GEOMETRY

Lecture 5

MODERN CRYPTOGRAPHY

by

PROFESSOR HAROLD THIMBLEBY
Gresham Professor of Geometry

28 February 2002

Explaining Cryptographic Systems

Tim Bell
University of Canterbury
Christchurch
New Zealand

Harold Thimbleby
Gresham College
& University College London
London, UK

Mike Fellows
University of Victoria, British
Columbia
Canada

Ian Witten
University of Waikato
Hamilton
New Zealand

Neil Koblitz
University of Washington
Seattle
USA

Matthew Powell
University of Canterbury
Christchurch
New Zealand

Abstract

Modern cryptography can achieve levels of security and authentication that non-specialists find literally incredible. Techniques including information-hiding protocols, zero-knowledge proofs and public key cryptosystems can be used to support applications like digital signatures, digital cash, on-line poker and secure voting in ways that are provably secure — far more secure than the traditional systems they replace. This paper describes simple versions of such applications that have been used to give school-children and the general public a broad understanding of what can be achieved, and how.

The material has been extensively and successfully used by the authors in schools, science festivals and with undergraduates, and even postgraduate specialists.

Introduction

Security and privacy are pressing social issues in an era when much commerce is conducted electronically, and personal information is stored on computers and transmitted over computer networks. Modern cryptographic systems can implement extremely high levels of security, but their capabilities are not widely appreciated by the general public. For example, many people use debit cards to pay for goods, where money is transferred directly from their bank account to the store's. In the process the bank finds out where the purchase is being made, and could build up a profile of the person's shopping habits. Despite this loss of privacy, debit cards are widely accepted. Most people are quite unaware that cryptographic protocols exist that enable the transaction to be carried out reliably without the bank being able to identify to whom the money is going! This seems incredible — literally unbelievable — to people who have never encountered public key cryptosystems and information hiding protocols.

If more people knew about such things, they would lobby for their adoption to better protect privacy in everyday transactions. Moreover, it would cultivate a higher level of trust for systems that use sophisticated protocols to protect information. Just as an understanding of biology goes a long way towards making informed decisions on environmental issues, understanding the technical issues involved in cryptography enables informed decisions on privacy issues. The activities described in this paper are intended to take some of the science fiction out of people's understanding of computer security. The upcoming generation of computer users deserves a clear view of the technical issues that underpin the myriad computerized systems that permeate our lives.

Most people's knowledge of information security dates from their school days, when they may have been introduced to a Caesar cipher (replacing one letter with another further along in the alphabet), or another substitution cipher that maps the alphabet to special symbols, and perhaps other private key systems such as the one-time pad (a long key which is used just once — it could conveniently be the text of a book). All of these schemes have the problem of getting every trusted person — and nobody else! — to know the keys. Before the internet and world-wide communication for business purposes, perhaps it was possible to distribute keys by sending spies or diplomats, but when you want to communicate securely with new business customers on the other side of the world, who by definition you never knew before, alternatives to conventional

secure key distribution must be found. Most people have no idea about the advanced cryptographic techniques that are now available to solve these problems and others such as authentication (how do you know who your new customer is?).

Unfortunately cryptographic experts often regard the new techniques as far too abstruse for school-children to understand. We disagree. The goal of our work is to make advanced ideas understandable to people who are not in a position to invest heavily in preparatory study.

This paper describes several simple activities, designed for active participation by children or lay-people, which inculcate an understanding of seemingly impossible cryptographic techniques. The activities use only basic arithmetic and elementary puzzle-solving ability. Moreover, they are "unplugged" in that they do *not* require the use of a computer. This makes them widely accessible, regardless of hardware or software availability. Although this paper is aimed at the cryptographic community and at computer science educators, the techniques described in it are aimed at the general public. We have included some mathematics in this paper: for some audiences, especially teenagers (and even postgraduate computing students!) with some math background, it is fun to go beyond games and show that there are real challenges in the material.

We begin by describing simple activities that expose some of the issues involved, namely key distribution and information hiding. We demonstrate a protocol for coin-tossing over a telephone, and present two public-key cryptosystems that are based on puzzles that are simple to understand but hard to solve — in computer science parlance, intractable.

We have found the best approach for all of these activities is to have students work through them using concrete examples, explaining each step as they proceed. We prefer to allow them to discover for themselves how the methods work, where possible figuring out the completion of the task by themselves. They can then contemplate ways to attack the protocols, and ways to deter attacks.

Presentations have been used with a range of audiences, including schoolchildren (from junior/elementary to senior high school), teachers (on training courses) and lay people from the general public. In all cases members of the audience have reported surprise and intrigue at being able to understand how to do something that they previously thought would have been impossible. In some groups lengthy discussions have ensued on how such schemes might be attacked or improved. More details of our computer-free approach to presenting computer science is available from the "Unplugged" web site at `http://unplugged.canterbury.ac.nz/`, and Bell (2000) and Koblitz (1997) are papers describing our experiences.

A survey we did (Bell, 2000) of 212 attendees from five presentations found, for instance, that 84% of males and 78% of females had an increased interest in computing science afterwards. Between the authors, around 150 presentations of the material described in this paper have been delivered (both to students and to teachers) up to December 2000.

Introducing the key distribution problem

We start by using an overhead foil to show a facsimile Elizabethan cipher (Figure 1). This is of some historical interest, and illustrates many elementary issues. A squiggle occurs frequently, and looks from the context to be the letter *e*, but closer inspection shows that *t* is more frequent. As there are 73 symbols used in total, it cannot be a simple substitution cipher, even allowing for a few null characters. This makes decryption harder, but it appears that the clear word division has been retained, and this makes things easier. On closer examination, one might determine that A is coded as triangle and square symbols, amongst others. Such discussion readily draws enthusiastic interaction from the audience and leads into standard techniques for coding. Higenbottam (1973) gives further extracts from the diary, and provides a full solution.

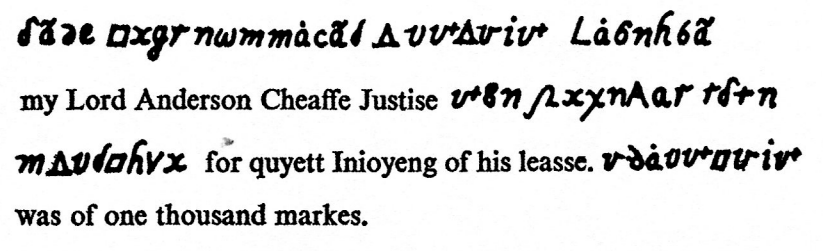


Figure 1. Coded entry from Sir Arthur Throckmorton's diary for Satterday [sic] 4 July 1584. The diary is at Canterbury Cathedral, and this illustration was based on F. Higenbottam, *Codes and Ciphers*, ©English Universities Press, 1973, p136.

Although Higenbottom was able to afford the hard and time-consuming work to decode this message, in general it requires the recipient to have the key. How is the key distributed? What happens if an eavesdropper acquires the key — for, by assumption, there are eavesdroppers, otherwise codes would not be needed in the first place!

To help explain the issues, we have used several methods.

Historical motivation. Breaking the World War II Enigma code is an exciting and well-documented story (e.g., Sebag-Motefiore, 2001). The deadly risks people took to obtain keys emphasises the significant roles of key management and distribution in conventional cryptography.

A chain and several padlocks. The loose chain is explained to be equivalent to a plain-text message. It is about a metre long, and its ends are painted a distinctive colour (this helps greatly when the chain is given to the audience, who tend to fumble if they can't find the ends quickly). When the ends of the chain are padlocked together, this is explained as representing a coded message. Clearly, to unlock the ends of the chain, a key is required.

A small lockable tool box or cash box (about 10cm by 5cm by 5cm). A tool box has the advantage that items the audience provides can be placed inside. It is then possible to send the actual messages that people in the audience have written, or even various goodies across the room. We have found that a robust metal toolbox is needed to withstand the most determined teenagers' attempts to open it without a key. It is important to have a lockable flap that is large enough to take several padlocks at once.[1]

We now divide the group into three (each group running front to back), and give the chain, padlock, and key to someone on, say, the right-hand side. The challenge is to pass the locked chain across the room to the left side, via the mischievous middle section. To avoid the chain being thrown across thoughtlessly (or even the key being lost), the audience is instructed that before anything happens, the people on the right-hand side must explain what is to happen so that everyone can follow it.

We emphasise that with the Internet, we want to have customers for e-commerce who we do not already know. We cannot get keys to everyone, and we don't know who to trust. This is a very different problem from the ordinary secret messages, where the sender and receiver know each other, and might, for instance share a one-time pad.

This illustrates the key distribution problem. With padlocks, there are several solutions. For instance, the chain could be sent across locked, locked with a second padlock on the other side, then returned. Back on the first side, the first padlock is removed (the key never having left that side), and the chain sent back, still padlocked by the left side's padlock. When the locked chain gets back, that side can remove the padlock easily with their key (which never left that side).

This technique relies on commutative coding, meaning that the resulting secret message text does not depend on what order the two coding systems were applied, and the final decoded message does not matter in which order the decoding steps are done in. This property is easily illustrated

[1] We have also used the toolbox 'metaphor' to help explain packet switching (one can use several boxes, numbered contents, and so on).

with a Caesar cipher, although the weakness of this particular cipher might already have been pointed out to the audience when discussing the Elizabethan cipher or other simple examples.

Here's how. Julius Caesar himself coded the alphabet by moving each letter along three, so A becomes D, B becomes E and so on. We will use two Caesar codes, one using a shift of 3 and one a shift of 5. Coding and decoding a single word is shown in Figure 2.

Original plain text	HELLO		HELLO
Code using 3	KHOOR	Code using 5	MJQQT
Code using 5	PMTTW	Code using 3	PMTTW
Message sent	PMTTW		PMTTW
Decode using 5	KHOOR	Decode using 3	MJQQT
Decode using 3	HELLO	Decode using 5	HELLO
Recovered plain text	HELLO		HELLO

Figure 2. Showing how HELLO can be coded by two Caesar codes in either order, and correctly decoded using the two codes, independently, in either order. Code 3 replaces letters three further along in the alphabet, e.g., A with D; Code 5 uses letters five along, e.g., A with F.

The technique, passing the box around the audience, also shows that the people in the middle saw three messages cross over. The audience may be able to deduce something from this activity — the existence of a message may constitute useful information in itself, and heightened activity may draw attention to itself. To avoid this, dummy messages could be sent to keep the level of activity consistent.

There is a problem with the two-padlock scheme, which we illustrate as follows. The middle group is given a third padlock. The left team now send their locked chain across. The middle group simulates the intended recipients — so far as the senders are concerned, the protocol has been followed exactly. Since they dare not enter into a plain-text discussion, they have no way of knowing that the intended recipient failed to get the message. Meanwhile, the middle group are celebrating having an unlocked chain. Even worse, the middle group can simulate the sender, and pass a locked message on to the intended recipient (possibly tampered with), and the victim again has no way of knowing that the code has been intercepted.

So not only do we have a key distribution problem (partly solved) but we have an authentication problem (not solved). We note that secrecy and identity are opposite poles. If you are very secret, how can anyone be sure who you are? These problems provide motivation for the public key systems described below.

How does the audience relate the chains to the digital world? Having prepared a sawn-up padlock, it is easy to show that a persistent code-breaker could always dismantle a padlock, or X-ray it, and hence crack the code. We show that knowing the inside of the padlock enables a key to be constructed (they are isomorphic). In the digital world we have to assume that it is easy for a code-breaker to see our message, and so techniques other than secrecy of the encryption method have to be employed.

Trap door functions

This leads into the abstract idea of one-way trapdoor functions, which are introduced as computer (or mathematical) objects that behave like padlocks: they are easy to lock, but hard to unlock unless you know the secret. In the later sections we introduce three one-way functions (Boolean circuits, perfect codes, and directed cycle partitions) that can be related back to the padlock demonstration.

Trapdoor functions are often introduced in terms of factoring integers, which is obviously relevant to cryptography, but is quite an abstract concept for many audiences. It is thus helpful to provide examples of trapdoor functions from the familiar physical world. We hold up a postcard or a large picture, and cut it into pieces. We now have a jigsaw. This can clearly be made as hard as one likes to solve (just cut into more pieces), but anyone who knows what the picture is supposed to be or how it was cut up finds it *much easier* to do.

Application of trapdoors: fair coin tossing

This activity involves using a one-way function to perform a coin-toss over a telephone. One of the better known cryptographic techniques relating to this is playing poker by telephone (Shamir, Rivest & Adleman, 1981). If an audience can be convinced that a fair coin toss is possible, the poker game becomes considerably more believable.

Again it is helpful to encourage the audience to consider how a fair coin toss could be achieved, and to point out how cheating is possible if you don't trust the person at the other end. The relevance of such techniques can be motivated (for children) by considering a coin toss with a rival school sports captain to determine where a game will be held, or (for adults) the problems faced in businesses when activities such as contracts, negotiations and voting need to be made electronically between people who can't completely trust each other. It is very effective to offer a candy bar to anyone in the audience who can guess whether a tossed coin comes down heads or tails;[2] of course, because the lecturer cheats (to demonstrate the need for guaranteed trust), nobody can win the reward!

If there are adults in the audience, reminding them that the game has a serious undertone is important. For example, take a large cheque book and write (or mime!) a generous cheque out for someone in the audience, perhaps to buy something off them. Explain that naturally you will deny that you ever signed the cheque, and in a world where anyone could do this, businesses would soon go bust. Remind them that if e-commerce (business over the internet) is to work, customers must not be able to repudiate money transfers. The game here shows how a single bit cannot be repudiated; obviously, by combining bits we could devise a protocol where entire cheques (up to whatever our generosity extends to) can be protected.

Depending on time or the audience, preparation time available, various techniques can be used:

Circuits

The first technique we use employs a randomly chosen boolean circuit, such as the one shown in Figure 3, as a one way function. In the figure there are six inputs and six outputs. The presenter will need to teach the operation of simple Boolean gates (only AND and OR gates are required). Instead of using 1s and 0s (as in Figure 3) it may be, for some audiences, more fun to colour in the 1s squares.

The coin toss operates as follows, using the traditional characters Alice and Bob to illustrate the process. Alice and Bob agree on a randomly chosen boolean circuit (perhaps they each supply half). Alice secretly selects a random input to the circuit (six zeroes and ones in the example), and calculates the output. She then supplies the output to Bob, via the phone (or across the room). Bob must guess the parity of the input (i.e., the number of ones). He could do this reliably if he could determine the input, but because a one-way function has been used, this is not possible. Thus he might as well toss a coin to guess the parity. If he is correct, he wins. Afterwards, Alice proves that she hasn't cheated by providing the input that produced the given output.

[2] Don't leave the candy on the overhead projector while engrossed in coin tossing!

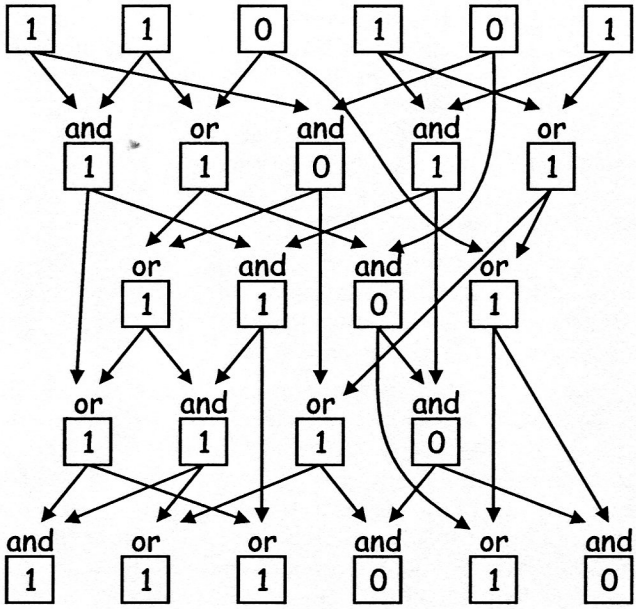


Figure 3. A boolean circuit used as a one-way function for the coin-toss.

Bob could cheat by finding an inverse function, but students can determine by trial and error that this is not a simple task. Alice could cheat by finding two inputs with opposite parity that produce the same output. However, since Bob has an equal and independent role in defining the circuit, she has no way to engineer such a situation, which means that all she can hope to do is to compute such a pair of inputs. Again, because the function is "one way" this is computationally difficult. Of course, for a six-bit domain and range a full enumeration of all mappings is possible, although this would be very tedious by hand.

There are many variations. If less time is available, a pre-prepared handout can be used. If the audience is younger, more time must be spent explaining AND and OR, and colouring can be used rather than writing binary digits. Explaining AND and OR also helps understand why the circuit is a trap door: if the output of an AND is zero, you do not know which of its inputs was zero; if the output of an OR is one, you do not know which of its inputs were one. Thus, although it is easy to go forwards 'through' the trap door, going backwards requires, at best, guessing.

A stronger system would use more bits, and this observation provides the opportunity to discuss how the strength of a cryptographic systems can be measured in bits, and the exponential increase in the search space as bits are added. Advanced audiences may like to be reminded of the well-known difficulty of solving satisfiability, which is the technical term for trying to invert the coin flip circuit in general. It is also worth pointing out that although we know that the code can be broken, we can ensure that it takes a long time to decode it. It would not matter if most business transactions were decoded, say, a year later, though keeping them secure for shorter periods might be very important.

Phonebook. An example suggested by Arto Salomaa (Bauer, 1997) is to use a phone book. Given someone's name, it is very easy to find a phone number (indeed this is what phone books are designed for). However, given a phone number it is very hard to discover the person's name, since this could mean reading through the entire phone book. A lively audience may realise that it is easy to find out a person's name — just ring them up, and ask who they are. Better, then, would be to make a checksum from the phone number, for instance adding the digits together — this is not a phone number that can be rung up, but it is determined one way from a person's name.

The phone book is used as follows: a coin toss represents one bit of a number (e.g., whether it is odd or even). So, choose a person's name that has an odd or even number of letters in their surname, and tell Bob the phone number. Thus Alice cannot change the name of the person, but

Bob does not know the name of the person. Bob guesses whether the name has an odd or even number of letters and challenges Alice.

Any book. Typically our talks show books that have had an interesting role in cryptography (e.g., Anderson, *et al*, 1999; Zimmermann, 1995). These books can be used at this point as more interesting aids than phone books!

Alice chooses an odd or even number to represent a head or tails toss, and then tells Bob several words on that page of a particular book. Alice cannot change the page number when challenged by Bob.

Calculators. A one-way function can be constructed using an identical pair of calculators. Alice enters a secret number into her calculator, and then presses the sin or square root, $\sqrt{}$, button a suitable number of times. The output of the function is (say) the last three digits displayed. Bob must then guess if Alice's original number was odd or even. More complicated, but more secure, is a function like $2x^2-1$ with a starting number around 0.3, which behaves chaotically.

These are examples of trapdoor functions, but how can they be used? The lecturer tosses the coin, and then looks up a person's name, beginning with either H or T. That person's phone number (or a checksum of it) can then be told to everyone in the room. The volunteer then guesses whether the coin came down heads or tails. The lecturer claims the volunteer is wrong, and of course can now be challenged to reveal the person's name used for the phone number. (This can also be done by having the audience member choose a name beginning with H or T, and the lecturer tosses a coin to guess which it is.)

We encourage the audience to invent further trapdoor functions, work out their weaknesses, and try to find solutions to the weaknesses.

The exact dominating set problem

Finding what is called an exact dominating set is an easily explained problem on graphs that can be used as an alternative one-way function for the coin-toss protocol, and also as the basis of the public key system described in the next section.

A exact dominating set[3] on a graph G=(V,E) is the set V′ such that for every vertex v there is exactly one u in N[v] with u in V′, where N[v] is the neighbourhood of v.

Of course, we don't use this definition with school-children! Instead, we present the graph as a street map, where edges are roads and vertices are intersections. Finding a dominating set is introduced as a resource placement problem: for example, students relate well to the "ice-cream vendor" problem, in which ice-cream vendors must be placed in a seaside township so that no-one has to walk past more than one intersection to buy an ice-cream. Figure 4 shows a map with a solution requiring just six vendors (open circles in the figure). Each intersection with a vendor "covers" (dominates) the neighboring intersections, so the solution shown covers the entire map.

[3] The exact dominating set is also known as the *perfect code problem*. The name "perfect codes" comes from coding theory, as the problem can be used to find the Hamming perfect error-correcting codes. This name may be confusing in this situation because it does not relate to cryptographic codes; in fact, we shall see that the cryptographic codes generated are far from perfect.

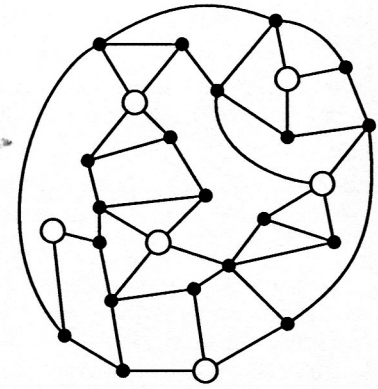


Figure 4. Solution to the ice-cream vendor problem using six vendors.

The solution in Figure 4 is an exact dominating set because every neighboring intersection is covered exactly once. An example of a dominating set that is not exact would be to put a vendor on every intersection.

After teaching children about the ice-cream vendor problem, we have them solve some sample maps. They soon discover that it is very difficult to find the minimal solution — in fact, the problem of determining whether a graph has a perfect code is NP-complete. We then show them how to generate their own maps to which they know the solution. This is very attractive, as a child can quickly draw a large map that they can solve, yet their parents and teachers can't.

The map is generated by first drawing the solution vertices as open circles, and adding random neighbouring vertices to them, as shown in Figure 5. At this stage the solution is trivially obvious. It is then disguised by randomly adding edges between the vertices, but not to any of the solution vertices. The map in Figure 4 was derived from the one in Figure 5 this way.

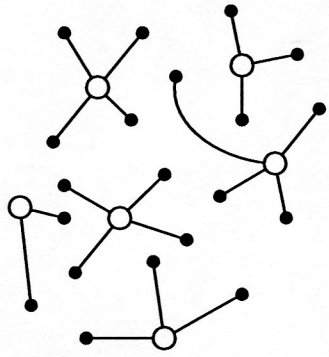


Figure 5. Generating an ice-cream vendor (perfect code) problem.

The problem can be used for the coin-toss protocol as follows. Alice devises a map that she knows the solution to, and she sends it to Bob. Bob must guess whether the cardinality of the solution is odd or even (any two perfect codes in a graph must have the same cardinality). Bob wins if he is correct, and if he loses, Alice can produce the perfect code to prove that he is wrong.

In the next section we show how to use the problem as the basis of a child-proof public key system.

A public key system

Public key systems have revolutionised security, providing methods for safe key distribution, authentication, secure financial transactions, and secure email.

To present the essence of the idea, we dress the problem up as passing a message in class so that even if the teacher knows how it was encoded, they can't decode it. The value of such a method is both obvious and attractive to students! Other applications can be mentioned, such as prisoners

communicating in front of a guard who can see everything they write, or sending a credit card number over the Internet with an eavesdropper recording every interaction.

The public key system is based on the dominating set problem described in the previous section. The public key is the graph, and the private key is the set of vertices that give an exact dominating set. The message to be encrypted is an integer m, which might represent a letter of the alphabet, or an entry in a codebook or a quantity of cash. The procedure is described briefly as follows. The Appendix provides a more expansive discussion, more suited to a handout.

1. Randomly associate an integer m_v with each vertex $v \in V$ so that $\sum_{v \in V} m_v = m$. In other words, for the code number m, find a set of random numbers that add up to it, and associate each number with a vertex.
2. Associate the integer s_v with each vertex $v \in V$, calculated as $s_v = \sum_{u \in N[v]} m_u$ where $N[v]$, as before, is the neighborhood of v, that is the set of all vertices adjacent to v, and v itself. In other words, replace each number on each vertex with the total of all the numbers on vertices immediately connected to it.
3. Transmit the s_v values. To transmit the values, you of course need an agreed (public) convention to send them in the right order, so that the recipient knows which number goes with which vertex.

To decrypt the message, sum the s_v values for the vertices that are solutions to the exact dominating set. This sum will include every m_v value exactly once, and therefore be equal to m.

We do not use the above description to present the system to a general audience. Rather, we work through a straight-forward example of the calculation as shown in Figure 6. In this example the message is the number 66. In the Figure, the first number at each vertex is m_v, and the number in parentheses is s_v. A solution to the perfect code is the vertices that have the m_v (s_v) values of 2 (13), 6 (13), 5 (22), and 1 (18), for which the s_v values sum to m = 66 which thus recovers the original number encoded.

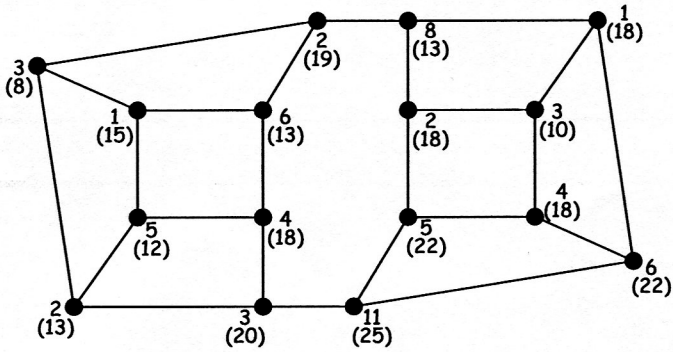


Figure 6. Calculations for the public key system to encode the number 66.

This system requires some care with arithmetic, and identifying adjacent vertices, but students will achieve a great deal by working through it.

This system is vulnerable to attack by forming a set of linear equations for the m_v values, which can be solved efficiently using Gaussian elimination. If the students have the mathematical maturity to see this, the observation provides an opportunity to discuss similar problems in "real" cryptosystems, which also rely on problems that we may learn of solutions for in the future.

A public key system based on the directed cycle partition problem

A public key system can also be based on the Directed Cycle Partition problem. A directed cycle partition is a variation on the Hamiltonian cycle for directed graphs. It is essentially a Hamiltonian cycle, but the graph is allowed to be partitioned before a cycle is found in each class.

A Hamiltonian cycle is the special case where the only partition allowed is the trivial partition with one class.

The directed cycle partition problem on a directed graph G=(V,E) is to determine whether the vertex set of G be partitioned so that there is a directed cycle through the vertices of each class of the partition. This problem is conjectured to be NP-complete.

In our public key system, the public key is the directed graph, and the private key is the partition and the directed cycles. To encrypt a message that is a positive integer m:

1. Randomly associate an integer m_v with each vertex $v \in V$ so that $\sum_{v \in V} m_v = m$
2. Randomly associate an integer r_v with each vertex $v \in V$.
3. Label each arc uv with the numerical value $m_u + r_u - r_v$
4. The encrypted message is the graph G together with the arc labels.

Decryption is performed by summing over the arc labels of the directed cycles. In this sum, the r_v terms cancel out, leaving m. This approach is vulnerable to the same linear algebra attack as the dominating set system.

The directed cycle partition problem should be introduced using a real-world problem that the audience can relate to. For example (continuing with the ice-cream theme), the problem might be to plan a route for a mobile ice-cream van that drives around, stopping at street corners — it is very important that the van does not go past a corner twice, because children will see the van and be disappointed that it doesn't stop. Partitioning can be introduced by allowing multiple vans.

Comparing information

Cryptography is a fascinating subject, with a very wide range of applications. It lends itself to being used in classroom settings such as those we described above.

Consider the problem of comparing information without revealing it. For example, Alice and Bob have heard confidential complaints about a sensitive matter; they wonder whether they have received the complaints about the same person or about two different people. How can Alice and Bob tell, without revealing the name of the person involved to each other? A paper on this problem (Fagin, Naor, & Winkler, 1999) provides thirteen different solutions; many of the solutions do not require computers. For example, if not too many possible names are involved, a set of boxes can be made, with each box labelled by a person's name. Alice and Bob then each put a marble (or other small object) in the appropriate labelled box. The labels are then removed (without looking inside the boxes). If one box contains two marbles, then Alice and Bob were told about the same person.

Politics of cryptography

Cryptography has been used for centuries for all sorts of activities, primarily spying and warfare (Kahn, 1996; Sebag-Montefiore, 2001). It is clearly an exciting subject, but the political interest in cryptography is not only historical. Martin Gardner wrote a column in the *Scientific American* in 1977 on trap door functions, which he discusses in (Gardner, 1989). As a result the US National Security Administration argued that public disclosure of trap door functions was illegal. The bizarre story of Pretty Good Privacy (Zimmermann, 1995), a freely available public key system, is now well-known — see Singh (1999) for a popular account (though one that is perhaps more popular than accurate, according to Diffie, 1999); more recent issues include the British Government's reconsideration of legislation that would have banned a conventional paper address book listing ordinary individuals' digital signatures (Anderson, *et al.*, 1999), and the story of Sarah Flannery (Flannery & Flannery, 2000), who won national and international science prizes for her work in cryptography.

Further discussion of these issues are beyond the scope of this paper. We refer the reader to Whitfield Diffie and Susan Landau's excellent and authoritative book (1998), and to Baker and Hurst (1998) who provide a world-wide perspective and survey of cryptographic issues, thus taking our "game" like approach into serious multinational legal issues.

Conclusions

The activities discussed above have been used with diverse groups of children and adults, ranging from elementary-school children to university students. They have been carried out with enthusiasm by young and old alike, and it is gratifying to see that many of those who participate reach an understanding of what are generally regarded as advanced cryptographic techniques. We urge other educators and cryptography experts to use these methods to improve the public understanding of these complex but important topics.

Acknowledgements

This paper has been developed from one presented at the First World Conference on Information Security Education (Bell, Thimbleby, Fellows & Witten, 1999). We are grateful to Josh Benaloh, Paddy Krishnan and Tad Takaoka for helpful discussions. Harold Thimbleby is a Royal Society-Wolfson Research Merit Award holder, and gratefully acknowledges this support.

References

Anderson, R., Crispo, B., Lee, J-H., Manifavas, C., Matyáš, W. Jr. & Petitcolas, F. (1999). *The Global Internet Trust Register*, Cambridge, Massachusetts: MIT Press.

Baker, S. A. & Hurst, P. R. (1998). *The Limits of Trust: Cryptography, Governments, and Electronic Commerce*, Kluwer Law International.

Bauer, F. L. (1997). *Decrypted Secrets*, Berlin: Springer-Verlag.

Bell, T. (2000). "A low-cost high-impact Computer Science show for family audiences," In *Australasian Computer Science Conference 2000* (ACSC 2000) (pp. 10–16), Canberra, Australia.

Bell, T., Thimbleby, H., Fellows, M., Witten, I. H. (1999). "Explaining Cryptographic Systems to the General Public," In Yngström, L. & Fischer-Hübner, S. (Eds.) *IFIP First World Conference on Information Security Education* (pp. 221–233), Department of Computer and Systems Science, Stockholm University, Report Series 99–008.

Diffie, W. (1999, September 10). "Of Riddles Wrapped in Enigmas," *Times Higher Education Supplement*, p. 25.

Diffie, W. & Landau, S. (1998). *Privacy on the Line: The Politics of Wiretapping and Encryption*, Cambridge, Massachusetts: MIT Press.

Fagin, R., Naor, M. & Winkler, P. (1999). "Comparing Information Without Leaking It," *Communications of the ACM*, **39**(5), 77–85.

Flannery, S. with Flannery, D. (2000). *In Code: A Mathematical Journey*, London: Profile Books.

Gardner, M. (1989). *Penrose Tiles to Trapdoor Ciphers*, New York: W. H. Freeman and Company.

Higenbottam, F. (1973). *Codes and Ciphers*, London: English Universities Press.

Kahn, D. (1996). *The Codebreakers*, Revised Edition, New York: Scribner.

Koblitz, N. (1997). "Cryptography as a Teaching Tool," *Cryptologia*, **21**(4), 317–326.

Sebag-Montefiore, H. (2001). *Enigma: The Battle for the Code*, London: Phœnix.

Shamir, A., Rivest, R. L. & Adleman, L. M. (1981). "Mental Poker," In Klarner, D. (Ed.), *The Mathematical Gardner* (pp. 37–43), Belmont, California: Wadsworth.

Singh, S. (1999). *The Code Book: The Secret History of Codes and Code-Breaking*, London: 4th. Estate.

Zimmermann, P. R. (1995). *The Official PGP User's Guide*, Cambridge, Massachusetts: MIT Press.

Appendix. A simple public key handout

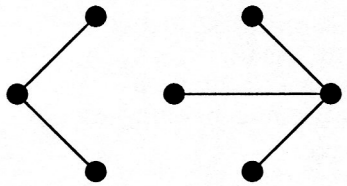
This handout describes how to create a public key and to use it, using a graphical problem called "dominating sets." For illustrative purposes, we'll use a simple key, that can be drawn by hand — the example is complicated enough to convey the right impression, but not so complicated that one runs out of time going through the entire example. We'll achieve a scheme where anyone in the world can send us a secret message (using our public key, which we let everyone in the world

know) but which only we can decode (using our private key that we keep secret and tell nobody). In our public key scheme, the secret message can be any number.

1 Create your private key

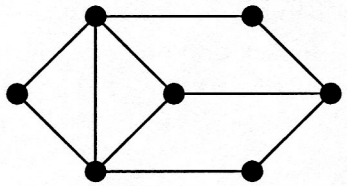
Put some dots on paper, and draw short lines from them. (It doesn't matter if the lines cross, but it's easier if they don't.) This is the private key; keep it secret.

In this example, we'll just use two dots. Using more dots makes it a *bit* harder to encode but *very much* harder to decode; with enough dots you can ensure that decoding (without the private key) is impossible in the time available, but that it is still easy-enough to encode.



2 Use your private key to make a public key

Join up the ends of the lines any way you like. Again, it's easier, but not necessary, if the lines don't cross. This is the public key that anyone can know and use. It is difficult to find the private key from the public key, which is the trapdoor effect that is required for the scheme to work.

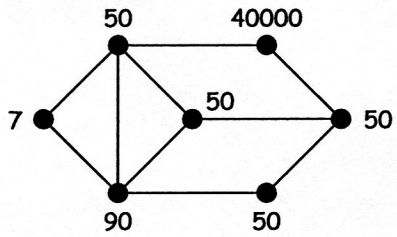


3 Distribute your public key around the world

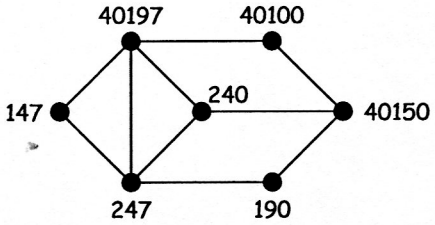
Don't forget to send instructions for using it — remember this is a public key system, and it does not matter that everyone knows how to use the system. The instructions can be public knowledge.

4 Someone somewhere wants to send a secret message to you

Someone somewhere else in the world might want to confide in you the date of a crime. Let's say it was the fourth of February, 1997, and nobody else in the world must find out. They must convert the message — in this case the date, 4/2/97 — into a number. So we'll use 40297 as the example secret message. There are 7 dots in the public key we have designed, so find any 7 numbers that add up to 40297. Our friends might choose an easy sum, 40297=40000+50+50+50+50+90+7, and they will put these numbers anywhere around the map on the dots:



Next our friends add each number to the next nearest numbers along the lines. So they replace 7 (on the left) with 7+50+90=147. They do this for each of the dots.



Then they send the map (with these numbers on it) to you. You might want to agree on a scheme to send the numbers without the map, for instance just send 4019 7401 0000 1470 0240 4015 0002 4700 190: messages can then be sent easily in email. Note that the way we have split up the numbers gives no hint on how big they are; this is something else that needs to be agreed between you.

5 You decode messages using your private key

Only you know that the dominating set is {147, 40150} because that is how you made your private key. Just add these numbers together, and you get 40297. So the day in question is the fourth of February, 1997.

6 Further thoughts...

Look up "dominating sets": you should find it is a computationally hard problem, in fact the difficulty doubles each time you add a node. Thus solving the code can be made pretty difficult by using a big enough graph. Unfortunately there's actually an easy way to crack this code. Can you find it? (*Hint*: write the problem as a set of equations.) Can you fix this weakness? (*Hint*: think of a way of coding a message using dominating sets that has no such equations.)