way of placing ice-cream vans to serve all street corners without anyone having to walk more than one block. We saw in Tourist Town that it's easy for Bill to make up such a map by starting with the pieces shown in Figure 18.3, and it's very hard for anyone else to find the minimal way to place ice-cream vans except by the brute-force method. The brute-force method is to try every possible configuration with one van, then every configuration with two vans, and so on until you hit upon a solution. No-one knows whether there is a better method for a general map—and you can bet that lots of people have tried to find one!

Providing Bill starts with a complicated enough map with, say, fifty or a hundred intersections, it seems like no-one could ever crack the code—even the cleverest mathematicians have tried hard and failed. (But there is a caveat: see below under *What's it all about?*)

6. Having been through one example with the whole class, divide the children into groups of, say, four. Give each pair of each group the public map on the blackline master on

page 193. Each pair should choose a "message" (any integer), encode it with the public key, and give the resulting map to the other group. The other group can try to decode it, but they are unlikely to be successful until they are given (or work out!) the private map. Then give out the private map and see if they can now decode it correctly.

7. Now each pair can design their own map, keeping the private version secret and giving the public version to the other pair—or indeed "publishing" it on the classroom board. The principle for designing maps is just the same as was discussed in the Tourist Town activity, and extra streets can be added to disguise the solution. Just be careful not to add extra streets into any of the "special" points. That would create an intersection from which *two* ice-cream vans could be reached in one hop, which is all right for the tourist town situation but would cause havoc when encrypting. That is because the special points no longer decompose the map into *non-overlapping* pieces, as illustrated in Figure 18.3, and this is essential for the trick to work.

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. 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.

Secrecy is only one side of cryptography. Another is *authentication*: When Amy receives a message from Bill, how does she know that it really comes from him and not from some imposter? Suppose she receives electronic mail that says, "Darling, I'm stuck here without any money. Please put $100 in my bank account, number 0241-45-784329 – love, Bill." How can she know whether it really comes from Bill? Some public-key cryptosystems can be used for this, too. Just as Amy sends Bill a secret message by encoding it with his public key, he can send her a message *that only he could have generated* by encoding it with his *private* key. If Amy can decode it with Bill's public key, then it must have come from him. Of course, anyone else could decode it too, since the key is public, but if the message is for her eyes only, Bill can then encode it a second time with Amy's public key. This dual encoding provides both secrecy and authentication with the same basic scheme of public and private keys.

Now is the time to admit that while 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 quite a large map is used.

The reason is that although there is no known way of finding the minimal way to place ice-cream vans on an arbitrary map, and so the scheme is indeed secure from this point of view, there happens to be a completely different way of attacking it. The idea is unlikely to occur to schoolchildren, at least up to high school level, but you should at least know that it exists. You might say that the scheme we have been looking at is school-child secure, but not mathematician-secure. Please ignore the next paragraph if you are not mathematically inclined!

Number the intersections on the map 1, 2, 3, ... Denote the original numbers that are assigned to intersections by b_1, b_2, b_3, ..., and the numbers that are actually transmitted by t_1, t_2, t_3, ... Suppose that intersection 1 is connected to intersections 2, 3, and 4. Then the number that is transmitted for that intersection is

$$t_1 = b_1 + b_2 + b_3 + b_4.$$

Of course, there are similar equations for every other intersection—in fact, there are the same number of equations as there are unknowns b_1, b_2, b_3, ... An eavesdropper knows the public map and the numbers t_1, t_2, t_3, ... that are transmitted, and can therefore write down the equations and solve them with an equation-solving computer program. Once the original numbers have been obtained, the message is just their sum—there is actually no need ever to discover the decryption map. The computational effort required to solve the equations directly using Gaussian elimination is proportional to the cube of the number of equations, but because these equations are sparse ones—most of the coefficients are zero—even more efficient techniques exist. Contrast this with the exponential computational effort that, as far as anyone knows, is the best one can do to come up with the decryption map.

We hope you don't feel cheated! In fact, 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.

What are the factors of the 100-digit number 9,412,343,607,359,262,946,971,172,136,294, 514,357,528,981,378,983,082,541,347,532,211,942,640,121,301,590,698,634,089,611,468, 911,681? Don't spend too long! They are 86,759,222,313,428,390,812,218,077,095,850,708,048, 977 and 108,488,104,853,637,470,612,961,399,842,972,948,409,834,611,525,790,577,216,753. There are no other factors: these two numbers are prime. Finding them is quite a job: in fact, it's a several-month project for a supercomputer.

Now in a real public-key cryptosystem, Bill might use the 100-digit number as his public key, and the two factors as the private key. It would not be too difficult to come up with such keys: all you need is a way of calculating large prime numbers. Find two prime numbers that are big enough (that's not hard to do), multiply them together, and—hey presto, there's your public key. Multiplying huge numbers together is no big deal for a computer. Given the public key, no-one can find your private key, unless they have access to several months of supercomputer time. And if you're worried that they might, use 200-digit primes instead of 100-digit ones—that'll slow them down for years! In practice, people use 512-bit keys, which is about 155 decimal digits.

We still haven't given a way to encode a message using a prime-number based public key in such a way that it can't be decoded without the private key. In order to do this, life is not quite as simple as we made out above. It's not the two prime numbers that are used as the private key and their product as the public key, instead it's numbers derived from them. But the effect is the same: you can crack the code by factoring the number. Anyway, it's not difficult to overcome these difficulties and make the scheme into a proper encryption and decryption algorithm, but let's not go into that here. This activity has already been enough work!

How secure is the system based on prime numbers? Well, factoring large numbers is a problem that has claimed the attention of the world's greatest mathematicians for several cen-

turies, and while methods have been discovered that are significantly better than the brute-force method of trying all possible factors, no-one has come up with a really fast (that is, polynomial-time) algorithm. (No-one has proved that such an algorithm is impossible, either.) Thus the scheme appears to be not just school-child secure, but also mathematician-secure. But beware: we must be careful. Just as there turned out to be a way of cracking Bill's code without solving the Tourist Town problem, there may be a way of cracking the prime-number codes without actually factoring large numbers. People have checked carefully for this, and it seems OK.

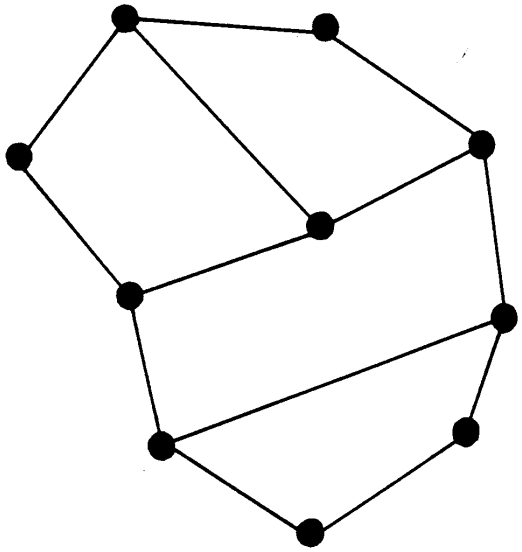
Another worry is that if there are just a few possible messages, an interloper could encrypt each of them in turn using the public key, and compare the actual message with all the possibilities. Amy's method avoids this because there are many ways of encrypting the same message, depending on what numbers were chosen to add up to the code value. In practice, cryptographic systems are designed so that there are just too many possible messages to even begin to try them all out, even with the help of a very fast computer.

It is not known whether a fast method for solving the prime factorization problem exists. No one has managed to devise one, but also it has not been proven that a fast method is impossible. If a fast algorithm for solving this problem is found, then many currently used cryptographic systems will become insecure. In Part IV we discussed *NP-complete* problems, which stand or fall together: if one of them is efficiently solvable then they all must be. Since so much (unsuccessful) effort has been put into finding fast algorithms for these problems, they would seem like excellent candidates for use in designing secure cryptosystems. Alas, there are difficulties with this plan, and so far the designers of cryptosystems have been forced to rely on problems (such as prime factorization) that might in fact be easier to solve than the NP-complete problems—maybe a lot easier. The answers to the questions raised by all this are worth many millions of dollars to industry and are regarded as vital to national security. Cryptography is now a very active area of research in computer science.

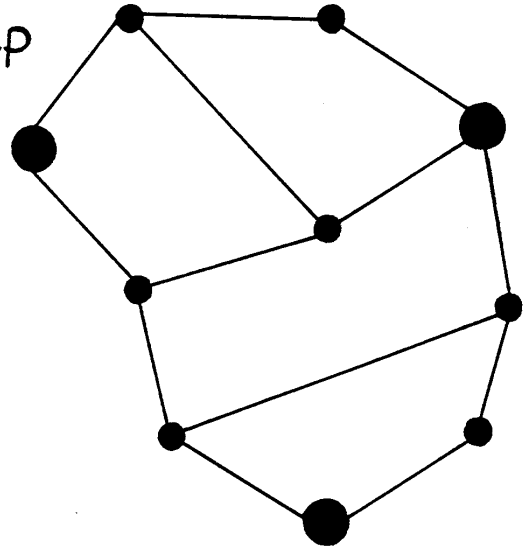
Further reading

Harel's book *Algorithmics* discusses public-key cryptography; it explains how to use large prime numbers to create a secure public-key system. The standard computer science text on cryptography is *Cryptography and data security* by Dorothy Denning, while a more practical book is *Applied cryptography* by Bruce Schneier. Dewdney's *Turing Omnibus* describes another system for performing public key cryptography.

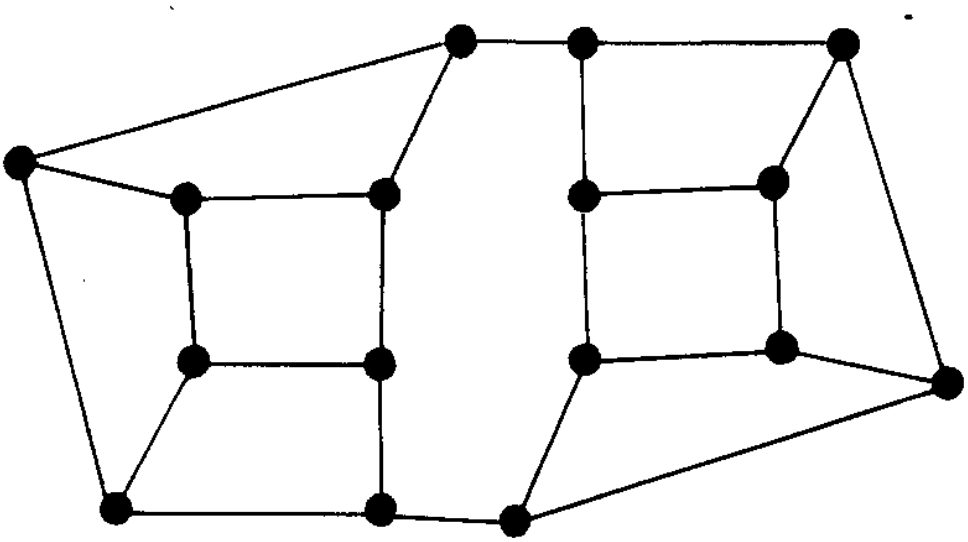
public Map



private Map



Instructions: *Use these maps as described in the text to encrypt and decrypt messages.*



Instructions: *Put this blackline master on an overhead projector transparency and use it to demonstrate the encoding of a message.*