Activity 17

The Peruvian coin flip—*Cryptographic protocols*

Age group	Older elementary and up.
Abilities assumed	Requires counting, and recognition of odd and even numbers. Some understanding of the concepts *and* and *or* is helpful. Children will get more out of this activity if they have learned binary number representation (see Activity 1, Count the dots), the concept of *parity* (see Activity 4, Card flip magic), and have seen the example of one-way functions in Activity 14, Tourist Town.
Time	About 30 minutes.
Size of group	From individuals to the whole classroom.

Focus

Boolean logic.

Functions.

Puzzle solving.

Summary

This activity shows how to accomplish a simple, but nevertheless 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.

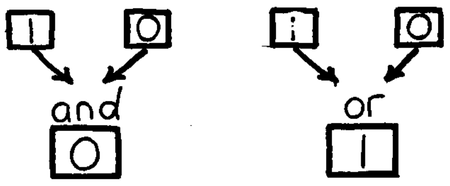


Figure 17.1: An *and*-gate and an *or*-gate

Technical terms

Distributed coin-tossing, computer security, cryptography, cryptographic protocol, *and*-gate, *or*-gate, combinatorial circuit.

Materials

Each group of children will need:

a copy of the reproducible sheet on page 183, and

about two dozen small buttons or counters of two different colors.

What to do

This activity was originally devised when one of the authors (MRF) was working with children in Peru, hence the name. You can customize the story to suit local conditions.

The women's soccer teams of Lima and Cuzco have to decide who gets to be the home team for the championship game. The simplest way would be to flip a coin. But the cities are far apart, and Alicia, representing Lima, and Benito, representing Cuzco, cannot spend the time and money to get together to flip a coin. 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 can simply say "sorry, it was tails" and Benito would be none the wiser. Alicia is not naturally deceitful but this, after all, is an important contest and the temptation is awfully strong. Even if Alicia were truthful, would Benito believe that if he 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, although admittedly in practice it could turn out to be more than a little tedious (fax machines would help!). During the construction process, each has an interest in ensuring that the circuit is complex enough that the other will be unable 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 (Figure 17.1). Each of the inputs can be either a 0 or a 1, which can be interpreted as *false* and *true*, respectively. The output of an *and*-gate is one (*true*) only if both inputs are one (*true*), and zero (*false*) otherwise. For example, the *and*-gate in Figure 17.1 has a one and a zero on its

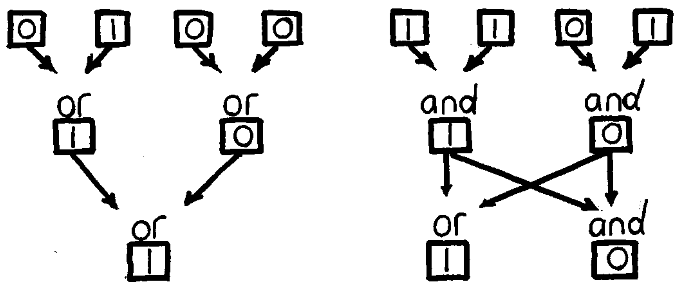


Figure 17.2: Combinations of *or*-gates and *and*-gates

inputs (at the top), so the output (the square at the bottom) is a zero. The output of an *or*-gate is one (*true*) if either (or both) of the inputs is one (*true*), and zero (*false*) only if both the inputs are zero. Thus in Figure 17.1 the output of the *or*-gate is a one for the inputs zero and one.

The output of one gate can be connected to the input of another (or several others) to produce a more complicated effect. For example, in the left-hand circuit of Figure 17.2 the outputs from two *or*-gates are connected to the inputs of a third *or*-gate, with the effect that if any of the four inputs is a one then the output will be a one. In the right-hand circuit of Figure 17.2 the outputs of each of the top two *and*-gates feeds into the lower two gates, so the whole circuit has two values in its output.

For the Peruvian coin flip we need even more complex circuits. The circuit in the reproducible sheet on page 183 has six inputs and six outputs. Figure 17.3 shows a worked example for one particular set of input values.

The way that 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 (zeros or ones), which she keeps secret. She puts the six 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 ones—in other words, she 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 (in fact, he could even toss a coin to choose!) Benito wins—and the playoff is in Cuzco—if his guess is correct; Alicia wins—and the playoff is in Lima—if Benito guesses incorrectly. Once Benito has told Alicia his guess, Alicia reveals her secret input so that Benito can confirm that it produces the claimed output.

1. Divide the children into small groups, give each group the circuit and some counters, and explain the story. The situation will probably be more meaningful to the children if they imagine one of their sports captains organizing the toss with a rival school. Establish a convention for the counter colors—red is 0, blue is 1, or some such—and have the children mark it on the legend at the top of the sheet to help them remember.

2. Show the children how to place counters on the inputs to show the digits that Alicia chooses. Then explain the rules of *and*-gates and *or*-gates, which are summarized at the bottom of the sheet (consider getting the children to color these in).

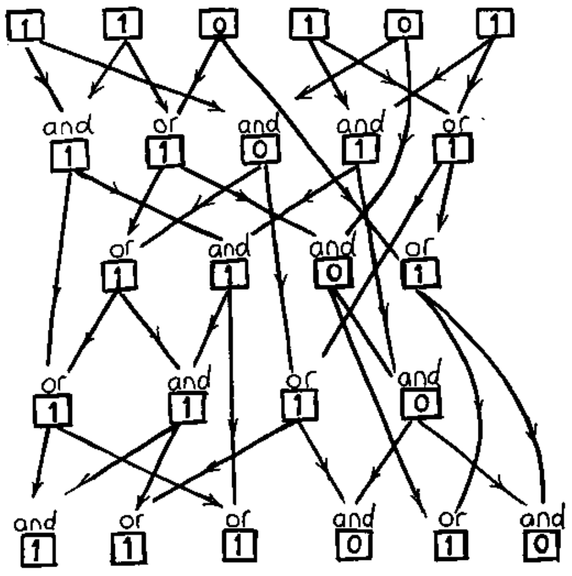


Figure 17.3: An example of Alicia's work

3. Show how to work through the circuit, placing counters at the nodes, to derive the corresponding output. This must be done accurately and takes some care; Table 17.1 (which should *not* be given to the children) shows the output for each possible input for your own reference in case of any doubt.

4. Now each group should elect an Alicia and a Benito. The group can split in half and each half side with Alicia or Benito respectively. Alicia should choose a random input for the circuit, calculate the output, and tell it to Benito. Benito guesses the parity of the input (whether it has an odd or even number of ones in it). It should become evident during this process that Benito's guess is essentially random. Alicia then tells everyone what the input was, and Benito wins if she guessed the correct parity. Benito can verify that Alicia's didn't change her chosen input by checking that it gives the correct output from the circuit.

   At this point the coin toss has been completed.

Benito can cheat if, given an output, he can find the input that produced it. Thus it is in Alicia's interests to ensure that the function of the circuit is *one-way*, in the sense discussed in Activity 14, to prevent Benito cheating. A one-way function is one for which the output is easy to calculate if you know what the input is, but the input is very difficult to calculate for a given output.

Alicia can cheat if she can find two inputs of opposite parity that produce the same output. Then, whichever way Benito guesses, Alicia can reveal the input that shows him to be wrong. Thus it is in Benito's interests to ensure that the circuit does not map many different inputs to the same output.

Input	000000	000001	000010	000011	000100	000101	000110	000111
Output	000000	010010	000000	010010	010010	010010	010010	010010
Input	001000	001001	001010	001011	001100	001101	001110	001111
Output	001010	011010	001010	011010	011010	011010	011010	011111
Input	010000	010001	010010	010011	010100	010101	010110	010111
Output	001000	011010	001010	011010	011010	011010	011010	011111
Input	011000	011001	011010	011011	011100	011101	011110	011111
Output	001010	011010	001010	011010	011010	011010	011010	011111
Input	100000	100001	100010	100011	100100	100101	100110	100111
Output	000000	010010	011000	011010	010010	010010	011010	011010
Input	101000	101001	101010	101011	101100	101101	101110	101111
Output	001010	011010	011010	011010	011010	011010	011010	011111
Input	110000	110001	110010	110011	110100	110101	110110	110111
Output	001000	011010	011010	011010	011010	111010	011010	111111
Input	111000	111001	111010	111011	111100	111101	111110	111111
Output	001010	011010	011010	011010	011010	111010	011010	111111

Table 17.1: Input-Output function for the circuit in Figure 17.3

5. See if the children can find a way for Alicia or Benito to cheat.

   From the first line of Table 17.1 you can see that several different inputs generate the output 010010—for example, 000001, 000011, 000101, etc. Thus if Alicia declares the output 010010, she can choose input 000001 if Benito guesses that the parity is even, and 000011 if he guesses that it is odd.

   With this circuit, it is hard for Benito to cheat. But if the output happens to be 011000, then the input must have been 100010—there is no other possibility (you can see this by checking right through Table 17.1). Thus if this is the number that Alicia happens to come up with, Benito can guess even parity and be sure of being correct. A computer-based system would use many more bits, so there would be too many possibilities to try (each extra bit doubles the number of possibilities).

6. Now ask the groups of children to devise their own circuits for this game. See if they can find a circuit that makes it easy for Alicia to cheat, and another that makes it easy for Benito to cheat. There is no reason why the circuit has to have six inputs, and it may even have different numbers of inputs and outputs.

Variations and extensions

1. An obvious problem in practice is the cooperation that is needed to construct a circuit acceptable to both Alicia and Benito. This might make the activity fun for the kids, but is likely to render the procedure inoperable in practice—particularly over the phone! However, there is a simple alternative in which Alicia and Benito construct their circuits independently and make them publicly available. Then Alicia puts her secret input through *both* circuits, and joins the two outputs together by comparing corresponding bits and making the final output a one if they are equal and zero otherwise. In this situation, neither participant can cheat if the other doesn't, for if just one of the circuits is a one-way function then the combination of them both is also a one-way function.

The next two variations relate not to cryptographic protocols or the coin-tossing problem *per se*, but rather to the idea of circuits constructed out of *and* and *or* gates. They explore some important notions in the fundamentals not only of computer circuits, but of logic itself. This kind of logic is called Boolean algebra , named after the mathematician George Boole (1815–64).

2. The children may have noticed that the all-zero input, 000000, is bound to produce the all-zero output, and likewise the all-one input 111111 is bound to produce the all-one output. (There may be other inputs that produce these outputs as well; indeed, there are for the example circuit—000010 produces all zeros, while 110111 produces all ones.) This is a consequence of the fact that the circuits are made up of *and* and *or* gates. By adding a *not*-gate (Figure 17.4), which takes just one input and produces the reverse as output (i.e. $0 \rightarrow 1$ and $1 \rightarrow 0$), the children can construct circuits that don't have this property.

3. Two other important kinds of gate are *and-not* and *or-not*, which are like *and* and *or* but followed by a *not*. Thus a *and-not* b is *not* (a *and* b). These do not allow any functionally

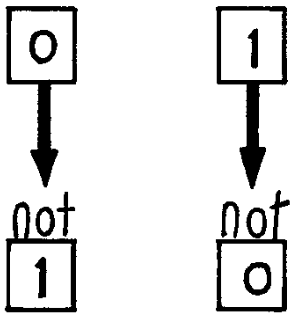


Figure 17.4: The *not*-gate

different circuits to be achieved, since their effect can always be obtained with the corresponding *and* or *or* gate, followed by *not*. However, they have the interesting property that all other gate types can be made out of *and-not* gates, and also out of *or-not* gates.

Having introduced *and-not* and *or-not*, challenge the children to discover whether any of the gates can be made from other gates connected together, and further, if they can be made from just one type of gate connected together. Figure 17.5 shows how the three basic gates, *and*, *or* and *not*, can be constructed from *and-not* gates, in the top row, and *or-not* gates, in the bottom row.

What's it all about?

Recent years have seen huge increases in the amount of commerce being conducted over computer networks, and it is essential to guarantee secure interchange of electronic funds, confidential transactions, and signed, legally binding, documents. The subject of *cryptography* is about communicating in secure and private ways. Two decades ago, computer science researchers discovered the counter-intuitive result that secrecy can be guaranteed by techniques that ensure that certain information is kept *public*. The result is the so-called "public key cryptosystem" of Activity 18, Kid Krypto, that is now regarded as the only completely secure way of exchanging information.

Cryptography is not just about keeping things secret, but about placing 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 unforgeable digital signatures and the ability to tell others that you possess a secret (like a password) without actually 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.

In a real situation, Alicia and Benito would not design a circuit themselves, but acquire a computer program that accomplishes the transformation internally. Probably neither would be interested in the innards of the software. But both would want to rest assured that the other is

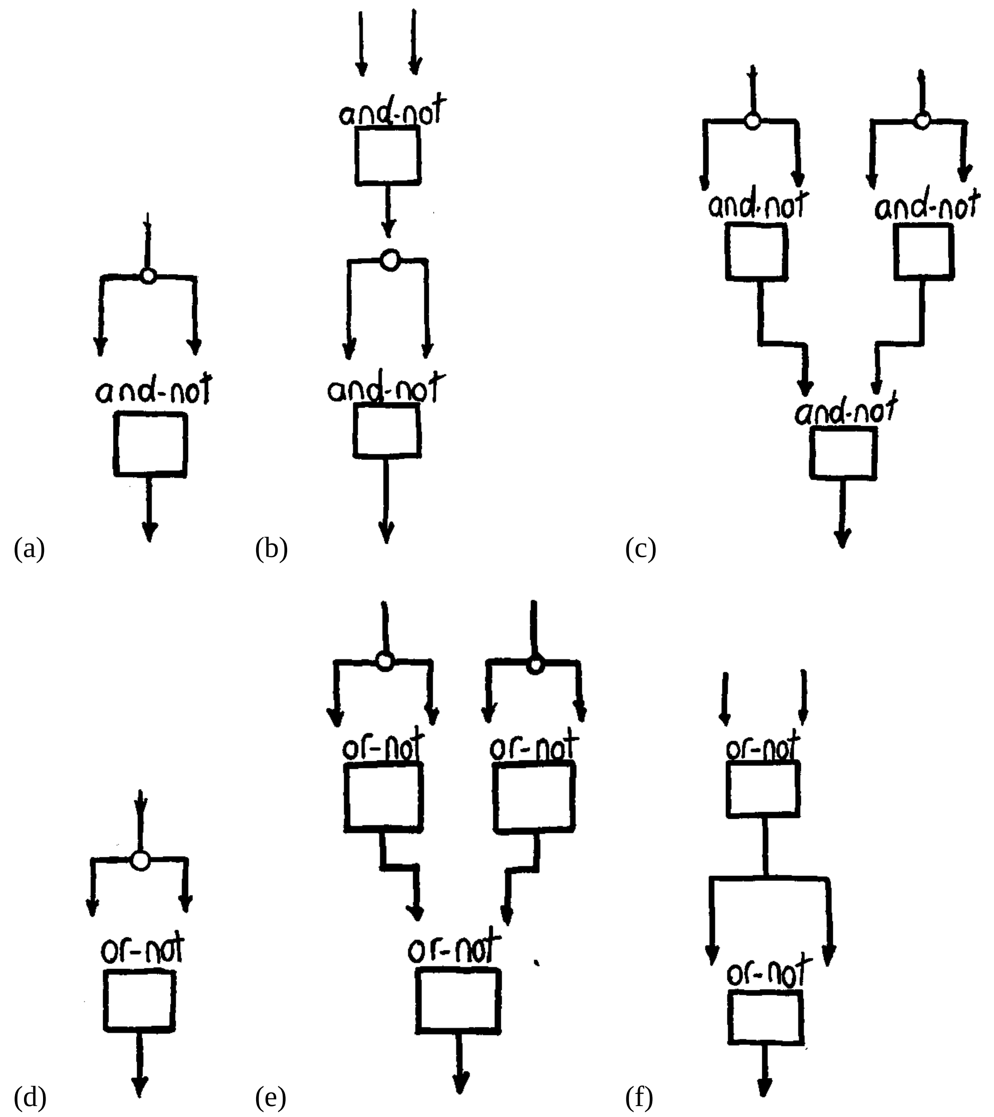


Figure 17.5: Making the three basic gates from *and-not* and *or-not* gates. (a) and (d) are *not*-gates, (b) and (e) are *and*-gates, while (c) and (f) are *or*-gates.

unable to influence the outcome of the decision, no matter how good their computer skills and how hard they tried.

In principle, any disputes would have to be resolved by appeal to a neutral judge. The judge would be given the circuit, Alicia's original binary number, the output that she originally sent Benito, and the guess that Benito sent in return. Once the interchange is over, all this is public information, so both participants will have to agree that this is what the outcome was based on. The judge will be able to put Alicia's original number through the circuit and check that the output is as claimed, and therefore decide whether the decision has been made fairly. Needless to say, the very fact that there is a clear procedure to check that the rules have been followed makes it unlikely that a dispute will arise. Compare with the situation where Alicia flips an actual coin and Benito calls heads or tails—no judge would take on that case!

A circuit as small as the one illustrated would not be much use in practice, for it is easy to come up with Table 17.1 and use it to cheat. Using thirty-two binary digits in the input would provide better protection. However, even this does not *guarantee* that it is hard to cheat—that depends on the particular circuit. Other methods could be used, such as the one-way function introduced in Activity 14, Tourist Town. Methods used in practice often depend on the factoring of large numbers, which is known to be a hard problem (although, as we will learn at the end of the next activity, it is not NP-complete). It is easy to check that one number is a factor of another, but finding the factors of a large number is very time consuming. This makes it more complex for Alicia and Benito (and the judge) to work through by hand, although, as noted above, in practice this will be done by off-the-shelf software.

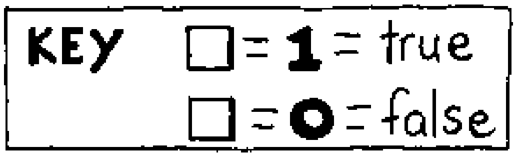
Digital signatures are based on a similar idea. By making public the output of the circuit for the particular secret input that she has chosen, Alicia is effectively able to prove that she is the one who generated the output—for, with a proper one-way function, no-one else can come up with an input that works. No-one can masquerade as Alicia! To make an actual digital signature, a more complex protocol is needed to ensure that Alicia can sign a particular message, and also to ensure that others can check that Alicia was the signatory even if she claims not to be. But the principle is the same.

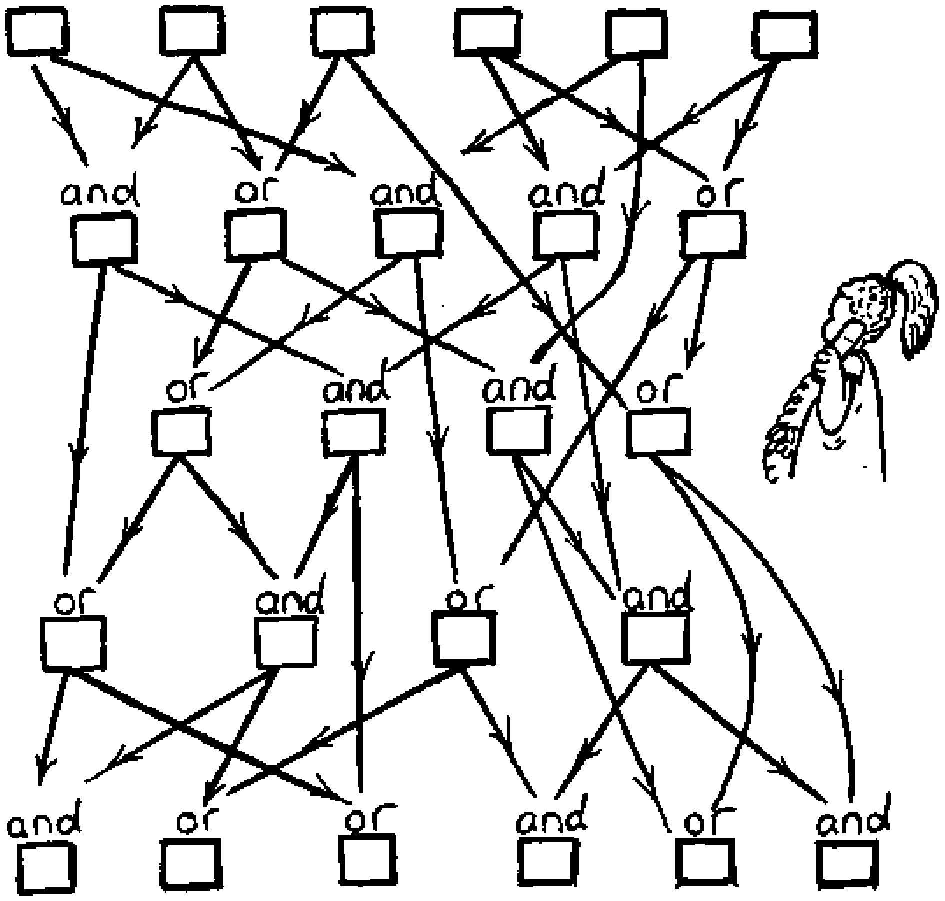
Another application is playing poker over the phone, in an environment in which there is no referee to deal the cards and record both player's hands. Everything must be carried out by the players themselves, with recourse to a judge at the end of the game in the event of a dispute. Similar situations arise in earnest with contract negotiations. Obviously, players must keep their cards secret during the game. But they must be kept honest—they must not be allowed to claim to have an ace unless they actually have one! This can be checked by waiting until the game is over, and then allowing each player to inspect the other's original hand and sequence of moves. Another problem is how to deal the cards while keeping each player's hand secret until after the game. Surprisingly, it is possible to accomplish this using a cryptographic protocol not dissimilar to the coin-tossing one.

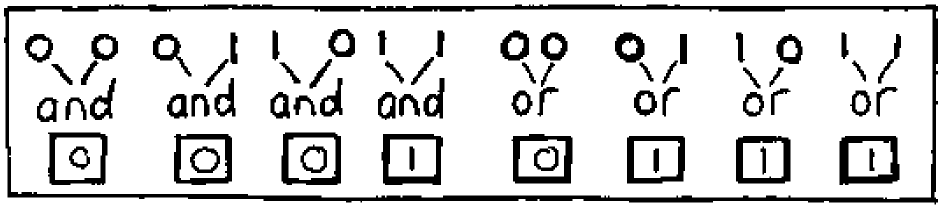
Cryptographic protocols are extremely important in electronic transactions, whether to identify the owner of a smart card, to authorize the use of a cellphone for a call, or to authenticate the sender of an electronic mail message. The ability to do these things reliably is crucial to the success of electronic commerce.

Further reading

Harel's book *Algorithmics* discusses digital signatures and associated cryptographic protocols. It also shows how to play poker over the phone, an idea that was first raised in 1981 in a chapter called "Mental poker", in the book *The Mathematical Gardener*, edited by D.A. Klarner. *Cryptography and data security* by Dorothy Denning is an excellent computer science text on cryptography. Dewdney's *Turing Omnibus* has a section on Boolean logic that discusses the building blocks used for the circuits in this activity.







Instructions: *Choose some inputs for this circuit and work out what the outputs are.*