Explaining cryptographic systems to the general public

Tim Bell [*], Harold Thimbleby[†], Mike Fellows[‡], Ian Witten[§], and Neil Koblitz[¶]

April 12, 1999

Abstract

Modern cryptography can achieve levels of security and authentication that non-specialists find literally incredible. Techniques include information-hiding protocols, zero-knowledge proofs and public key cryptosystems; they 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.

1 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 which enable the transaction

[*]University of Canterbury, Christchurch, New Zealand. tim@cosc.canterbury.ac.nz
[†]Middlesex University, London, UK
[‡]University of Victoria, BC, Canada
[§]University of Waikato, Hamilton, New Zealand
[¶]University of Washington, Seattle, USA

to be carried out reliably without the bank being able to identify who the money is going to! 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. Understanding the technical issues involved goes a long way towards enabling informed decisions on privacy issues, just as an understanding of biology goes a long way towards making informed decisions on environmental 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 of 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, with variations that involve mappings to special symbols, and perhaps other private key systems such as the one-time pad. But most people have no idea about the advanced cryptographic techniques that are available to solve problems such as key distribution and authentication. Cryptographic experts often regard the techniques as 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.

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.

More details of our computer-free approach to presenting computer science is available from the "Unplugged" web site at `http://unplugged.canterbury.ac.nz/`

2 The key distribution problem

We start by using an overhead foil to show a facsimile Elizabethan cipher. 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. This discussion leads into standard techniques for coding.

To decode this sort of message requires the recipient to have a 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 use 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.

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.

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. This is easily illustrated with a Caesar cipher, although the weakness of this particular cipher has already been pointed out.

The technique 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.

This leads into the abstract idea of one-way trapdoor functions, which are introduced as 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.

3 Information hiding

The next activity illustrates the role of random numbers in keeping information private. The challenge to the group is to work out the average age (or salary, or some other sensitive value) of the group, without anyone revealing their age to anyone else. If there are adults in the group then their reaction to the challenge often confirms that the information is indeed sensitive!

We find it helpful to begin by asking the group to suggest how this could be done. Usually the methods suggested will involve leaking some information.

The method that we use involves passing around a pad of paper, with one sheet for each person. The first person writes down a four-digit random number (which they retain), adds their age to it, and passes the pad with the sum written on the top page to the next person. (At this time you can point out that the only information available is that the person's age is less than some four digit number.) That person adds their age to the number, tears off the top page, and passes their sum on to the third person. This continues around the group. At the end the first person is given the pad; they subtract their original random number, and divide by the number of people in the group to obtain the average age. Rather than explaining the whole procedure beforehand, we recommend getting the group started and allowing them to work out for themselves how to complete the task.

This can be followed by discussion about how people might attempt to cheat, and why it may or may not work. A primitive voting scheme could be proposed, where each person adds one or zero to the total depending on whether they are for or against an issue. Problems such as people adding more than one, or less than zero, can be discussed.

4 Coin tossing by telephone

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 [1]. 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.

The technique we use employs a randomly chosen boolean circuit, such as the one shown in Figure 1, 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).

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

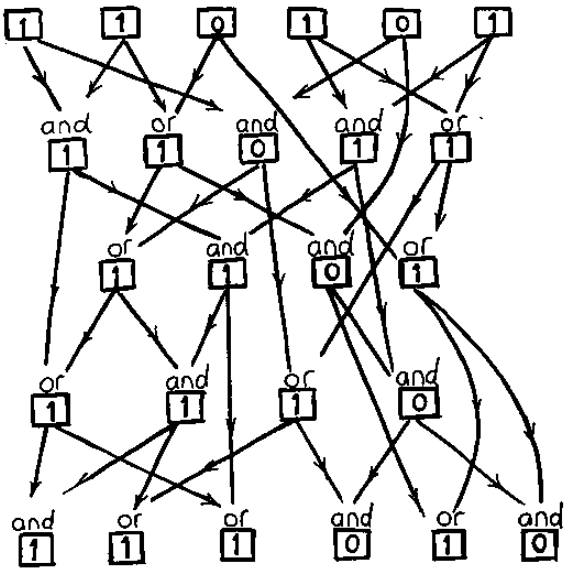

Figure 1: A boolean circuit used as a one-way function for the coin-toss

to Bob, via the phone. 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.

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. 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.

A simpler 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 sqrt) button three 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.

5 The perfect code problem

Finding a perfect code 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 perfect code[1] 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 neighborhood 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 2 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.

The solution in Figure 2 is a perfect code (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 neighboring vertices to them, as shown in Figure 3. At this stage the solution is trivially obvious. It is then disguised by randomly adding edges between the vertices, but not to

[1]The perfect code problem is also known as the *exact dominating set* 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.

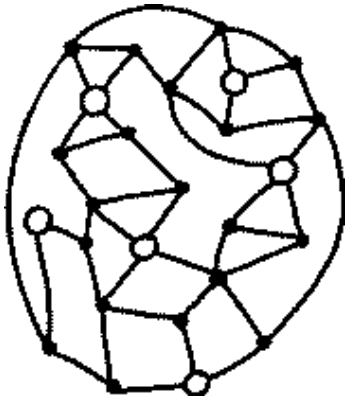

Figure 2: Solution to the ice-cream vendor problem using six vendors

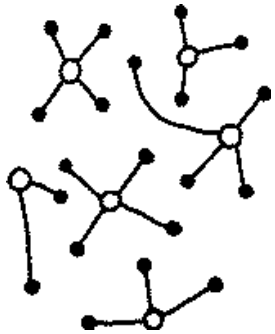

Figure 3: Generating an ice-cream vendor (perfect code) problem.

any of the solution vertices. The map in Figure 2 was derived from the one in Figure 3 this way.

The perfect code 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 perfect code as the basis of a child-proof public key system.

6 A Public Key System

Public key systems have revolutionised network 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 perfect code 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. The procedure is as follows:

1. Randomly associate an integer m_v with each vertex $v \in V$ so that
$$\sum_{v \in V} m_v = m$$
2. Associate the integer s_v with each vertex $v \in V$, calculated as
$$s_v = \sum_{u \in N[v]} m_u$$
where $N[v]$ is the neighborhood of v i.e. the set of all vertices adjacent to v, and v itself.
3. Transmit the s_v values.

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.

Again, we would not use the above description to present the system to a general audience. Rather, we work through an example of the calculation as shown in Figure 4. In this example the message is the number 66. 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 $m = 66$.

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

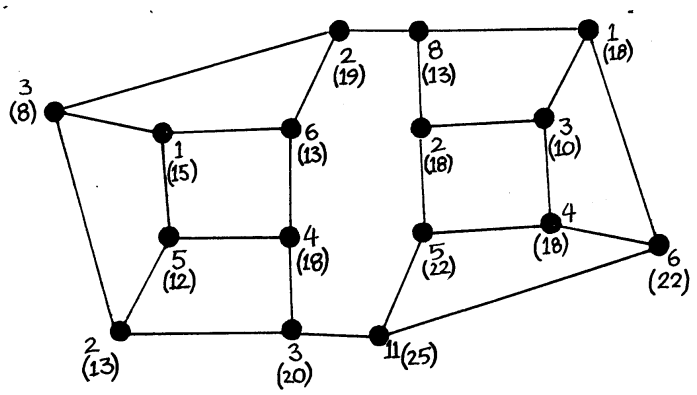

Figure 4: Calculations for the public key system to encode the number 66

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 may be solved in the future.

7 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

$$m_u + r_u - r_v$$

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 is vulnerable to the same linear algebra attack as the perfect code 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.

8 Conclusion

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 hope that other educators and cryptography experts will use these methods to improve the public understanding of these complex but important topics.

9 Acknowledgements

We are grateful to Josh Benaloh, Paddy Krishnan, and Tad Takaoka for helpful discussions.

References

[1] A. Shamir, R. L. Rivest, and L. M. Adleman. Mental poker. In D. Klarner, editor, *The Mathematical Gardner*, pages 37–43. Wadsworth, Belmont, California, 1981.