Identification Schemes 283

9 Identification Schemes

9.1 Introduction

Cryptographic techniques enable many seemingly impossible problems to be solved. One such problem is the construction of secure identification schemes. There are many common, everyday situations where it is necessary to electronically "prove" one's identity. Some typical scenarios are as follows:

1. To withdraw money from an automated teller machine (or ATM), we use a card together with a four-digit personal identification number (PIN).
2. To charge purchases over the telephone to a credit card, all that is necessary is a credit card number (and the expiry date).
3. To charge long-distance telephone calls (using a calling card), one requires only a telephone number together with a four-digit PIN.
4. To do a remote login to a computer over a network, it suffices to know a valid user name and the corresponding password.

In practice, these types of schemes are not usually implemented in a secure way. In the protocols performed over the telephone, any eavesdropper can use the identifying information for their own purposes. This could include the person who is the recipient of the information; many credit card "scams" operate in this way. An ATM card is somewhat more secure, but there are still weaknesses. For example, someone monitoring the communication line can obtain all the information encoded on the card's magnetic strip, as well as the PIN. This could allow an imposter to gain access to a bank account. Finally, remote computer login is a serious problem due to the fact that user IDs and passwords are transmitted over the network in unencrypted form. Thus they are vulnerable to anyone who is monitoring the computer network.

The goal of an identification scheme is that someone "listening in" as Alice identifies herself to Bob, say, should not subsequently be able to misrepresent herself as Alice. Furthermore, we should try to guard against the possibility that

FIGURE 9.1
Challenge-and-response protocol

1. Bob chooses a *challenge*, x, which is a random 64-bit string. Bob sends x to Alice.
2. Alice computes
$$y = e_K(x)$$
and sends it to Bob.
2. Bob computes
$$y' = e_K(x)$$
and verifies that $y' = y$.

Bob himself might try to impersonate Alice after she has identified herself to him. In other words, Alice wants to be able to prove her identity electronically without "giving away" her identifying information.

Several such identification schemes have been discovered. One practical objective is to find a scheme that is simple enough that it can be implemented on a smart card, which is essentially a credit card equipped with a chip that can perform arithmetic computations. Hence, both the amount of computation and the memory requirements should be kept as small as possible. Such a card would be a more secure alternative to current ATM cards. However, it is important to note that the "extra" security pertains to someone monitoring the communication line. Since it is the card that is "proving" its identity, we have no extra protection against a lost card. It would still be necessary to include a PIN in order to establish that it is the real owner of the card who is initiating the identification protocol.

In later sections, we will describe some of the more popular identification schemes. But first, we give a very simple scheme that can be based on any private-key cryptosystem, e.g., **DES**. The protocol, which is described in Figure 9.1, is called a *challenge-and-response* protocol. In it, we assume that Alice is identifying herself to Bob, and Alice and Bob share a common secret key, K, which specifies an encryption function e_K.

We illustrate this protocol with a small example.

Example 9.1
Assume Alice and Bob use an encryption function which does a modular exponentiation:

$$e_K(x) = x^{101379} \bmod 167653.$$

Suppose Bob's challenge is $x = 77835$. Then Alice responds with $y = 100369$. $\square$

Virtually all identification schemes are challenge-and-response protocols, but the most useful schemes do not require shared keys. This idea will be pursued in the remainder of the chapter.

9.2 The Schnorr Identification Scheme

We begin by describing the **Schnorr Identification Scheme**, which is one of the most attractive practical identification schemes. The scheme requires a trusted authority, which we denote by TA. The TA will choose parameters for the scheme as follows:

1. p is a large prime (i.e., $p \geq 2^{512}$) such that the discrete log problem in $\mathbb{Z}_p{}^*$ is intractible.
2. q is a large prime divisor of $p - 1$ (i.e., $q \geq 2^{140}$).
3. $\alpha \in \mathbb{Z}_p{}^*$ has order q (such an α can be computed as the $(p-1)/q$th power of a primitive element).
4. A *security parameter* t such that $q > 2^t$. For most practical applications, $t = 40$ will provide adequate security.
5. The TA also establishes a secure signature scheme with a secret signing algorithm sig_{TA} and a public verification algorithm ver_{TA}.
6. A secure hash function is specified. As usual, all information is to be hashed before it is signed. In order to make the protocols easier to read, we will omit the hashing steps from the descriptions of the protocols.

The parameters p, q, and α, the public verification algorithm ver_{TA} and the hash function are all made public.

A certificate will be issued to Alice by the TA. When Alice wants to obtain a certificate from the TA, the steps in Figure 9.2 are carried out. At a later time, when Alice wants to prove her identity to Bob, say, the protocol of Figure 9.3 is executed.

As mentioned above, t is a security parameter. Its purpose is to prevent an impostor posing as Alice, say Olga, from guessing Bob's challenge, r. For, if Olga guessed the correct value of r, she could choose any value for y and compute

$$\gamma = \alpha^y v^r \bmod p.$$

She would give Bob γ in step 1, and then when she receives the challenge r, she would supply the value y she has already chosen. Then γ would be verified by

FIGURE 9.2
Issuing a certificate to Alice

1. The TA establishes Alice's identity by means of conventional forms of identification such as a birth certificate, passport, etc. Then the TA forms a string ID(Alice) which contains her identification information.
2. Alice secretly chooses a random exponent a, where $0 \leq a \leq q-1$. Alice computes
$$v = \alpha^{-a} \bmod p$$
and gives v to the TA.
3. The TA generates a signature
$$s = sig_{\mathrm{TA}}(I, v).$$
The certificate
$$\mathrm{C(Alice)} = (\mathrm{ID(Alice)}, v, s)$$
is given to Alice.

Bob in step 6.

The probability that Olga will guess the value of r correctly is 2^{-t} if r is chosen at random by Bob. Thus, $t = 40$ should be a reasonable value for most applications. (But notice that Bob should choose his challenge r at random every time Alice identifies herself to him. If Bob always used the same challenge r, then Olga could impersonate Alice by the method described above.)

Basically, there are two things happening in the verification protocol. First, the signature s proves the validity of Alice's certificate. Thus Bob verifies the signature of the TA on Alice's certificate to convince himself that the certificate itself is authentic. This is essentially the same way that certificates were used in Chapter 8.

The second part of the protocol concerns the secret number a. The value a functions like a PIN in that it convinces Bob that the person carrying out the identification protocol is indeed Alice. But there is an important difference from a PIN: in the identification protocol, the value of a is not revealed. Instead Alice (or more accurately, Alice's smart card) "proves" that she/it knows the value of a in step 5 of the protocol by computing the value y in response to the challenge r issued by Bob. Since the value of a is not revealed, this technique is called a *proof of knowledge*.

FIGURE 9.3
The Schnorr identification scheme

1. Alice chooses a random number k, where $0 \leq k \leq q-1$, and computes
$$\gamma = \alpha^k \bmod p.$$
2. Alice sends her certificate $C(\text{Alice}) = (\text{ID}(\text{Alice}), v, s)$ and γ to Bob.
3. Bob verifies the signature of the TA by checking that $ver_{\text{TA}}(\text{ID}(\text{Alice}), v, s) = \text{true}$.
4. Bob chooses a random number r, $1 \leq r \leq 2^t$ and gives it to Alice.
5. Alice computes
$$y = k + ar \bmod q$$
and gives y to Bob.
6. Bob verifies that
$$\gamma \equiv \alpha^y v^r \pmod p.$$

The following congruences demonstrate that Alice will be able to prove her identity to Bob:

$$\begin{aligned}\alpha^y v^r &\equiv \alpha^{k+ar} v^r \pmod p\\ &\equiv \alpha^{k+ar} \alpha^{-ar} \pmod p\\ &\equiv \alpha^k \pmod p\\ &\equiv \gamma \pmod p.\end{aligned}$$

Thus Bob will accept Alice's proof of identity (assuming he is honest), and the protocol is said to have the *completeness* property.

Here is a small (toy) example illustrating the challenge-and-response aspect of the protocol.

Example 9.2
Suppose $p = 88667$, $q = 1031$ and $t = 10$. The element $\alpha = 70322$ has order q in $\mathbb{Z}_p{}^*$. Suppose Alice's secret exponent is $a = 755$; then

$$v = \alpha^{-a} \bmod p$$

$$\begin{aligned} &= 70322^{1031-755} \bmod 88667 \\ &= 13136. \end{aligned}$$

Now suppose Alice chooses $k = 543$. Then she computes

$$\begin{aligned} \gamma &= \alpha^k \bmod p \\ &= 70322^{543} \bmod 88667 \\ &= 84109. \end{aligned}$$

and sends γ to Bob. Suppose Bob issues the challenge $r = 1000$. Then Alice computes

$$\begin{aligned} y &= k + ar \bmod q \\ &= 543 + 755 \times 1000 \bmod 1031 \\ &= 851 \end{aligned}$$

and sends y to Bob. Bob then verifies that

$$84109 \equiv 70322^{851} 13136^{1000} \pmod{88667}.$$

So Bob believes that he is communicating with Alice. $\square$

Next, let's consider how someone might try to impersonate Alice. An imposter, Olga, might try to impersonate Alice by forging a certificate

$$\mathrm{C}'(\mathrm{Alice}) = (\mathrm{ID}(\mathrm{Alice}), v', s'),$$

where $v' \neq v$. But s' is supposed to be a signature of $(\mathrm{ID}(\mathrm{Alice}), v')$, and this is verified by Bob in step 3 of the protocol. If the signature scheme of the TA is secure, Olga will not be able to forge a signature s' which will subsequently be verified by Bob.

Another approach would be for Olga to use Alice's correct certificate $\mathrm{C}(\mathrm{Alice}) = (\mathrm{ID}(\mathrm{Alice}), v, s)$ (recall that certificates are not secret, and the information on a certificate is revealed each time the identification protocol is executed). But Olga will not be able to impersonate Alice unless she also knows the value of a. This is because of the "challenge" r in step 4. In step 5, Olga would have to compute y, but y is a function of a. The computation of a from v involves solving a discrete log problem, which we assume is intractible.

We can prove a more precise statement about the security of the protocol, as follows.

THEOREM 9.1
Suppose Olga knows a value γ for which she has probability $\epsilon \geq 1/2^{t-1}$ of successfully impersonating Alice in the verification protocol. Then Olga can compute a in polynomial time.

PROOF For a fraction ϵ of the 2^t possible challenges r, Olga can compute a value y which will be accepted in step 6 by Bob. Since $\epsilon \geq 1/2^{t-1}$, we have that $2^t/\epsilon \geq 2$, and therefore Olga can compute values y_1, y_2, r_1 and r_2 such that

$$y_1 \not\equiv y_2 \pmod p$$

and

$$\gamma \equiv \alpha^{y_1} v^{r_1} \equiv \alpha^{y_2} v^{r_2} \pmod p.$$

It follows that

$$\alpha^{y_1 - y_2} \equiv v^{r_2 - r_1} \pmod p.$$

Since $v = \alpha^{-a}$, we have that

$$y_1 - y_2 \equiv a(r_1 - r_2) \pmod q.$$

Now, $0 < |r_2 - r_1| < 2^t$ and $q > 2^t$ is prime. Hence $\gcd(r_2 - r_1, q) = 1$, and Olga can compute

$$a = (y_1 - y_2)(r_1 - r_2)^{-1} \bmod q,$$

as desired. ∎

The above theorem proves that anyone who has a non-negligible chance of successfully executing the identification protocol must know (or be able to compute in polynomial time) Alice's secret exponent a. This property is often referred to as *soundness*.

We illustrate with an example.

Example 9.3
Suppose we have the same parameters as in Example 9.2: $p = 88667$, $q = 1031$, $t = 10$, $\alpha = 70322$, $a = 755$ and $v = 13136$. Suppose Olga learns that

$$\alpha^{851} v^{1000} \equiv \alpha^{454} v^{19} \pmod p.$$

Then she can compute

$$a = (851 - 454)(1000 - 19)^{-1} \bmod 1031 = 755,$$

and thus discover Alice's secret exponent. □

We have proved that the protocol is sound and complete. But soundness and completeness are not sufficient to ensure that the protocol is "secure." For example, if Alice simply revealed the value of her exponent a to prove her identity to Olga (say), the protocol would still be sound and complete. However, it would be completely insecure, since Olga could subsequently impersonate Alice.

This motivates the consideration of the secret information released to a verifier (or an observer) who takes part in the protocol (in this protocol, the secret information is the value of the exponent a). Our hope is that no information about a can be gained by Olga when Alice proves her identity, for then Olga would then be able to masquerade as Alice.

In general, we could envision a situation whereby Alice proves her identity to Olga, say, on several different occasions. Perhaps Olga does not choose her challenges (i.e., the values of r) in a random way. After several executions of the protocol, Olga will try to determine the value of a so she can subsequently impersonate Alice. If Olga can determine no information about the value of a by taking part in a polynomial number of executions of the protocol and then performing a polynomial amount of computation, then we would be convinced that the protocol is secure.

It has not been proven that the **Schnorr Scheme** is secure. But in the next section, we present a modification of the **Schnorr Scheme**, due to Okamoto, that can be proved to be secure given a certain computational assumption.

The **Schnorr Scheme** was designed to be very fast and efficient, both from a computational point of view and in the amount of information that needs to be exchanged in the protocol. It is also designed to minimize the amount of computation performed by Alice. This is desirable because in many practical applications, Alice's computations will be performed by a smart card with low computing power, while Bob's computations will be performed by a more powerful computer.

For the purpose of discussion, let's assume that ID(Alice) is a 512-bit string. v also comprises 512 bits, and s will be 320 bits if the **DSS** is used as a signature scheme. The total size of the certificate $\mathrm{C}(\mathrm{Alice})$ (which needs to be stored on Alice's smart card) is then 1444 bits.

Let us consider Alice's computations: step 1 requires a modular exponentiation to be performed; step 5 comprises one modular addition and one modular multiplication. It is the modular exponentiation that is computationally intensive, but this can be precomputed offline, if desired. The online computations to be performed by Alice are very modest.

It is also a simple matter to calculate the number of bits that are communicated during the protocol. We can depict the information that is communicated in the

Previous Chapter Previous Page Home Next page Next Chapter

form of a diagram:

$$\begin{tikzcd}
\text{Alice} \arrow[r, "{C, \gamma}", shift left=2] & \text{Bob} \arrow[l, "r"] \\
\text{Alice} \arrow[r, "y"] & \text{Bob}
\end{tikzcd}$$

Alice gives Bob $1444 + 512 = 1956$ bits of information in step 2; Bob gives Alice 40 bits in step 4; and Alice gives Bob 140 bits in step 6. So the communication requirements are quite modest, as well.

9.3 The Okamoto Identification Scheme

In this section, we present a modification of the **Schnorr Scheme** due to Okamoto. This modification can be proved secure, assuming the intractibility of computing a particular discrete logarithm in $\mathbb{Z}_p$.

To set up the scheme, the TA chooses p and q as in the **Schnorr Scheme**. The TA also chooses two elements $\alpha_1, \alpha_2 \in \mathbb{Z}_p{}^*$ both having order q. The value $c = \log_{\alpha_1} \alpha_2$ is kept secret from all the participants, including Alice. We will assume that it is infeasible for anyone (even a coalition of Alice and Olga, say) to compute the value c. As before, the TA chooses a signature scheme and hash function. The certificate issued to Alice by the TA is constructed as described in Figure 9.4. The **Okamoto Identification Scheme** is presented in Figure 9.5.

Here is an example of the **Okamoto Scheme**.

Example 9.4
As in previous examples, we will take $p = 88667, q = 1031$, and $t = 10$. Suppose $\alpha_1 = 58902$ and $\alpha_2 = 73611$ (both α_1 and α_2 have order q in $\mathbb{Z}_p{}^*$). Now, suppose $a_1 = 846$ and $a_2 = 515$; then $v = 13078$.

Suppose Alice chooses $k_1 = 899$ and $k_2 = 16$; then $\gamma = 14573$. If Bob issues the challenge $r = 489$ then Alice will respond with $y_1 = 131$ and $y_2 = 287$. Bob will verify that

$$58902^{131} 73611^{287} 13078^{489} \equiv 14574 \pmod{88667}.$$

So Bob will accept Alice's proof of identity. $\square$

The proof that the protocol is complete (i.e., that Bob will accept Alice's proof of identity) is straightforward. The main difference between Okamoto's and Schnorr's scheme is that we can prove that the **Okamoto Scheme** is secure provided that the computation of the discrete logarithm $\log_{\alpha_1} \alpha_2$ is intractible.

FIGURE 9.4
Issuing a certificate to Alice

1. The TA establishes Alice's identity and issues an identification string ID(Alice).
2. Alice secretly chooses two random exponents a_1, a_2, where $0 \leq a_1, a_2 \leq q-1$. Alice computes
$$v = {\alpha_1}^{-a_1}{\alpha_2}^{-a_2} \bmod p$$
and gives v to the TA.
3. The TA generates a signature
$$s = sig_{\mathrm{TA}}(I, v).$$
The certificate
$$\mathrm{C(Alice)} = (\mathrm{ID(Alice)}, v, s)$$
is given to Alice.

The proof of security is quite subtle. Here is the general idea: As before, Alice identifies herself to Olga polynomially many times by executing the protocol. We then suppose (hoping to obtain a contradiction) that Olga is able to learn some information about the values of Alice's secret exponents a_1 and a_2. If this is so, then we will show that (with high probability) Alice and Olga together will be able to compute the discrete logarithm c in polynomial time. This contradicts the assumption made above, and proves that Olga must be unable to obtain any information about Alice's exponents by taking part in the protocol.

The first part of this procedure is similar to the soundness proof for the **Schnorr Scheme**.

THEOREM 9.2
Suppose Olga knows a value γ for which she has probability $\epsilon \geq 1/2^{t-1}$ of successfully impersonating Alice in the verification protocol. Then, in polynomial time, Olga can compute values b_1 and b_2 such that
$$v \equiv {\alpha_1}^{-b_1}{\alpha_2}^{-b_2} \pmod p.$$

PROOF For a fraction ϵ of the 2^t possible challenges r, Olga can compute values y_1, y_2, z_1, z_2, r and s with $r \neq s$ and
$$\gamma \equiv {\alpha_1}^{y_1}{\alpha_2}^{y_2}v^r \equiv {\alpha_1}^{z_1}{\alpha_2}^{z_2}v^s \pmod p.$$

FIGURE 9.5
The Okamoto identification scheme

1. Alice chooses random numbers k_1, k_2, where $0 \leq k_1, k_2 \leq q-1$, and computes
$$\gamma = {\alpha_1}^{k_1}{\alpha_2}^{k_2} \bmod p.$$
2. Alice sends her certificate $\mathrm{C(Alice)} = (\mathrm{ID(Alice)}, v, s)$ and γ to Bob.
3. Bob verifies the signature of the TA by checking that $ver_{\mathrm{TA}}(\mathrm{ID(Alice)}, v, s) = \mathrm{true}$.
4. Bob chooses a random number r, $1 \leq r \leq 2^t$ and gives it to Alice.
5. Alice computes
$$y_1 = k_1 + a_1 r \bmod q$$
and
$$y_2 = k_2 + a_2 r \bmod q$$
and gives y_1 and y_2 to Bob.
6. Bob verifies that
$$\gamma \equiv {\alpha_1}^{y_1}{\alpha_2}^{y_2} v^r \pmod{p}.$$

Define
$$b_1 = (y_1 - z_1)(r - s)^{-1} \bmod q$$
and
$$b_2 = (y_2 - z_2)(r - s)^{-1} \bmod q.$$
Then it is easy to check that
$$v \equiv {\alpha_1}^{-b_1}{\alpha_2}^{-b_2} \pmod{p},$$
as desired. ∎

We now proceed to show how Alice and Olga can together compute the value of c.

THEOREM 9.3
Suppose Olga knows a value γ for which she has probability $\epsilon \geq 1/2^{t-1}$ of successfully impersonating Alice in the verification protocol. Then, with probability $1 - 1/q$, Alice and Olga can together compute $\log_{\alpha_1} \alpha_2$ in polynomial time.

PROOF By Theorem 9.2, Olga is able to determine values b_1 and b_2 such that

$$v \equiv {\alpha_1}^{-b_1}{\alpha_2}^{-b_2} \pmod p.$$

Now suppose that Alice reveals the values a_1 and a_2 to Olga. Of course

$$v \equiv {\alpha_1}^{-a_1}{\alpha_2}^{-a_2} \pmod p,$$

so it must be the case that

$${\alpha_1}^{a_1-b_1} \equiv {\alpha_2}^{b_2-a_2} \pmod p.$$

Suppose that $(a_1, a_2) \neq (b_1, b_2)$. Then $(a_1 - b_1)^{-1} \bmod q$ exists, and the discrete log

$$c = \log_{\alpha_1} \alpha_2 = (a_1 - b_1)(b_2 - a_2)^{-1} \bmod q$$

can be computed in polynomial time.

There remains to be considered the possibility that $(a_1, a_2) = (b_1, b_2)$. If this happens, then the value of c cannot be computed as described above. However, we will argue that $(a_1, a_2) = (b_1, b_2)$ will happen only with very small probability $1/q$, so the procedure whereby Alice and Olga compute c will almost surely succeed.

Define

$$\mathcal{A} = \{(a_1', a_2') \in \mathbb{Z}_q \times \mathbb{Z}_q : {\alpha_1}^{-a_1'}{\alpha_2}^{-a_2'} \equiv {\alpha_1}^{-a_1}{\alpha_2}^{-a_2} \pmod p\}.$$

That is, $\mathcal{A}$ consists of all the possible ordered pairs that could be Alice's secret exponents. Observe that

$$\mathcal{A} = \{(a_1 - c\theta, a_2 + \theta) : \theta \in \mathbb{Z}_q\},$$

where $c = \log_{\alpha_1} \alpha_2$. Thus $\mathcal{A}$ consists of q ordered pairs.

The ordered pair (b_1, b_2) computed by Olga is certainly in the set $\mathcal{A}$. We will argue that the value of the pair (b_1, b_2) is independent of the value of the pair (a_1, a_2) that comprises Alice's secret exponents. Since (a_1, a_2) was originally chosen at random by Alice, it must be the case that the probability that $(a_1, a_2) = (b_1, b_2)$ is $1/q$.

So, we need to say what we mean by (b_1, b_2) being "independent" of (a_1, a_2). The idea is that Alice's pair (a_1, a_2) is one of the q possible ordered pairs in the set $\mathcal{A}$, and no information about which is the "correct" ordered pair is revealed by Alice identifying herself to Olga. (Stated informally, Olga knows that an ordered pair from $\mathcal{A}$ comprises Alice's exponents, but she has no way of telling which one.)

Let's look at the information that is exchanged during the identification protocol. Basically, in each execution of the protocol, Alice chooses a γ; Olga chooses an r; and Alice reveals y_1 and y_2 such that

$$\gamma \equiv {\alpha_1}^{y_1}{\alpha_2}^{y_2}v^r \pmod p.$$

Recall that Alice computes

$$y_1 = k_1 + a_1 r \bmod q$$

and

$$y_2 = k_2 + a_2 r \bmod q,$$

where

$$\gamma = {\alpha_1}^{k_1}{\alpha_2}^{k_2} \bmod p.$$

But note that k_1 and k_2 are not revealed (nor are a_1 and a_2).

The particular quadruple (γ, r, y_1, y_2) that is generated during one execution of the protocol appears to depend on Alice's ordered pair (a_1, a_2), since y_1 and y_2 are defined in terms of a_1 and a_2. But we will show that each such quadruple could equally well be generated from any other ordered pair $(a_1', a_2') \in \mathcal{A}$. To see this, suppose $(a_1', a_2') \in \mathcal{A}$, i.e., $a_1' = a_1 - c\theta$ and $a_2' = a_2 + \theta$, where $0 \leq \theta \leq q-1$. We can express y_1 and y_2 as follows:

$$\begin{aligned} y_1 &= k_1 + a_1 r \\ &= k_1 + (a_1' + c\theta)r \\ &= (k_1 + rc\theta) + a_1' r, \end{aligned}$$

and

$$\begin{aligned} y_2 &= k_2 + a_2 r \\ &= k_2 + (a_2' - \theta)r \\ &= (k_2 - r\theta) + a_2' r, \end{aligned}$$

where all arithmetic is performed in $\mathbb{Z}_q$. That is, the quadruple (γ, r, y_1, y_2) is also consistent with the ordered pair (a_1', a_2') using the random choices $k_1' = k_1 + rc\theta$ and $k_2' = k_2 - r\theta$ to produce (the same) γ. We have already noted that the values of k_1 and k_2 are not revealed by Alice, so the quadruple (γ, r, y_1, y_2) yields no information regarding which ordered pair in $\mathcal{A}$ Alice is actually using for her secret exponents. This completes the proof. ∎

This security proof is certainly quite elegant and subtle. It would perhaps be useful to recap the features of the protocol that lead to the proof of security. The basic idea involves having Alice choose two secret exponents rather than one.

There are a total of q pairs in the set $\mathcal{A}$ that are "equivalent" to Alice's pair (a_1, a_2). The fact that leads to the ultimate contradiction is that knowledge of two different pairs in $\mathcal{A}$ provides an efficient method of computing the discrete logarithm c. Alice, of course, knows one pair in $\mathcal{A}$; and we proved that if Olga can impersonate Alice, then Olga is able to compute a pair in $\mathcal{A}$ which (with high probability) is different from Alice's pair. Thus Alice and Olga together can find two pairs in $\mathcal{A}$ and compute c, which provides the desired contradiction.

Here is an example to illustrate the computation of $\log_{\alpha_1} \alpha_2$ by Alice and Olga.

Example 9.5

As in Example 9.4, we will take $p = 88667$, $q = 1031$ and $t = 10$, and assume that $v = 13078$.

Suppose Olga has determined that

$$\alpha_1{}^{131}\alpha_2{}^{287}v^{489} \equiv \alpha_1{}^{890}\alpha_2{}^{303}v^{199} \pmod p.$$

Then she can compute

$$b_1 = (131 - 890)(489 - 199)^{-1} \bmod 1031 = 456$$

and

$$b_2 = (287 - 303)(489 - 199)^{-1} \bmod 1031 = 519.$$

Now, using the values of a_1 and a_2 supplied by Alice, the value

$$c = (846 - 456)(519 - 515)^{-1} \bmod 1031 = 613$$

is computed. This value c is in fact $\log_{\alpha_1} \alpha_2$, as can be verified by calculating

$$58902^{613} \bmod 88667 = 73611.$$

□

Finally, we should emphasize that, although there is no known proof that the **Schnorr Scheme** is secure (even assuming that the discrete logarithm problem is intractible), neither is there any known weakness in the scheme. Actually, the **Schnorr Scheme** might be preferred in practice to the **Okamoto Scheme** simply because it is somewhat faster.

9.4 The Guillou-Quisquater Identification Scheme

In this section, we describe another identification scheme, due to Guillou and Quisquater, that is based on **RSA**.

FIGURE 9.6
Issuing a certificate to Alice

1. The TA establishes Alice's identity and issues an identification string ID(Alice).
2. Alice secretly chooses an integer u, where $0 \leq u \leq n-1$. Alice computes
$$v = (u^{-1})^b \bmod n$$
and gives u to the TA.
3. The TA generates a signature
$$s = sig_{TA}(I, v).$$
The certificate
$$\mathrm{C(Alice)} = (\mathrm{ID(Alice)}, v, s)$$
is given to Alice.

The set-up of the scheme is as follows: The TA chooses two primes p and q and forms the product $n = pq$. The values of p and q are secret, while n is public. As is usually the case, p and q should be chosen large enough that factoring n is intractible. Also, the TA chooses a large prime integer b which will function as a security parameter as well as being a public RSA encryption exponent; to be specific, let us suppose that b is a 40-bit prime. Finally, the TA chooses a signature scheme and hash function.

The certificate issued to Alice by the TA is constructed as described in Figure 9.6. When Alice wants to prove her identity to Bob, say, the protocol of Figure 9.7 is executed. We will prove that the **Guillou-Quisquater Scheme** is sound and complete. However, the scheme has not been proved to be secure (even assuming that the **RSA** cryptosystem is secure).

Example 9.6
Suppose the TA chooses $p = 467$ and $q = 479$, so $n = 223693$. Suppose also that $b = 503$ and Alice's secret integer $u = 101576$. Then she will compute

$$\begin{aligned} v &= (u^{-1})^b \bmod n \\ &= (101576^{-1})^{503} \bmod 223693 \\ &= 89888. \end{aligned}$$

FIGURE 9.7
The Guillou-Quisquater identification scheme

1. Alice chooses a random number k, where $0 \leq k \leq n-1$ and computes
$$\gamma = k^b \bmod n.$$
2. Alice gives Bob her certificate $C(\text{Alice}) = (\text{ID}(\text{Alice}), v, s)$ and γ.
3. Bob verifies the signature of the TA by checking that $ver_{TA}(\text{ID}(\text{Alice}), v, s) = \text{true}$.
4. Bob chooses a random number r, $0 \leq r \leq b-1$ and gives it to Alice.
5. Alice computes
$$y = ku^r \bmod n$$
and gives y to Bob.
6. Bob verifies that
$$\gamma \equiv v^r y^b \pmod{n}.$$

Now, let's assume that Alice is proving her identity to Bob and she chooses $k = 187485$; then she gives Bob the value

$$\begin{aligned}\gamma &= k^b \bmod n\\ &= 187485^{503} \bmod 223693\\ &= 24412.\end{aligned}$$

Suppose Bob responds with the challenge $r = 375$. Then Alice will compute

$$\begin{aligned}y &= ku^r \bmod n\\ &= 187485 \times 101576^{375} \bmod 223693\\ &= 93725\end{aligned}$$

and gives it to Bob. Bob then verifies that

$$24412 \equiv 89888^{375} 93725^{503} \pmod{223693}.$$

Hence, Bob accepts Alice's proof of identity. $\square$

As is generally the case, proving completeness is quite simple:

$$\begin{aligned} v^r y^b &\equiv (u^{-b})^r (ku^r)^b \pmod n \\ &\equiv u^{-br} k^b u^{br} \pmod n \\ &\equiv k^b \pmod n \\ &\equiv \gamma \pmod n. \end{aligned}$$

Now, let us consider soundness. We will prove that the scheme is sound provided that it is infeasible to compute u from v. Since v is formed from u by RSA encryption, this is a plausible assumption to make.

THEOREM 9.4

Suppose Olga knows a value γ for which she has probability $\epsilon > 1/b$ of successfully impersonating Alice in the verification protocol. Then, in polynomial time, Olga can compute u.

PROOF For some γ, Olga can compute values y_1, y_2, r_1, r_2 with $r_1 \neq r_2$, such that

$$\gamma \equiv v^{r_1} {y_1}^b \equiv v^{r_2} {y_2}^b \pmod n.$$

Suppose, without loss of generality, that $r_1 > r_2$. Then we have

$$v^{r_1 - r_2} \equiv (y_2/y_1)^b \pmod n.$$

Since $0 < r_1 - r_2 < b$ and b is prime, $t = (r_1 - r_2)^{-1} \bmod b$ exists, and it can be computed in polynomial time by Olga using the Euclidean algorithm. Hence, we have that

$$v^{(r_1 - r_2)t} \equiv (y_2/y_1)^{bt} \pmod n.$$

Now,

$$(r_1 - r_2)t = \ell b + 1$$

for some positive integer ℓ, so

$$v^{\ell b + 1} \equiv (y_2/y_1)^{bt} \pmod n,$$

or equivalently,

$$v \equiv (y_2/y_1)^{bt} (v^{-1})^{\ell b} \pmod n.$$

Now raise both sides of the congruence to the power $b^{-1} \bmod \phi(n)$, to get the following:

$$u^{-1} \equiv (y_2/y_1)^t (v^{-1})^\ell \pmod n.$$

Finally, compute the inverse modulo n of both sides of this congruence, to obtain the following formula for u:

$$u = (y_1/y_2)^t v^\ell \bmod n.$$

Olga can use this formula to compute u in polynomial time. ∎

Example 9.7
As in the previous example, suppose that $n = 223693$, $b = 503$, $u = 101576$ and $v = 89888$. Suppose Olga has learned that

$$v^{401} 103386^b \equiv v^{375} 93725^b \pmod{n}.$$

She will first compute

$$\begin{aligned} t &= (r_1 - r_2)^{-1} \bmod b \\ &= (401 - 375)^{-1} \bmod 503 \\ &= 445. \end{aligned}$$

Next, she calculates

$$\begin{aligned} \ell &= \frac{(r_1 - r_2)t - 1}{b} \\ &= \frac{(401 - 375)445 - 1}{503} \\ &= 23. \end{aligned}$$

Finally, she can obtain the secret value u as follows:

$$\begin{aligned} u &= (y_1/y_2)^t v^\ell \bmod n \\ &= (103386/93725)^{445} 89888^{23} \bmod 223693 \\ &= 101576. \end{aligned}$$

Thus Alice's secret exponent has been compromised. □

9.4.1 Identity-based Identification Schemes

The **Guillou-Quisquater Identification Scheme** can be tranformed into what is known as an *identity-based* identification scheme. This basically means that certificates are not necessary. Instead, the TA computes the value of u as a function of Alice's ID string, using a public hash function h with range $\mathbb{Z}_n$. This is done as indicated in Figure 9.8. The identification protocol now works as described

FIGURE 9.8
Issuing a value u to Alice

1. The TA establishes Alice's identity and issues an identification string ID(Alice).
2. The TA computes

$$u = (h(\text{ID(Alice)})^{-1})^a \bmod n$$

and gives u to Alice.

FIGURE 9.9
The Guillou-Quisquater identity-based identification scheme

1. Alice chooses a random number k, where $0 \le k \le n-1$ and computes

$$\gamma = k^b \bmod n.$$

2. Alice gives ID(Alice) and γ to Bob.
3. Bob computes

$$v = h(\text{ID(Alice)}).$$

4. Bob chooses a random number r, $0 \le r \le b-1$ and gives it to Alice.
5. Alice computes

$$y = ku^r \bmod n$$

and gives y to Bob.

6. Bob verifies that

$$\gamma \equiv v^r y^b \pmod{n}.$$

in Figure 9.9. The value v is computed from Alice's ID string via the public hash function h. In order to carry out the identification protocol, Alice needs to know the value of u, which can be computed only by the TA (assuming that the **RSA** cryptosystem is secure). If Olga tries to identify herself as Alice, she will not succeed because she does not know the value of u.

9.5 Converting Identification to Signature Schemes

There is a standard method of converting an identification scheme to a signature scheme. The basic idea is to replace the verifier (Bob) by a public hash function, h. In a signature scheme obtained by this approach, the message is not hashed before it is signed; the hashing is integrated into the signing algorithm.

We illustrate this approach by converting the **Schnorr Scheme** into a signature scheme. See Figure 9.10. In practice, one would probably take the hash function h to be the **SHS**, with the result reduced modulo q. Since the **SHS** produces a bitstring of length 160 and q is a 160-bit prime, the modulo q reduction is necessary only if the message digest produced by the **SHS** exceeds q; and even in this situation it is necessary only to subtract q from the result.

In proceeding from an identification scheme to a signature scheme, we replaced a 40-bit challenge by a 160-bit message digest. 40 bits suffice for a challenge since an impostor needs to be able to guess the challenge in order to precompute a response that will be accepted. But in the context of a signature scheme, we need message digests of a much larger size, in order to prevent attacking the scheme by finding collisions in the hash function.

Other identification schemes can be converted to signature schemes in a similar fashion.

9.6 Notes and References

The **Schnorr Identification Scheme** is from [SC91], the **Okamoto Scheme** was presented in [OK93], and the **Guillou-Quisquater Scheme** can be found in [GQ88]. Another scheme that can be proved secure under a plausible computational assumption has been given by Brickell and McCurley [BM92].

Other popular identification schemes include the **Fiege-Fiat-Shamir Scheme** [FFS88] (see also [FS87]) and Shamir's **Permuted Kernel Scheme** [SH90]. The **Fiege-Fiat-Shamir Scheme** is proved secure using zero-knowledge techniques (see Chapter 13 for more information on zero-knowledge proofs).

The method of constructing signature schemes from identification schemes is due to Fiat and Shamir [FS87]. They also describe an identity-based version of their identification scheme.

FIGURE 9.10
Schnorr Signature Scheme

Let p be a 512-bit prime such that the discrete log problem in $\mathbb{Z}_p$ is intractible, and let q be a 160-bit prime that divides $p-1$. Let $\alpha \in \mathbb{Z}_p{}^*$ be a qth root of 1 modulo p. Let h be a hash function with range $\mathbb{Z}_q$. Define $\mathcal{P} = \mathbb{Z}_p{}^*$, $\mathcal{A} = \mathbb{Z}_p{}^* \times \mathbb{Z}_q$, and define

$$\mathcal{K} = \{(p, q, \alpha, a, v) : v \equiv \alpha^{-a} \pmod p\}.$$

The values p, q, α, and v are public, and a is secret.

For $K = (p, q, \alpha, a, v)$, and for a (secret) random number $k \in \mathbb{Z}_q{}^*$, define

$$sig_K(x, k) = (\gamma, y),$$

where

$$\gamma = \alpha^k \bmod p$$

and

$$y = k + ah(x, \gamma) \bmod q.$$

For $x, \gamma \in \mathbb{Z}_p{}^*$ and $y \in \mathbb{Z}_q$, define

$$ver(x, \gamma, y) = \text{true} \Leftrightarrow \gamma \equiv \alpha^y v^{h(x,\gamma)} \pmod p.$$

Surveys on identification schemes have been published by Burmester, Desmedt, and Beth [BDB92] and de Waleffe and Quisquater [DWQ93].

Exercises

9.1 Consider the following possible identification scheme. Alice possesses a secret key $n = pq$, where p and q are prime and $p \equiv q \equiv 3 \pmod 4$. The values n and ID(Alice) are signed by the TA, as usual, and stored on Alice's certificate. When Alice wants to identify herself to Bob, say, Bob will present Alice with a random quadratic residue modulo n, say x. Then Alice will compute a square root y of x and give it to Bob. Bob then verifies that $y^2 \equiv x \pmod n$. Explain why this scheme is insecure.

9.2 Suppose Alice is using the **Schnorr Scheme** where $q = 1201, p = 122503, t = 10$ and $\alpha = 11538$.

(a) Verify that α has order q in $\mathbb{Z}_p{}^*$.
(b) Suppose that Alice's secret exponent is $a = 357$. Compute v.
(c) Suppose that $k = 868$. Compute γ.
(d) Suppose that Bob issues the challenge $r = 501$. Compute Alice's response y.
(e) Perform Bob's calculations to verify y.

9.3 Suppose that Alice uses the **Schnorr Scheme** with p, q, t and α as in Exercise 9.2. Now suppose that $v = 51131$, and Olga has learned that

$$\alpha^3 v^{148} \equiv \alpha^{151} v^{1077} \pmod p.$$

Show how Olga can compute Alice's secret exponent a.

9.4 Suppose that Alice is using the **Okamoto Scheme** with $q = 1201$, $p = 122503$, $t = 10$, $\alpha_1 = 60497$ and $\alpha_2 = 17163$.
(a) Suppose that Alice's secret exponents are $a_1 = 432$ and $a_2 = 423$. Compute v.
(b) Suppose that $k_1 = 389$ and $k_2 = 191$. Compute γ.
(c) Suppose that Bob issues the challenge $r = 21$. Compute Alice's response, y_1 and y_2.
(d) Perform Bob's calculations to verify y_1 and y_2.

9.5 Suppose that Alice uses the **Okamoto Scheme** with p, q, t, α_1, and α_2 as in Exercise 9.4. Suppose also that $v = 119504$.
(a) Verify that

$$\alpha_1{}^{70}\alpha_2{}^{1033}v^{877} \equiv \alpha_1{}^{248}\alpha_2{}^{883}v^{992} \pmod p.$$

(b) Use this information to compute b_1 and b_2 such that

$$\alpha_1{}^{-b_1}\alpha_2{}^{-b_2} \equiv v \pmod p.$$

(c) Now suppose that Alice reveals that $a_1 = 484$ and $a_2 = 935$. Show how Alice and Olga together will compute $\log_{\alpha_1} \alpha_2$.

9.6 Suppose that Alice is using the **Quisquater Scheme** with $p = 503$, $q = 379$, and $b = 509$.
(a) Suppose that Alice's secret $u = 155863$. Compute v.
(b) Suppose that $k = 123845$. Compute γ.
(c) Suppose that Bob issues the challenge $r = 487$. Compute Alice's response, y.
(d) Perform Bob's calculations to verify y.

9.7 Suppose that Alice is using the **Quisquater Scheme** with $n = 199543$, $b = 523$ and $v = 146152$. Suppose that Olga has discovered that

$$v^{456}101360^b \equiv v^{257}36056^b \pmod n.$$

Show how Olga can compute u.