13

Zero-knowledge Proofs

13.1 Interactive Proof Systems

Very informally, a zero-knowledge proof system allows one person to convince another person of some fact without revealing any information about the proof. We first discuss the idea of an interactive proof system. In an interactive proof system, there are two participants, Peggy and Vic. Peggy is the *prover* and Vic is the *verifier*. Peggy knows some fact, and she wishes to prove to Vic that she does.

It is necessary to describe the kinds of computations that Peggy and Vic will be allowed to perform, and also to describe the interaction that takes place. It is convenient to think of both Peggy and Vic as being probabilistic algorithms. Peggy and Vic will each perform private computations, and each of them has a private random number generator. They will communicate to each other through a communication channel. Initially, Peggy and Vic both possess an input x. The object of the interactive proof is for Peggy to convince Vic that x has some specified property. More precisely, x will be a yes-instance of a specified decision problem Π.

The interactive proof, which is a challenge-and-response protocol, consists of a specified number of rounds. During each round, Peggy and Vic alternately do the following:

1. receive a message from the other party
2. perform a private computation
3. send a message to the other party.

A typical *round* of the protocol will consist of a *challenge* by Vic, and a *response* by Peggy. At the end of the proof, Vic either *accepts* or *rejects*, depending on whether or not Peggy successfully replies to all of Vic's challenges. We define the protocol to be an *interactive proof system* for the decision problem Π if the following two properties are satisfied whenever Vic follows the protocol:

FIGURE 13.1
Graph Isomorphism

Problem Instance Two graphs on n vertices, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$.
Question Is there a bijection $\pi : V_1 \rightarrow V_2$ such that $\{u, v\} \in E_1$ if and only if $\{\pi(u), \pi(v)\} \in E_2$? (In other words, are G_1 and G_2 *isomorphic*?)

completeness

If x is a yes-instance of the decision problem Π, then Vic will always accept Peggy's proof.

completeness

If x is a no-instance of Π, then the probability that Vic accepts the proof is very small.

We will restrict our attention to interactive proof systems in which the computations performed by Vic can be done in polynomial time. On the other hand, we do not place any bound on the computation time required by Peggy (informally, Peggy is "all-powerful").

We begin by presenting an interactive proof system for the problem of **Graph Non-isomorphism**. The **Graph Isomorphism** problem is described in Figure 13.1. This is an interesting problem since no polynomial-time algorithm to solve it is known, but it is not known to be NP-complete.

We will present an interactive proof system which will allow Peggy to "prove" to Vic that two specified graphs are not isomorphic. For simplicity, let us suppose that G_1 and G_2 each have vertex set $\{1, \ldots, n\}$. The interactive proof system for **Graph Non-isomorphism** is presented in Figure 13.2.

We present a toy example.

Example 13.1
Suppose $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, where $V = \{1, 2, 3, 4\}$, $E_1 = \{12, 14, 23, 34\}$ and $E_2 = \{12, 13, 14, 34\}$.

Suppose in some round of the protocol that Vic gives Peggy the graph $H = (V, E_3)$, where $E_3 = \{13, 14, 23, 24\}$ (see Figure 13.3). The graph H is isomorphic to G_1 (one isomorphism from H to G_1 is the permutation (1 3 4 2)). So Peggy answers "1." □

It is easy to see that this proof system satisfies the completeness and soundness properties. If G_1 is not isomorphic to G_2, then j will equal i in every round, and

FIGURE 13.2
An interactive proof system for Graph Non-isomorphism

Input: two graphs G_1 and G_2, each having vertex set $\{1, \ldots, n\}$

1. Repeat the following steps n times:
2. Vic chooses a random integer $i = 1$ or 2 and a random permutation π of $\{1, \ldots, n\}$. Vic computes H to be the image of G_i under the permutation π, and sends H to Peggy.
3. Peggy determines the value j such that G_j is isomorphic to H, and sends j to Vic.
4. Vic checks to see if $i = j$.
5. Vic accepts Peggy's proof if $i = j$ in each of the n rounds.

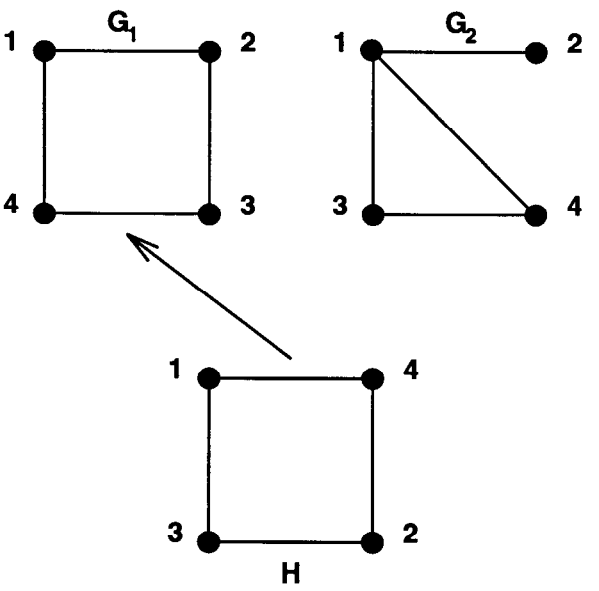

FIGURE 13.3
Peggy's non-isomorphic graphs and Vic's challenge

Vic will accept with probability 1. Hence, the protocol is complete.

On the other hand, suppose that G_1 is isomorphic to G_2. Then any challenge graph H submitted by Vic is isomorphic to both G_1 and G_2. Peggy has no way of determining if Vic constructed H as an isomorphic copy of G_1 or of G_2, so she can do no better than make a guess $j = 1$ or 2 for her response. The only way that Vic will accept is if Peggy is able to guess all n choices of i made by Vic. Her probability of doing this is 2^{-n}. Hence, the protocol is sound.

Notice that Vic's computations are all polynomial-time. We cannot say anything about Peggy's computation time since the **Graph Isomorphism** problem is not known to be solvable in polynomial time. However, recall that we assumed that Peggy has infinite computing power, so this allowed under the "rules of the game."

13.2 Perfect Zero-knowledge Proofs

Although interactive proof systems are of interest in their own right, the most interesting type of interactive proof is a zero-knowledge proof. This is one in which Peggy convinces Vic that x possesses some specified property, but at the end of the protocol, Vic still has no idea of how to prove (himself) that x has this property. This is a very tricky concept to define formally, and we present an example before attempting any definitions.

In Figure 13.4, we present a zero-knowledge interactive proof for **Graph Isomorphism**. A small example will illustrate the workings of the protocol.

Example 13.2

Suppose $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, where $V = \{1, 2, 3, 4\}$, $E_1 = \{12, 13, 14, 34\}$ and $E_2 = \{12, 13, 23, 24\}$. One isomorphism from G_2 to G_1 is the permutation $\sigma = (4\ 1\ 3\ 2)$.

Now suppose in some round of the protocol that Peggy chooses the permutation $\pi = (2\ 4\ 1\ 3)$. Then H has edge set $\{12, 13, 23, 24\}$ (see Figure 13.5).

If Vic's challenge is $i = 1$, then Peggy gives Vic the permutation π and Vic checks that the image of G_1 under π is H. If Vic's challenge is $i = 2$, then Peggy gives Vic the composition $\rho = \pi \circ \sigma = (3\ 2\ 1\ 4)$ and Vic checks that the image of G_2 under ρ is H. □

Completeness and soundness of the protocol are easy to verify. It is easy to see that the probablity that Vic accepts is 1 if G_1 is isomorphic to G_2. On the other hand, if G_1 is not isomorphic to G_2, then the only way for Peggy to deceive Vic is for her to correctly guess the value i that Vic will choose in each round, and write a (random) isomorphic copy of G_i on the communication tape. Her probability

FIGURE 13.4
A perfect zero-knowledge interactive proof system for Graph Isomorphism

Input: two graphs G_1 and G_2, each having vertex set $\{1, \ldots, n\}$

1. Repeat the following steps n times:
2. Peggy chooses a random permutation π of $\{1, \ldots, n\}$. She computes H to be the image of G_1 under the permutation π, and sends H to Vic.
3. Vic chooses a random integer $i = 1$ or 2 and sends it to Peggy.
4. Peggy computes a permutation ρ of $\{1, \ldots, n\}$ such that H is the image of G_i under ρ. Peggy sends ρ to Vic. (If $i = 1$, then Peggy defines $\rho = \pi$; and if $i = 2$, then Peggy defines ρ to be the composition of σ and π, where σ is some fixed permutation such that the image of G_2 under σ is G_1.)
5. Vic checks to see if H is the image of G_i under ρ.
6. Vic accepts Peggy's proof if H is the image of G_i in each of the n rounds.

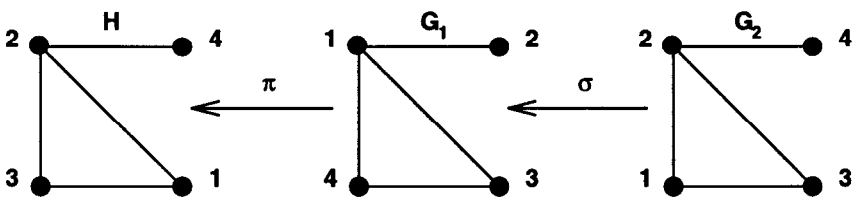

FIGURE 13.5
Peggy's isomorphic graphs

of correctly guessing Vic's n random challenges is 2^{-n}.

All of Vic's computations can be done in polynomial time (as a function of n, the number of vertices in G_1 and G_2). Although it is not necessary, notice that Peggy's computations can also be done in polynomial time provided that she knows the existence of one permutation σ such that the image of G_2 under σ is G_1.

Why would we refer to this proof system as a zero-knowledge proof? The reason is that, although Vic is convinced that G_1 is isomorphic to G_2, he does not gain any "knowledge" that would help him find a permutation σ that carries G_2 to G_1. All he sees in each round of the proof is a random isomorphic copy H of the graphs G_1 and G_2, together with a permutation that carries G_1 to H or G_2 to H (but not both!). But Vic can compute random isomorphic copies of these graphs by himself, without any help from Peggy. Since the graphs H are chosen independently and at random in each round of the proof, it seems unlikely that this will help Vic find an isomorphism from G_1 to G_2.

Let us look carefully at the information that Vic obtains by participating in the interactive proof system. We can represent Vic's view of the interactive proof by means of a *transcript* that contains the following information:

1. the graphs G_1 and G_2
2. all the messages that are transmitted by both Peggy and Vic
3. the random numbers used by Vic to generate his challenges.

Hence, a transcript T for the above interactive proof of **Graph Isomorphism** would have the following form:

$$T = \left((G_1, G_2); (H_1, i_1, \rho_1); \ldots; (H_n, i_n, \rho_n)\right).$$

The essential point, which forms the basis for the formal definition of zero-knowledge proof, is that Vic (or anyone else) can forge transcripts — without participating in the interactive proof — that "look like" real transcripts. This can be done provided that the input graphs G_1 and G_2 are isomorphic. Forging is accomplished by means of the algorithm presented in Figure 13.6. The forging algorithm is a polynomial-time probabilistic algorithm. In the vernacular of zero-knowledge proofs, a forging algorithm is often called a *simulator*.

The fact that a simulator can forge transcripts has a very important consequence. Anything that Vic (or anyone else) can compute from the transcript could also be computed from a forged transcript. Hence, participating in the proof system does not increase Vic's ability to perform any computation; and in particular, it does not enable Vic himself to "prove" that G_1 and G_2 are isomorphic. Moreover, Vic cannot subsequently convince someone else that G_1 and G_2 are isomorphic by showing them the transcript T, since there is no way to distinguish a legitimate transcript from one that has been forged.

We still have to make precise the idea that a forged transcript "looks like" a real one. We give a rigorous definition in terms of probability distributions.

FIGURE 13.6
Forging algorithm for transcripts for Graph Isomorphism

Input: two isomorphic graphs G_1 and G_2, each having vertex set $\{1, \ldots, n\}$
1. $T = (G_1, G_2)$
2. **for** $j = 1$ **to** n **do**
3. Choose $i_j = 1$ or 2 at random;
4. Choose ρ_j to be a random permutation of $\{1, \ldots, n\}$;
5. Compute H_j to be the image of G_{i_j} under ρ_j;
6. concatenate (H_j, i_j, ρ_j) onto the end of T

DEFINITION 13.1 *Suppose that we have a polynomial-time interactive proof system for a decision problem* Π, *and a polynomial-time simulator* S. *Denote the set of all possible transcripts that could be produced as a result of Peggy and Vic carrying out the interactive proof with a yes-instance* x *by* $\mathcal{T}(x)$, *and denote the the set of all possible forged transcripts that could be produced by* S *by* $\mathcal{F}(x)$. *For any transcript* $T \in \mathcal{T}(x)$, *let* $p_{\mathcal{T}}(T)$ *denote the probability that* T *is the transcript produced from the interactive proof. Similarly, for* $T \in \mathcal{F}(x)$, *let* $p_{\mathcal{F}}(T)$ *denote the probability that* T *is the (forged) transcript produced by* S. *Suppose that* $\mathcal{T}(x) = \mathcal{F}(x)$, *and for any* $T \in \mathcal{T}(x)$, *suppose that* $p_{\mathcal{T}}(T) = p_{\mathcal{F}}(T)$. *(In other words, the set of real transcripts is identical to the set of forged transcripts, and the two probability distributions are identical.) Then we define the interactive proof system to be* ***perfect zero-knowledge for Vic.***

Of course we can define zero-knowledge however we like. But it is important that the definition captures our intuitive concept of what "zero-knowledge" should mean. We are saying that an interactive proof system is zero-knowlege for Vic if there exists a simulator that produces transcripts with an identical probability distribution to those produced when Vic actually takes part in the protocol. (This is a related but stronger concept than that of indistinguishable probability distributions that we studied in Chapter 12.) We have observed that a transcript contains all the information gained by Vic by taking part in the protocol. So it should seem reasonable to say that whatever Vic might be able to do after taking part in the protocol he could equally well do by just using the simulator to generate a forged transcript. We are perhaps not defining "knowledge" by this approach; but whatever "knowledge" might be, Vic doesn't gain any!

We will now prove that the interactive proof system for **Graph Isomorphism** is perfect zero-knowledge for Vic.

THEOREM 13.1
The interactive proof system for **Graph Isomorphism** *is perfect zero-knowledge for Vic.*

PROOF Suppose that G_1 and G_2 are isomorphic graphs on n vertices. A transcript T (real or forged) contains n triples of the form (H, i, ρ), where $i = 1$ or 2, ρ is a permutation of $\{1, \ldots, n\}$, and H is the image of G_i under the permutation ρ. Call such a triple a *valid* triple and denote by $\mathcal{R}$ the set of all valid triples. We begin by computing $|\mathcal{R}|$, the number of valid triples. Evidently $|\mathcal{R}| = 2 \times n!$ since each choice of i and ρ determines a unique graph H.

In any given round, say j, of the forging algorithm, it is clear that each valid triple (H, i, ρ) occurs with equal probability $1/(2 \times n!)$. What is the probability that the valid triple (H, i, ρ) is the jth triple on a real transcript? In the interactive proof system, Peggy first chooses a random permutation π and then computes H to be the image of G_1 under π. The permutation ρ is defined to be π if $i = 1$, and it is defined to be the composition of the two permutations π and σ if $i = 2$.

We are assuming that the value of i is chosen at random by Vic. If $i = 1$, then all $n!$ permutations ρ are equiprobable, since $\rho = \pi$ in this case and π was chosen to be a random permutation. On the other hand, if $i = 2$, then $\rho = \pi \circ \sigma$, where π is random and σ is fixed. In this case as well, every possible permutation ρ is equally probable. Now, since the two cases $i = 1$ and 2 are equally probable, and each permutation ρ is equally probable (independent of the value of i), and since i and ρ together determine H, it follows that all triples in $\mathcal{R}$ are equally likely.

Since a transcript consists of the concatenation of n independent random triples, it follows that

$$p_{\mathcal{T}}(T) = p_{\mathcal{F}}(T) = \frac{1}{(2 \times n!)^n}$$

for every possible transcript T. ∎

The proof of Theorem 13.1 assumes that Vic follows the protocol when he takes part in the interactive proof system. The situation is much more subtle if Vic does not follow the protocol. Is it true that an interactive proof remains zero-knowledge even if Vic deviates from the protocol?

In the case of **Graph Isomorphism**, the only way that Vic can deviate from the protocol is to choose his challenges i in a non-random way. Intuitively, it seems that this does not provide Vic with any "knowledge." However, transcripts produced by the simulator will not "look like" transcripts produced by Vic if he deviates from the protocol. For example, suppose Vic chooses $i = 1$ in every round of the proof. Then a transcript of the interactive proof will have $i_j = 1$ for $1 \leq j \leq n$; whereas a transcript produced by the simulator will have $i_j = 1$ for $1 \leq j \leq n$ only with probability 2^{-n}.

The way around this difficulty is to show that, no matter how a "cheating" Vic deviates from the protocol, there exists a polynomial-time simulator that will

produce forged transcripts that "look like" the transcripts produced by Peggy and (the cheating) Vic during the interactive proof. As before, the phrase "looks like" is formalized by saying that two probability distributions are identical.

Here is a more formal definition.

DEFINITION 13.2 *Suppose that we have a polynomial-time interactive proof system for a given decision problem Π. Let V^* be any polynomial-time probabilistic algorithm that (a possibly cheating) verifier uses to generate his challenges. (That is, V^* represents either an honest or cheating verifier.) Denote the set of all possible transcripts that could be produced as a result of Peggy and V^* carrying out the interactive proof with a yes-instance x of Π by $\mathcal{T}(V^*, x)$. Suppose that, for every such V^*, there exists an expected polynomial-time probabilistic algorithm $S^* = S^*(V^*)$ (the simulator) which will produce a forged transcript. Denote the set of possible forged transcripts by $\mathcal{F}(V^*, x)$. For any transcript $T \in \mathcal{T}(V^*, x)$, let $p_{\mathcal{T}}(T)$ denote the probability that T is the transcript produced by V^* taking part in the interactive proof. Similarly, for $T \in \mathcal{F}(x)$, let $p_{\mathcal{F}}(T)$ denote the probability that T is the (forged) transcript produced by S^*. Suppose that $\mathcal{T}(V^*, x) = \mathcal{F}(V^*, x)$, and for any $T \in \mathcal{T}(V^*, x)$, suppose that $p_{\mathcal{F},V^*}(T) = p_{\mathcal{T},V^*}(T)$. Then the interactive proof system is said to be* ***perfect zero-knowledge*** *(without qualification).*

In the special case where V^* is the same as Vic (i.e., when Vic is honest), the above definition is exactly the same as what we defined as "perfect zero-knowledge for Vic."

In order to prove that a proof system is perfect zero-knowledge, we need a generic transformation which will construct a simulator S^* from any V^*. We proceed to do this for the proof system for **Graph Isomorphism**. The simulator will play the part of Peggy, using V^* as a "restartable subroutine." Informally, S^* tries to guess the challenge i_j that V^* will make in each round j. That is, S^* generates a random valid triple of the form (H_j, i_j, ρ_j), and then executes the algorithm V^* to see what its challenge is for round j. If the guess i_j is the same as the challenge i'_j (as produced by V^*), then the triple (H_j, i_j, ρ_j) is appended to the forged transcript. If not, then this triple is discarded, S^* guesses a new challenge i_j, and the algorithm V^* is restarted after resetting its "state" to the way it was at the beginning of the current round. By the term "state" we mean the values of all variables used by the algorithm.

We now give a more detailed description of the simulation algorithm S^*. At any given time during the execution of the program V^*, the current state of V^* will be denoted by state(V^*). A pseudo-code description of the simulation algorithm is given in Figure 13.7.

It is possible that the simulator will run forever, if it never happens that $i_j = i'_j$. However, we can show that the average running time of the simulator is polynomial, and that the two probability distributions $p_{\mathcal{F},V^*}(T)$ and $p_{\mathcal{T},V^*}(T)$ are identical.

FIGURE 13.7
Forging algorithm for V^* for transcripts for Graph Isomorphism

```
    Input: two isomorphic graphs G_1 and G_2, each having vertex
    set {1, ..., n}
1.  T = (G_1, G_2)
2.  for j = 1 to n do
3.      define oldstate = state(V*)
4.      repeat
5.          Choose i_j = 1 or 2 at random
6.          Choose ρ_j to be a random permutation of {1, ..., n}
7.          Compute H_j to be the image of G_i under ρ
8.          call V* with input H_j, obtaining a challenge i'_j
9.          if i_j = i'_j then
                concatenate (H_j, i_j, ρ_j) onto the end of T
            else
                reset V* by defining state(V*) = oldstate
10.     until i_j = i'_j
```

THEOREM 13.2

The interactive proof system for **Graph Isomorphism** *is perfect zero-knowledge.*

PROOF First, we observe that, regardless of how V^* generates its challenges, the probability that the guess i'_j of S^* is the same as the challenge i_j is $1/2$. Hence, on average, S^* will generate two triples for every triple that it concatenates to the forged transcript. Hence, the average running time is polynomial in n.

The more difficult task is to show that the two probability distributions $p_{\mathcal{F},V^*}(T)$ and $p_{\mathcal{T},V^*}(T)$ are identical. In Theorem 13.1, where Vic was honest, we were able to compute the two probability distributions and see that they were identical. We also used the fact that triples (H, i, ρ) generated in different rounds of the proof are independent. However, in the current setting, we have no way of explicitly computing the two probability distributions. Further, triples generated in different rounds of the proof need not be independent. For example, the challenge that V^* presents in round j may depend in some very complicated way on challenges from previous rounds and on the way Peggy replied to those challenges.

The way to handle these difficulties is to look at the probability distributions on the possible partial transcripts during the course of the simulation or interactive proof, and proceed by induction on the number of rounds. For $0 \leq j \leq n$, we

define probability distributions $p_{\mathcal{T},V^*,j}$ and $p_{\mathcal{F},V^*,j}$ on the set of partial transcripts $\mathcal{T}_j$ that could occur at the end of round j. Notice that $p_{\mathcal{T},V^*,n} = p_{\mathcal{T},V^*}$ and $p_{\mathcal{F},V^*,n} = p_{\mathcal{F},V^*}$. Hence, if we can show that the two distributions $p_{\mathcal{T},V^*,j}$ and $p_{\mathcal{F},V^*,j}$ are identical for all j, then we will be done.

The case $j = 0$ corresponds to the beginning of the algorithm; at this point the transcript contains only the two graphs G_1 and G_2. Hence, the probability distributions are identical when $j = 0$. We use this for the start of the induction.

We make an inductive hypothesis that the two probability distributions $p_{\mathcal{T},V^*,j-1}$ and $p_{\mathcal{F},V^*,j-1}$ on $\mathcal{T}_{j-1}$ are identical, for some $j \geq 1$. We now prove that the two probability distributions $p_{\mathcal{T},V^*,j}$ and $p_{\mathcal{F},V^*,j}$ on $\mathcal{T}_j$ are identical.

Consider what happens during round j of the interactive proof. The probability that V^*'s challenge $i_j = 1$ is some real number p_1 and the probability that his challenge $i_j = 2$ is $1 - p_1$, where p_1 depends on the state of the algorithm V^* at the beginning of round j. We noted earlier that in the interactive proof, all possible graphs H are chosen by Peggy with equal probability. As well, any permutation ρ occurs with equal probability, independent of the value of p_1, since all permutations are equally likely for either possible challenge i_j. Hence, the probability that the jth triple on the transcript is (H, i, ρ) is $p_1/n!$ if $i = 1$, and $(1 - p_1)/n!$ if $i = 2$.

Next, let's do a similar analysis for the simulation. In any given iteration of the **repeat** loop, S^* will choose any graph H with probability $1/n!$. The probability that $i = 1$ and V^*'s challenge is 1 is $p_1/2$; and the probability that $i = 2$ and V^*'s challenge is 2 is $(1 - p_1)/2$. In each of these situations, (H, i, ρ) is written as the jth triple on the transcript. With probability $1/2$, nothing is written on the tape during any given iteration of the **repeat** loop.

Let us first consider the case $i = 1$. As mentioned above, the probability that V^*'s challenge is 1 is p_1. The probability that a triple $(H, 1, \rho)$ is written as the jth triple on the transcript during the ℓth iteration of the **repeat** loop is

$$\frac{p_1}{2^\ell \times n!}.$$

Hence, the probability that $(H, 1, \rho)$ is the jth triple on the transcript is

$$\frac{p_1}{2 \times n!}\left(1 + \frac{1}{2} + \frac{1}{4} + \ldots\right) = \frac{p_1}{n!}.$$

The case $i = 2$ is analyzed in a similar fashion: the probability that $(H, 2, \rho)$ is written as the jth triple on the transcript is $(1 - p_1)/n!$

Hence, the two probability distributions on the partial transcripts at the end of round j are identical. By induction, the two probability distributions $p_{\mathcal{F},V^*}(T)$ and $p_{\mathcal{T},V^*}(T)$ are identical, and the proof is complete. ∎

It is interesting also to look at the interactive proof system for **Graph Non-isomorphism**. It is not too difficult to prove that this proof is perfect zero-knowledge if Vic follows the protocol (i.e., if Vic chooses each challenge graph

FIGURE 13.8
A perfect zero-knowledge interactive proof system for Quadratic Residues

Input: an integer n with unknown factorization $n = pq$, where p and q are prime, and $x \in \mathrm{QR}(n)$

1. Repeat the following steps $\log_2 n$ times:
2. Peggy chooses a random $v \in \mathbb{Z}_n^*$, and computes
$$y = v^2 \bmod n.$$
Peggy sends y to Vic.
3. Vic chooses a random integer $i = 0$ or 1 and sends it to Peggy.
4. Peggy computes
$$z = u^i v \bmod n,$$
where u is a square root of x, and sends z to Vic.
5. Vic checks to see if
$$z^2 \equiv x^i y \pmod{n}.$$
6. Vic accepts Peggy's proof if the computation of step 5 is verified in each of the $\log_2 n$ rounds.

to be a random isomorphic copy of G_i where $i = 1$ or 2 is chosen at random). Further, provided that Vic constructs each challenge graph by taking an isomorphic copy of either G_1 or G_2, the protocol remains zero-knowledge even if Vic chooses his challenges in a non-random fashion. However, suppose that our ubiquitous troublemaker, Oscar, gives a graph H to Vic which is isomorphic to one of G_1 or G_2, but Vic does not know which G_i is isomorphic to H. If Vic uses this H as one of his challenge graphs in the interactive proof system, then Peggy will give Vic an isomorphism he didn't previously know, and (possibly) couldn't figure out for himself. In this situation, the proof system is (intuitively) not zero-knowledge, and it does not seem likely that a transcript could be forged by a simulator.

It is possible to alter the proof of **Graph Non-isomorphism** so it is perfect zero-knowledge, but we will not go into the details.

We now present some other examples of perfect zero-knowledge proofs. A perfect zero-knowledge proof for **Quadratic Residues** (modulo $n = pq$, where p and q are prime) is given in Figure 13.8. Peggy is proving that x is a quadratic residue. In each round, she generates a random quadratic residue y and sends it to Vic. Then, depending on Vic's challenge, Peggy either gives Vic a square root

FIGURE 13.9
Subgroup Membership

Problem Instance Two positive integers n and ℓ, and two distinct elements $\alpha, \beta \in \mathbb{Z}_n^*$, where α has order ℓ in $\mathbb{Z}_n^*$.
Question Is $\beta = \alpha^k$ for some integer k such that $0 \leq k \leq \ell - 1$? (In other words, is β a member of the subgroup of $\mathbb{Z}_n^*$ generated by α?)

of y or a square root of xy.

It is clear that the protocol is complete. To prove soundness, observe that if x is not a quadratic residue, then Peggy can answer only one of the two possible challenges since, in this case, y is a quadratic residue if and only if xy is not a quadratic residue. So Peggy will be caught in any given round of the protocol with probability $1/2$, and her probability of deceiving Vic in all n rounds is only $2^{-\log_2 n} = 1/n$. (The reason for having $\log_2 n$ rounds is that the size of the problem instance is proportional to the number of bits in the binary representation of n, which is $\log_2 n$. Hence, the deception probability for Peggy is exponentially small as a function of the size of the problem instance, as in the zero-knowledge proof for **Graph Isomorphism**.)

Perfect zero-knowledge for Vic can be shown in a similar manner as was done for **Graph Isomorphism**. Vic can generate a triple (y, i, z) by first choosing i and z, and then defining

$$y = z^2(x^i)^{-1} \bmod n.$$

Triples generated in this fashion have exactly the same probability distribution as those generated during the protocol, assuming Vic chooses his challenges at random. Perfect zero-knowledge (for an arbitrary V^*) is proved by following the same strategy as for **Graph Isomorphism**. It requires building a simulator S^* that guesses V^*'s challenges and keeps only the triples where the guesses are correct.

As a further illustration, we present one more example of a perfect zero-knowledge proof, this one for a decision problem related to the **Discrete Logarithm** problem. The problem, which we call **Subgroup Membership**, is defined in Figure 13.9. Of course, the integer k (if it exists) is just the discrete logarithm of β.

We present a perfect zero-knowledge proof for **Subgroup Membership** in Figure 13.10. The analysis of this protocol is similar to the others that we have looked at; the details are left to the reader.

FIGURE 13.10
A perfect zero-knowledge interactive proof system for Subgroup Membership

Input: A positive integer n, and two distinct elements $\alpha, \beta \in \mathbb{Z}_n^*$, where the order of α is denoted by ℓ and is publicly known

1. Repeat the following steps $\log_2 n$ times:
2. Peggy chooses a random j such that $0 \leq j \leq \ell - 1$, and computes
$$\gamma = \alpha^j \bmod n.$$
Peggy sends γ to Vic.
3. Vic chooses a random integer $i = 0$ or 1 and sends it to Peggy.
4. Peggy computes
$$h = j + ik \bmod \ell,$$
where
$$k = \log_\alpha \beta,$$
and sends h to Vic.
5. Vic checks to see if
$$\alpha^h \equiv \beta^i \gamma \pmod n.$$
6. Vic accepts Peggy's proof if the computation of step 5 is verified in each of the $\log_2 n$ rounds.

13.3 Bit Commitments

The zero-knowledge proof system for **Graph Isomorphism** is interesting, but it would be more useful to have zero-knowledge proof systems for problems that are known to be NP-complete. There is theoretical evidence that perfect zero-knowledge proofs do not exist for NP-complete problems. However, we can describe proof systems that attain a slightly weaker form of zero-knowledge called *computational* zero-knowledge. The actual proof systems are described in the next section; in this section we describe the technique of bit commitment that is an essential tool used in the proof system.

Suppose Peggy writes a message on a piece of paper, and then places the message in a safe for which she knows the combination. Peggy then gives the

safe to Vic. Even though Vic doesn't know what the message is until the safe is opened, we would agree that Peggy is *committed* to her message because she cannot change it. Further, Vic cannot learn what the message is (assuming he doesn't know the combination of the safe) unless Peggy opens the safe for him. (Recall that we used a similar analogy in Chapter 4 to describe the idea of a public-key cryptosystem, but in that case, it was the recipient of the message, Vic, who could open the safe.)

Suppose the message is a bit $b = 0$ or 1, and Peggy encrypts b in some way. The encrypted form of b is sometimes called a *blob* and the encryption method is called a *bit commitment scheme*. In general, a bit commitment scheme will be a function $f : \{0,1\} \times X \to Y$, where X and Y are finite sets. An encryption of b is any value $f(b,x)$, $x \in X$. We can informally define two properties that a bit commitment scheme should satisfy:

concealing

For a bit $b = 0$ or 1, Vic cannot determine the value of b from the blob $f(b,x)$.

binding

Peggy can later "open" the blob, by revealing the value of x used to encrypt b, to convince Vic that b was the value encrypted. Peggy should not be able to open a blob as both a 0 and a 1.

If Peggy wants to commit any bitstring, she simply commits every bit independently.

One way to perform bit commitment is to use the **Goldwasser-Micali Probabilistic Cryptosystem** described in Section 12.4. Recall that in this system, $n = pq$, where p and q are primes, and $m \in \tilde{\mathrm{QR}}(n)$. The integers n and m are public; the factorization $n = pq$ is known only to Peggy. In our bit commitment scheme, we have $X = Y = \mathbb{Z}_n{}^*$ and

$$f(b,x) = m^b x^2 \bmod n.$$

Peggy encrypts a value b by choosing a random x and computing $y = f(b,x)$; the value y comprises the blob.

Later, when Peggy wants to open y, she reveals the values b and x. Then Vic can verify that

$$y \equiv m^b x^2 \pmod{n}.$$

Let us think about the concealing and binding properties. A blob is an encryption of 0 or of 1, and reveals no information about the plaintext value x provided that the **Quadratic Residues** problem is infeasible (we discussed this at length in Chapter 12). Hence, the scheme is concealing.

Is the scheme binding? Let us suppose not; then

$$m{x_1}^2 \equiv {x_2}^2 \pmod{n}$$

for some $x_1, x_2 \in \mathbb{Z}_n^*$. But then

$$m \equiv (x_2 x_1^{-1})^2 \pmod{n},$$

which is a contradiction since $m \in \tilde{\mathrm{QR}}(n)$.

We will be using bit commitment schemes to construct zero-knowledge proofs. However, they have another nice application, to the problem of *coin-flipping by telephone*. Suppose Alice and Bob want to make some decision based on a random coin flip, but they are not in the same place. This means that it is impossible for one of them to flip a real coin and have the other verify it. A bit commitment scheme provides a way out of this dilemma. One of them, say Alice, chooses a random bit b, and computes a blob, y. She gives y to Bob. Now Bob guesses the value of b, and then Alice opens the blob to reveal b. The concealing property means that it is infeasible for Bob to compute b given y, and the binding property means that Alice can't "change her mind" after Bob reveals his guess.

We now give another example of a bit commitment scheme, this time based on the **Discrete Logarithm** problem. Recall from Section 5.1.2 that if $p \equiv 3 \pmod 4$ is a prime such that the **Discrete Logarithm** problem in $\mathbb{Z}_p^*$ is infeasible, then the second least significant bit of a discrete logarithm is secure. Actually, it has been proved for primes $p \equiv 3 \pmod 4$ that any Monte Carlo algorithm for the **Second Bit** problem having error probability $1/2 - \epsilon$ with $\epsilon > 0$ can be used to solve the **Discrete Log** problem in $\mathbb{Z}_p^*$. This much stronger result is the basis for the bit commitment scheme.

This bit commitment scheme will have $X = \{1, \ldots, p-1\}$ and $Y = \mathbb{Z}_p^*$. The second least significant bit of an integer x, denoted by $\mathrm{SLB}(x)$, is defined as follows:

$$\mathrm{SLB}(x) = \begin{cases} 0 & \text{if } x \equiv 0, 1 \pmod 4 \\ 1 & \text{if } x \equiv 2, 3 \pmod 4. \end{cases}$$

The bit commitment scheme f is defined by

$$f(b, x) = \begin{cases} \alpha^x \bmod p & \text{if } \mathrm{SLB}(x) = b \\ \alpha^{p-x} \bmod p & \text{if } \mathrm{SLB}(x) \neq b. \end{cases}$$

In other words, a bit b is encrypted by choosing a random element having second last bit b, and raising α to that power modulo p. (Note that $\mathrm{SLB}(p-x) \neq \mathrm{SLB}(x)$ since $p \equiv 3 \pmod 4$.)

The scheme is binding, and by the remarks made above, it is concealing provided that the **Discrete Logarithm** problem in $\mathbb{Z}_p^*$ is infeasible.

13.4 Computational Zero-knowledge Proofs

In this section, we give a zero-knowledge proof system for the NP-complete decision problem **Graph 3-Colorability**, which is defined in Figure 13.11. The

FIGURE 13.11
Graph 3-Colorability

Problem Instance A graph $G = (V, E)$ on n vertices,
Question Is there a *proper 3-coloring* of G? (In mathematical terms, is there a function $\phi : V(G) \rightarrow \{1, 2, 3\}$ such that $\{u, v\} \in E$ implies $\phi(u) \neq \phi(v)$?)

proof system uses a bit commitment scheme; to be specific, we will employ the bit commitment scheme presented in Section 13.3 that is based on probabilistic encryption. We assume that Peggy knows a 3-coloring ϕ of a graph G, and she wants to convince Vic that G is 3-colorable in a zero-knowledge fashion. Without loss of generality, we assume that G has vertex set $V = \{1, \ldots, n\}$. Denote $m = |E|$. The proof system will be described in terms of a commitment scheme $f : \{0, 1\} \times X \rightarrow Y$ which is made public. Since we want to encrypt a color rather than a bit, we will replace the color 1 by the two bits 01, the color 2 by 10 and the color 3 by 11. Then we encrypt each of the two bits representing the color by using f.

The interactive proof system is presented in Figure 13.12. Informally, what happens is the following. In each round, Peggy commits a coloring that is a permutation of the fixed coloring ϕ. Vic requests that Peggy open the blobs corresponding to the endpoints of some randomly chosen edge. Peggy does so, and then Vic checks that the commitments are as claimed and that the two colors are different. Notice that all Vic's computations are polynomial-time, and so are Peggy's, provided that she knows the existence of one 3-coloring ϕ.

Here is a very small example to illustrate.

Example 13.3
Suppose G is the graph (V, E), where

$$V = \{1, 2, 3, 4, 5\}$$

and

$$E = \{12, 14, 15, 23, 34, 45\}.$$

Suppose that Peggy knows the 3-coloring ϕ where $\phi(1) = 1$, $\phi(2) = \phi(4) = 2$ and $\phi(3) = \phi(5) = 3$. Suppose also that the parameters of the bit commitmtent scheme are $n = 321389$ and $m = 156897$, so $f(b, x) = m^b x^2 \bmod n$, where $b = 0, 1$ and $x \in \mathbb{Z}_n{}^*$.

Suppose that Peggy chooses the permutation $\pi = (1\ 3\ 2)$ in some round of the

FIGURE 13.12
A computational zero-knowledge interactive proof system for Graph 3-colorability

Input: a graph $G = (E, V)$ on vertex set $\{1, \ldots, n\}$

1. Repeat the following steps m^2 times:
2. Let ϕ be a 3-coloring of G. Peggy chooses a random permutation π of $\{1, 2, 3\}$. For $1 \leq i \leq n$, she defines

$$c_i = \pi(\phi(i)),$$

and writes c_i as a bitstring of length two:

$$c_i = c_{i,1}c_{i,2}.$$

Then, for $1 \leq i \leq n$, she chooses two random elements $r_{i,1}, r_{i,2} \in X$, and computes

$$R_{i,j} = f(c_{i,j}, r_{i,j}),$$

$j = 1, 2$. She sends the list

$$(R_{1,1}, R_{1,2}, \ldots, R_{n,1}, R_{n,2})$$

to Vic.

3. Vic chooses a random edge $\{u, v\} \in E$ and sends it to Peggy.
4. Peggy sends $(c_{u,1}, c_{u,2}, r_{u,1}, r_{u,2})$ and $(c_{v,1}, c_{v,2}, r_{v,1}, r_{v,2})$ to Vic.
5. Vic checks that

$$(c_{u,1}, c_{u,2}) \neq (c_{v,1}, c_{v,2}),$$
$$(c_{u,1}, c_{u,2}) \neq (0, 0),$$
$$(c_{v,1}, c_{v,2}) \neq (0, 0),$$
$$R_{u,j} = f(c_{u,j}, r_{u,j}), j = 1, 2, \text{ and}$$
$$R_{v,j} = f(c_{v,j}, r_{v,j}), j = 1, 2.$$

6. Vic accepts Peggy's proof if the computation of step 5 is verified in each of the m^2 rounds.

proof. Then she computes:

$$c_1 = 1$$
$$c_2 = 3$$
$$c_3 = 2$$
$$c_4 = 3$$
$$c_5 = 2.$$

She will encode this coloring in binary as the 10-tuple

$$0111101110$$

and then compute commitments of these ten bits. Suppose that she does this as follows:

b	x	$f(b,x)$
0	147658	176593
1	318856	205585
1	14497	189102
1	285764	294039
1	128589	230968
0	228569	77477
1	53369	305090
1	194634	276484
1	202445	292707
0	177561	290599

Then Peggy gives Vic the ten values $f(b,x)$ computed above.

Next, suppose that Vic chooses the edge 34 as his challenge. Then Peggy opens four blobs: the two that correspond to vertex 3 and the two that correspond to vertex 4. So Peggy gives Vic the ordered pairs

$$(b,x) = (1,128589),(0,228569),(1,53369),(1,194634).$$

Vic will first check that the two colors are distinct: 10 encodes color 2 and 11 encodes color 3, so this is all right. Next, Vic verifies that the four commitments are valid and hence this round of the proof is completed successfully. □

As in previous proof systems we have studied, Vic will accept a valid proof with probability 1, so we have completeness. What is the probability that Vic will accept if G is not 3-colorable? In this case, for any coloring, there must be at least one edge ij such that i and j have the same color. Vic's chances of choosing such an edge are at least $1/m$. Peggy's probability of fooling Vic in all m^2 rounds is at

most

$$\left(1 - \frac{1}{m}\right)^{m^2}.$$

Since $(1 - 1/m)^m \to e^{-1}$ as $m \to \infty$, we see that $(1 - 1/m)^{m^2} \to e^{-m}$, which approaches zero exponentially quickly as a function of $m = |E|$. Hence, we have soundness as well.

Let's now turn to the zero-knowledge aspect of the proof system. All that Vic sees in any given round of the protocol is an encrypted 3-colouring of G, together with the two distinct colours of the endpoints of one particular edge, as previously committed by Peggy. Since the colors are permuted in each round, it seems that Vic cannot combine information from different rounds to reconstruct the 3-coloring.

The proof system is not perfect zero-knowledge, but it does provide a weaker form of zero-knowledge called *computational zero-knowledge*. Computational zero-knowledge is defined exactly as perfect zero-knowledge, except that the relevant probability distributions of transcripts are required only to be polynomially indistinguishable (in the sense of Chapter 12) rather than identical.

We begin by showing how transcripts can be forged. We give an explicit algorithm that will forge transcripts that cannot be distinguished from those produced by an honest Vic. If Vic deviates from the protocol, then it is possible to construct a simulator which uses the algorithm V^* as a restartable subroutine to construct forged transcripts. Both forging algorithms follow the pattern of the related algorithms for the **Graph Isomorphism** proof system.

Here, we consider only the case where Vic follows the protocol. A transcript T for the interactive proof of **Graph 3-colorability** would have the form

$$(G; A_1; \ldots; A_{m^2}),$$

where A_j consists of $2n$ blobs computed by Peggy, the edge uv chosen by Vic, the colors assigned by Peggy in round j to u and v, and the four random numbers used by Peggy to encrypt the colors of these two vertices. A transcript is forged by means of the forging algorithm presented in Figure 13.13.

Proving (computational) zero-knowledge for Vic requires showing that the two probability distributions on transcripts (as produced by the Vic taking part in the protocol, and as produced by the simulator) are indistinguishable. We will not do this here, but we will make a couple of comments. Notice that the two probability distributions are not identical. This is because virtually all the R_{ji}'s in a forged transcript are blobs encrypting 1; whereas the R_{ji}'s on a real transcript will (usually) be encryptions of more equal numbers of 0's and 1's. However, it is possible to show that the two probability distributions cannot be distinguished in polynomial time, provided that the underlying bit commitment scheme is secure. More precisely, this means that the probability distribution on blobs encrypting color c are indistinguishable from the probability distribution on blobs encrypting

FIGURE 13.13
Forging algorithm for transcripts for Graph 3-colorability

Input: a graph $G = (V, E)$ having vertex set $V = \{1, \ldots, n\}$

1. $T = (G)$
2. **for** $j = 1$ **to** m^2 **do**
3. Choose an edge $\{u, v\} \in E$ at random
4. Choose $d = d_1 d_2$ and $e = e_1 e_2$ to be random, distinct colors, where $d_1, d_2, e_1, e_2 \in \{0, 1\}$
5. Choose $r_{i,j}$ to be a random element of X, for $1 \leq i \leq n$, $j = 1, 2$
6. For $1 \leq i \leq n$ and $j = 1, 2$, define

$$R_{i,j} = \begin{cases} f(1, r_{i,j}) & \text{if } i \neq u, v \\ f(d_j, r_{i,j}) & \text{if } i = u \\ f(e_j, r_{i,j}) & \text{if } i = v. \end{cases}$$

7. concatenate

$$(R_{1,1}, \ldots, R_{n,2}, u, v, d_1, d_2, r_{d,1}, r_{d,2}, e_1, e_2, r_{e,1}, r_{e,2})$$

onto the end of T.

color d if $c \neq d$.

Readers familiar with NP-completeness theory will realize that, having given a zero-knowledge proof for one particular NP-complete problem, we can obtain a zero-knowledge proof for any other NP-complete problem. This can be done by applying a polynomial transformation from a given NP-complete problem to the **Graph 3-coloring** problem.

13.5 Zero-knowledge Arguments

Let us recap the basic properties of the computational zero-knowledge proof for **Graph 3-colorability** presented in the last section. No assumptions are needed to prove completeness and soundness of the protocol. A computational assumption is needed to prove zero-knowledge, namely that the underlying bit commitment scheme is secure. Observe that if Peggy and Vic take part in the protocol, then Vic may later try to break the bit commitment scheme that was used in the protocol (for

example, if the scheme based on quadratic residuosity were used, then Vic would try to factor the modulus). If at any future time Vic can break the bit commitment scheme, then he can decrypt the blobs used by Peggy in the protocol and extract the 3-coloring.

This analysis depends on the properties of the blobs that were used in the protocol. Although the binding property of the blobs is unconditional, the concealing property relies on a computational assumption.

An interesting variation is to use blobs in which the concealing property is unconditional but the binding property requires a computational assumption. This leads to a protocol that is known as a zero-knowledge *argument* rather than a zero-knowledge proof. The reader will recall that we have assumed up until now that Peggy is all-powerful; in a zero-knowledge argument we will assume that Peggy's computations are required to be polynomial-time. (In fact, this assumption creates no difficulties, for we have already observed that Peggy's computations are polynomial-time provided she knows one 3-coloring of G.)

Let us begin by describing a couple of bit commitment schemes of this type and then examine the ramifications of using them in the protocol for **Graph 3-coloring**.

The first scheme is (again) based on the **Quadratic Residues** problem. Suppose $n = pq$, where p and q are prime, and let $m \in \mathrm{QR}(n)$ (note that in the previous scheme m was a pseudo-square). In this scheme neither the factorization of n nor the square root of m should be known to Peggy. So either Vic should construct these values or they should be obtained from a (trusted) third party.

Let $X = \mathbb{Z}_n^*$ and $Y = \mathrm{QR}(n)$, and define

$$f(b, x) = m^b x^2 \bmod n.$$

As before, Peggy encrypts a value b by choosing a random x and computing the blob $y = f(b, x)$. In this scheme all the blobs are quadratic residues. Further, any $y \in \mathrm{QR}(n)$ is both an encryption of 0 and an encryption of 1. For suppose $y = x^2 \bmod n$ and $m = k^2 \bmod n$. Then

$$y = f(0, x) = f(1, xk^{-1} \bmod n).$$

This means that the concealing property is achieved unconditionally. On the other hand, what happens to the binding property? Peggy can open any given blob both as a 0 and as a 1 if and only if she can compute k, a square root of m. So, in order for the scheme to be (computationally) binding, we need to make the assumption that it is infeasible for Peggy to compute a square root of m. (If Peggy were all-powerful, then she could, of course, do this. This is one reason why we are now assuming that Peggy is computationally bounded.)

As a second bit commitment scheme of this type, we give an example of a scheme based on the **Discrete Logarithm** problem. Let p be a prime such that the discrete log problem in $\mathbb{Z}_p^*$ is infeasible, let α be a primitive element of $\mathbb{Z}_p^*$ and let $\beta \in \mathbb{Z}_p^*$. The value of β should be chosen by Vic, or by a trusted third party,

TABLE 13.1
Comparison of Properties of Proofs and Arguments

property	zero-knowledge proof	zero-knowledge argument
completeness	unconditional	unconditional
soundness	unconditional	computational
zero-knowledge	computational	perfect
concealing blobs	unconditional	computational
binding blobs	computational	unconditional

rather than by Peggy. This scheme will have $X = \{0, \ldots, p-1\}$, $Y = \mathbb{Z}_p{}^*$, and f is defined by

$$f(b, x) = \beta^b \alpha^x \bmod p.$$

It is not hard to see that this scheme is unconditionally concealing, and it is binding if and only if it is infeasible for Peggy to compute the discrete logarithm $\log_\alpha \beta$.

Now, suppose we use one of these two bit commitment schemes in the protocol for **Graph 3-colorability**. It is easy to see that the protocol remains complete. But now the soundness condition depends on a computational assumption: the protocol is sound if and only if the bit commitment scheme is binding. What happens to the zero-knowledge aspect of the protocol? Because the bit commitment scheme is unconditionally concealing, the protocol is now

perfect zero-knowledge rather than just computational zero-knowledge. Thus we have a perfect zero-knowledge argument.

Whether one prefers an argument to a proof depends on the application, and whether one wants to make a computational assumption regarding Peggy or Vic. A comparison of the properties of proofs and arguments is summarized in Table 13.1. In the column "zero-knowledge proof," the computational assumptions pertain to Peggy's computing power; in the column "zero-knowledge argument," the computational assumptions refer to Vic's computing power.

13.6 Notes and References

Most of the material in this chapter is based on Brassard, Chaum, and Crépeau [BCC88] and on Goldreich, Micali, and Wigderson [GMW91]. The bit commitment schemes we present, and a thorough discussion of the differences between proofs and arguments, can be found in [BCC88] (however, note that the term "argument" was first used in [BC90]). Zero-knowledge proofs for **Graph Isomorphism**, **Graph Non-isomorphism** and **Graph 3-colorability** can be found in [GMW91]. Another relevant paper is Goldwasser, Micali, and Rackoff [GMR89],

FIGURE 13.14
An interactive proof system for Quadratic Non-residues

Input: an integer n with unknown factorization $n = pq$, where p and q are prime, and $x \in \widetilde{\mathrm{QR}}(n)$

1. Repeat the following steps $\log_2 n$ times:
2. Vic chooses a random $v \in \mathbb{Z}_n{}^*$, and computes
$$y = v^2 \bmod n.$$
Vic chooses $i = 0$ or 1 at random, and he sends
$$z = x^i y \bmod n$$
to Peggy.
3. If $z \in \mathrm{QR}(n)$, then Peggy defines $j = 0$, otherwise she defines $j = 1$. Then she sends j to Vic.
4. Vic checks to see if $i = j$.
5. Vic accepts Peggy's proof if the computation of step 4 is verified in each of the $\log_2 n$ rounds.

in which interactive proof systems are first defined formally. The zero-knowledge proof for **Quadratic Residues** is from this paper.

The idea of coin-flipping by telephone is due to Blum [BL82].

A very informal and entertaining illustration of the concept of zero-knowledge is presented by Quisquater and Guillou [QG90]. Also, see Johnson [JO88] for a more mathematical survey of interactive proof systems.

Exercises

13.1 Consider the interactive proof system for the problem **Quadratic Non-residues** presented in Figure 13.14. Prove that the system is sound and complete, and explain why the protocol is not zero-knowledge.

13.2 Devise an interactive proof system for the problem **Subgroup Non-membership**. Prove that your protocol is sound and complete.

13.3 Consider the zero-knowledge proof for **Quadratic Residues** that was presented in Figure 13.8.

(a) Define a **valid triple** to be one having the form (y, i, z), where $y \in \mathrm{QR}(n)$, $i = 0$ or 1, $z \in \mathbb{Z}_n{}^*$ and $z^2 \equiv x^i y \pmod n$. Show that the number of valid triples is $2(p-1)(q-1)$, and each such triple is generated with equal probability if Peggy and Vic follow the protocol.

(b) Show that Vic can generate triples having the same probability distribution without knowing the factorization $n = pq$.

(c) Prove that the protocol is perfect zero-knowledge for Vic.

13.4 Consider the zero-knowledge proof for **Subgroup Membership** that was presented in Figure 13.10.

(a) Prove that the protocol is sound and complete.

(b) Define a **valid triple** to be one having the form (γ, i, h), where $\gamma \in \mathbb{Z}_n^*$, $i = 0$ or 1, $0 \leq h \leq \ell - 1$ and $\alpha^h \equiv \beta^i \gamma \pmod{n}$. Show that the number of valid triples is 2ℓ, and each such triple is generated with equal probability if Peggy and Vic follow the protocol.

(c) Show that Vic can generate triples having the same probability distribution without knowing the discrete logarithm $\log_\alpha \beta$.

(d) Prove that the protocol is perfect zero-knowledge for Vic.

13.5 Prove that the **Discrete Logarithm** bit commitment scheme presented in Section 13.5 is unconditionally concealing, and prove that it is binding if and only if Peggy cannot compute $\log_\alpha \beta$.

13.6 Suppose we use the **Quadratic Residues** bit commitment scheme presented in Section 13.5 to obtain a zero-knowledge argument for **Graph 3-coloring**. Using the forging algorithm presented in Figure 13.13, prove that this protocol is perfect zero-knowledge for Vic.

Further Reading

Other recommended textbooks and monographs on cryptography include the following: Beker and Piper [BP82], Beutelspacher [BE94], Brassard [BR88], Biham and Shamir [BS93], Denning [DE82], Kahn [KA67], Koblitz [KO87], Konheim [KO81], Kranakis [KR86], Menezes [ME93], Meyer and Matyas [MM82], Patterson [PA87], Pomerance [PO90A], Rueppel [RU86], Salomaa [SA90], Schneier [SC93], Seberry and Pieprzyk [SP89], Simmons [SI92B], van Tilborg [VT88], and Welsh [WE88].

The main research journals in cryptography are the *Journal of Cryptology* and *Designs, Codes and Cryptography*. The *Journal of Cryptology* is the journal of the International Association for Cryptologic Research (or IACR) which also sponsors the two main annual cryptology conferences, CRYPTO and EUROCRYPT.

CRYPTO has been held since 1981 in Santa Barabara. The proceedings of CRYPTO have been published annually since 1982: CRYPTO '82 [CRS83], CRYPTO '83 [CH84], CRYPTO '84 [BC85], CRYPTO '85 [WI86], CRYPTO '86 [OD87], CRYPTO '87 [PO88], CRYPTO '88 [GO90], CRYPTO '89 [BR90], CRYPTO '90 [MV91], CRYPTO '91 [FE92], CRYPTO '92 [BR93], CRYPTO '93 [ST94], and CRYPTO '94 [DE94]. EUROCRYPT has been held annually since 1982, and except for 1983 and 1986, its proceedings have been published, as follows: EUROCRYPT '82 [BE83], EUROCRYPT '84 [BCI85], EUROCRYPT '85 [PI86], EUROCRYPT '87 [CP88], EUROCRYPT '88 [GU88A], EUROCRYPT '89 [QV90], EUROCRYPT '90 [DA91], EUROCRYPT '91 [DA91A], EUROCRYPT '92 [RU93], and EUROCRYPT '93 [HE94].

A third conference series, AUSCRYPT/ASIACRYPT, has been held "in association with" the IACR. Its conference proceedings have also been published: AUSCRYPT '90 [SP90], ASIACRYPT '91 [IRM93], and AUSCRYPT '92 [SZ92].