8

Key Distribution and Key Agreement

8.1 Introduction

We have observed that public-key systems have the advantage over private-key systems that a secure channel is not needed to exchange a secret key. But, unfortunately, most public-key systems are much slower than private-key systems such as **DES**, for example. So, in practice, private-key systems are usually used to encrypt "long" messages. But then we come back to the problem of exchanging secret keys.

In this chapter, we discuss several approaches to the problem of establishing secret keys. We will distinguish between key distribution and key agreement. *Key distribution* is defined to be a mechanism whereby one party chooses a secret key and then transmits it to another party or parties. *Key agreement* denotes a protocol whereby two (or more) parties jointly establish a secret key by communicating over a public channel. In a key agreement scheme, the value of the key is determined as a function of inputs provided by both parties.

As our setting, we have an insecure network of n users. In some of our schemes, we will have a *trusted authority* (denoted by TA) that is reponsible for such things as verifying the identities of users, choosing and transmitting keys to users, etc.

Since the network is insecure, we need to protect against potential opponents. Our opponent, Oscar, might be a *passive adversary*, which means that his actions are restricted to eavesdropping on messages that are transmitted over the channel. On the other hand, we might want to guard against the possibility that Oscar is an *active adversary*. An active adversary can do various types of nasty things such as the following:

1. alter messages that he observes being transmitted over the network
2. save messages for reuse at a later time
3. attempt to masquerade as various users in the network.

The objective of an active adversary might be one of the following:

1. to fool U and V into accepting an "invalid" key as valid (an invalid key could be an old key that has expired, or a key chosen by the adversary, to mention two possibilities)
2. to make U or V believe that they have exchanged a key with other when they have not.

The objective of a key distribution or key agreement protocol is that, at the end of the protocol, the two parties involved both have possession of the same key K, and the value of K is not known to any other party (except possibly the TA). Certainly it is much more difficult design a protocol providing this type of security in the presence of an active adversary as opposed to a passive one.

We first consider the idea of *key predistribution* in Section 8.2. For every pair of users $\{\mathrm{U}, \mathrm{V}\}$, the TA chooses a random key $K_{\mathrm{U,V}} = K_{\mathrm{V,U}}$ and transmits it "off-band" to U and V over a secure channel. (That is, the transmission of keys does not take place over the network, since the network is not secure.) This approach is unconditionally secure, but it requires a secure channel between the TA and every user in the network. But, of possibly even more significance is the fact that each user must store $n - 1$ keys, and the TA needs to transmit a total of $\binom{n}{2}$ keys securely (this is sometimes called the "n^2 problem"). Even for relatively small networks, this can become prohibitively expensive, and thus it is not really a practical solution.

In Section 8.2.1, we discuss an interesting unconditionally secure key predistribution scheme, due to Blom, that allows a reduction in the amount of secret information to be stored by the users in the network. We also present in Section 8.2.2 a computationally secure key predistribution scheme based on the discrete logarithm problem.

A more practical approach can be described as *on-line key distribution by TA*. In such a scheme, the TA acts as a *key server*. The TA shares a secret key K_{U} with every user U in the network. When U wishes to communicate with V, she requests a *session key* from the TA. The TA generates a session key K and sends it in encrypted form for U and V to decrypt. The well-known **Kerboros** system, which we describe in Section 8.3, is based on this approach.

If it is impractical or undesirable to have an on-line TA, then a common approach is to use a *key agreement protocol*. In a key agreement protocol, U and V jointly choose a key by communicating over a public channel. This remarkable idea is due to Diffie and Hellman, and (independently) to Merkle. We describe a few of the more popular key agreement protocols. A variation of the original protocol of Diffie and Hellman, modified to protect against an active adversary, is presented in Section 8.4.1. Two other interesting protocols are also discussed: the **MTI** scheme is presented in Section 8.4.2 and the **Girault** scheme is covered in Section 8.4.3.

8.2 Key Predistribution

In the basic method, the TA generates $\binom{n}{2}$ keys, and gives each key to a unique pair of users in a network of n users. As mentioned above, we require a secure channel between the TA and each user to transmit these keys. This is a significant improvement over each pair of users independently exchanging keys over a secure channel, since the number of secure channels required has been reduced from $\binom{n}{2}$ to n. But if n is large, this solution is not very practical, both in terms of the amount of information to be transmitted securely, and in the amount of information that each user must store securely (namely, the secret keys of the other other $n-1$ users).

Thus, it is of interest to try to reduce the amount of information that needs to be transmitted and stored, while still allowing each pair of users U and V to be able to (independently) compute a secret key $K_{\mathrm{U,V}}$. An elegant scheme to accomplish this, called the **Blom Key Predistribution Scheme**, is discussed in the next subsection.

8.2.1 Blom's Scheme

As above, we suppose that we have a network of n users. For convenience, we suppose that keys are chosen from a finite field $\mathbb{Z}_p$, where $p \geq n$ is prime. Let k be an integer, $1 \leq k \leq n-2$. The value k is the largest size coalition against which the scheme will remain secure. In the **Blom Scheme**, the TA will transmit $k+1$ elements of $\mathbb{Z}_p$ to each user over a secure channel (as opposed to $n-1$ in the basic key predistribution scheme). Each pair of users, U and V, will be able to compute a key $K_{\mathrm{U,V}} = K_{\mathrm{V,U}}$, as before. The security condition is as follows: any set of at most k users disjoint from $\{\mathrm{U}, \mathrm{V}\}$ must be unable to determine any information about $K_{\mathrm{U,V}}$ (note that we are speaking here about unconditional security).

We first present the special case of Blom's scheme where $k = 1$. Here, the TA will transmit two elements of $\mathbb{Z}_p$ to each user over a secure channel, and any individual user W will be unable to determine any information about $K_{\mathrm{U,V}}$ if $\mathrm{W} \neq \mathrm{U}, \mathrm{V}$. Blom's scheme is presented in Figure 8.1. We illustrate the **Blom Scheme** with $k = 1$ in the following example.

Example 8.1
Suppose the three users are U, V and W, $p = 17$, and their public elements are $r_{\mathrm{U}} = 12$, $r_{\mathrm{V}} = 7$ and $r_{\mathrm{W}} = 1$. Suppose that the TA chooses $a = 8$, $b = 7$ and $c = 2$, so the polynomial f is

$$f(x, y) = 8 + 7(x + y) + 2xy.$$

The g polynomials are as follows:

$$g_{\mathrm{U}}(x) = 7 + 14x$$

FIGURE 8.1
Blom Key Distribution Scheme ($k = 1$)

1. A prime number p is made public, and for each user U, an element $r_{\mathrm{U}} \in \mathbb{Z}_p$ is made public. The elements r_{U} must be distinct.
2. The TA chooses three random elements $a, b, c \in \mathbb{Z}_p$ (not necessarily distinct), and forms the polynomial
$$f(x, y) = a + b(x + y) + cxy \bmod p.$$
3. For each user U, the TA computes the polynomial
$$g_{\mathrm{U}}(x) = f(x, r_{\mathrm{U}}) \bmod p$$
and transmits $g_{\mathrm{U}}(x)$ to U over a secure channel. Note that $g_{\mathrm{U}}(x)$ is a linear polynomial in x, so it can be written as
$$g_{\mathrm{U}}(x) = a_{\mathrm{U}} + b_{\mathrm{U}}x,$$
where
$$a_{\mathrm{U}} = a + br_{\mathrm{U}} \bmod p$$
and
$$b_{\mathrm{U}} = b + cr_{\mathrm{U}} \bmod p.$$
4. If U and V want to communicate, then they use the common key
$$K_{\mathrm{U,V}} = K_{\mathrm{V,U}} = f(r_{\mathrm{U}}, r_{\mathrm{V}}) = a + b(r_{\mathrm{U}} + r_{\mathrm{V}}) + cr_{\mathrm{U}}r_{\mathrm{V}} \bmod p,$$
where U computes $K_{\mathrm{U,V}}$ as
$$f(r_{\mathrm{U}}, r_{\mathrm{V}}) = g_{\mathrm{U}}(r_{\mathrm{V}})$$
and V computes $K_{\mathrm{U,V}}$ as
$$f(r_{\mathrm{U}}, r_{\mathrm{V}}) = g_{\mathrm{V}}(r_{\mathrm{U}}).$$

$$g_V(x) = 6 + 4x$$
$$g_W(x) = 15 + 9x.$$

The three keys are thus

$$K_{U,V} = 3$$
$$K_{U,W} = 4$$
$$K_{V,W} = 10.$$

U would compute $K_{U,V}$ as

$$g_U(r_V) = 7 + 14 \times 7 \bmod 17 = 3$$

V would compute $K_{U,V}$ as

$$g_V(r_U) = 6 + 4 \times 12 \bmod 17 = 3.$$

We leave the computation of the other keys as an exercise for the reader. ▯

We now prove that no one user can determine any information about the key of two other users.

THEOREM 8.1
The **Blom Scheme** *with $k = 1$ is unconditionally secure against any individual user.*

PROOF Let's suppose that user W wants to try to compute the key

$$K_{U,V} = a + b(r_U + r_V) + c r_U r_V \bmod p.$$

The values r_U and r_V are public, but a, b and c are unknown. W does know the values

$$a_W = a + b r_W \bmod p$$

and

$$b_W = b + c r_W \bmod p$$

since these are the coefficients of the polynomial $g_W(x)$ that was sent to W by the TA.

What we will do is show that the information known by W is consistent with any possible value $\ell \in \mathbb{Z}_p$ of the key $K_{U,V}$. Hence, W cannot rule out any values for $K_{U,V}$. Consider the following matrix equation (in $\mathbb{Z}_p$):

$$\begin{pmatrix} 1 & r_U + r_V & r_U r_V \\ 1 & r_W & 0 \\ 0 & 1 & r_W \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \ell \\ a_W \\ b_W \end{pmatrix}.$$

The first equation represents the hypothesis that $K_{\mathrm{U,V}} = \ell$; the second and third equations contain the information that W knows about a, b and c from $g_{\mathrm{W}}(x)$.

The determinant of the coefficient matrix is

$$r_{\mathrm{W}}^2 + r_{\mathrm{U}} r_{\mathrm{V}} - (r_{\mathrm{U}} + r_{\mathrm{V}}) r_{\mathrm{W}} = (r_{\mathrm{W}} - r_{\mathrm{U}})(r_{\mathrm{W}} - r_{\mathrm{V}}),$$

where all arithmetic is done in $\mathbb{Z}_p$. Since $r_{\mathrm{W}} \neq r_{\mathrm{U}}$ and $r_{\mathrm{W}} \neq r_{\mathrm{V}}$, it follows that the coefficient matrix has non-zero determinant, and hence the matrix equation has a unique solution for a, b, c. In other words, any possible value ℓ of $K_{\mathrm{U,V}}$ is consistent with the information known to W. ▮

On the other hand, a coalition of two users, say $\{\mathrm{W}, \mathrm{X}\}$, will be able to determine any key $K_{\mathrm{U,V}}$ where $\{\mathrm{W}, \mathrm{X}\} \cap \{\mathrm{U}, \mathrm{V}\} = \emptyset$. W and X together know that

$$\begin{aligned} a_{\mathrm{W}} &= a + b r_{\mathrm{W}} \\ b_{\mathrm{W}} &= b + c r_{\mathrm{W}} \\ a_{\mathrm{X}} &= a + b r_{\mathrm{X}} \\ b_{\mathrm{X}} &= b + c r_{\mathrm{X}}. \end{aligned}$$

Thus they have four equations in three unknowns, and they can easily compute a unique solution for a, b and c. Once they know a, b and c, they can form the polynomial $f(x, y)$ and compute any key they wish.

It is straightforward to generalize the scheme to remain secure against coalitions of size k. The only thing that changes is step 2. The TA will use a polynomial $f(x, y)$ having the form

$$f(x, y) = \sum_{i=0}^{k} \sum_{j=0}^{k} a_{i,j} x^i y^j \bmod p,$$

where $a_{i,j} \in \mathbb{Z}_p$ $(0 \leq i \leq k, 0 \leq j \leq k)$, and $a_{i,j} = a_{j,i}$ for all i, j. The remainder of the protocol is unchanged.

8.2.2 Diffie-Hellman Key Predistribution

In this section, we describe a key predistribution scheme that is a modification of the well-known Diffie-Hellman key exchange protocol that we will discuss a bit later, in Section 8.4. We call this the **Diffie-Hellman Key Predistribution Scheme**. The scheme is computationally secure provided a problem related to the **Discrete Logarithm** problem is intractible.

We will describe the scheme over $\mathbb{Z}_p$, where p is prime, though it can be implemented in any finite group in which the **Discrete Logarithm** problem is intractible. We will assume that α is a primitive element of $\mathbb{Z}_p$, and that the values p and α are publicly known to everyone in the network.

In this scheme, ID(U) will denote certain identification information for each user U in the network, e.g., his or her name, e-mail address, telephone number, or other relevant information. Also, each user U has a secret exponent a_U (where $0 \leq a_U \leq p-2$), and a corresponding public value

$$b_U = \alpha^{a_U} \bmod p.$$

The TA will have a signature scheme with a (public) verification algorithm ver_{TA} and a secret signing algorithm sig_{TA}. Finally, we will implicitly assume that all information is hashed, using a public hash function, before it is signed. To make the procedures easier to read, we will not include the necessary hashing in the description of the protocols.

Certain information pertaining to a user U will be authenticated by means of a *certificate* which is issued and signed by the TA. Each user U will have a certificate

$$C(U) = (ID(U), b_U, sig_{TA}(ID(U), b_U)),$$

where b_U is formed as described above (note that the TA does not need to know the value of a_U). A certificate for a user U will be issued when U joins the network. Certificates can be stored in a public database, or each user can store his or her own certificate. The signature of the TA on a certificate allows anyone in the network to verify the information it contains.

It is very easy for U and V to compute the common key

$$K_{U,V} = \alpha^{a_U a_V} \bmod p,$$

as shown in Figure 8.2.

We illustrate the algorithm with a small example.

Example 8.2

Suppose $p = 25307$ and $\alpha = 2$ are publicly known (p is prime and α is a primitive root modulo p). Suppose U chooses $a_U = 3578$. Then she computes

$$\begin{aligned} b_U &= \alpha^{a_U} \bmod p \\ &= 2^{3578} \bmod 25307 \\ &= 6113, \end{aligned}$$

which is placed on her certificate. Suppose V chooses $a_V = 19956$. Then he computes

$$\begin{aligned} b_V &= \alpha^{a_V} \bmod p \\ &= 2^{19956} \bmod 25307 \\ &= 7984, \end{aligned}$$

which is placed on his certificate.

FIGURE 8.2
Diffie-Hellman Key Predistribution

1. A prime p and a primitive element $\alpha \in \mathbb{Z}_p{}^*$ are made public.
2. V computes
$$K_{\mathrm{U,V}} = \alpha^{a_{\mathrm{U}} a_{\mathrm{V}}} \bmod p = b_{\mathrm{U}}{}^{a_{\mathrm{V}}} \bmod p,$$
using the public value b_{U} from U's certificate, together with his own secret value a_{V}.
3. U computes
$$K_{\mathrm{U,V}} = \alpha^{a_{\mathrm{U}} a_{\mathrm{V}}} \bmod p = b_{\mathrm{V}}{}^{a_{\mathrm{U}}} \bmod p,$$
using the public value b_{V} from V's certificate, together with her own secret value a_{U}.

Now U can compute the key

$$\begin{aligned} K_{\mathrm{U,V}} &= b_{\mathrm{V}}{}^{a_{\mathrm{U}}} \bmod p \\ &= 7984^{3578} \bmod 25307 \\ &= 3694, \end{aligned}$$

and V can compute the same key

$$\begin{aligned} K_{\mathrm{U,V}} &= b_{\mathrm{U}}{}^{a_{\mathrm{V}}} \bmod p \\ &= 6113^{19956} \bmod 25307 \\ &= 3694. \end{aligned}$$

□

Let us think about the security of this scheme in the presence of a passive or active adversary. The signature of the TA on users' certificates effectively prevents W from altering any information on someone else's certificate. Hence we need only worry about passive attacks. So the pertinent question is: Can a user W compute $K_{\mathrm{U,V}}$ if W $\neq$ U, V? In other words, given $\alpha^{a_{\mathrm{U}}} \bmod p$ and $\alpha^{a_{\mathrm{V}}} \bmod p$ (but not a_{U} nor a_{V}), is it feasible to compute $\alpha^{a_{\mathrm{U}} a_{\mathrm{V}}} \bmod p$? This problem is called the **Diffie-Hellman** problem, and it is formally defined (using an equivalent but slightly different presentation) in Figure 8.3. It is clear that **Diffie-Hellman Key Predistribution** is secure against a passive adversary if and

Previous Chapter Previous Page Home Next page Next Chapter

FIGURE 8.3
The Diffie-Hellman problem

Problem Instance $I = (p, \alpha, \beta, \gamma)$, where p is prime, $\alpha \in \mathbb{Z}_p{}^*$ is a primitive element, and $\beta, \gamma \in \mathbb{Z}_p{}^*$.

Objective Compute $\beta^{\log_\alpha \gamma} \bmod p$ $(= \gamma^{\log_\alpha \beta} \bmod p)$.

only if the **Diffie-Hellman** problem is intractible.

If W could determine a_U from b_U, or if he could determine a_V from b_V, then he could compute $K_{U,V}$ exactly as U (or V) does. But both these computations are instances of the **Discrete Log** problem. So, provided that the **Discrete Log** problem in $\mathbb{Z}_p$ is intractible, **Diffie-Hellman Key Predistribution** is secure against this particular type of attack. However, it is an unproven conjecture that any algorithm that solves the **Diffie-Hellman** problem could also be used to solve the **Discrete Log** problem. (This is very similar to the situation with **RSA**, where it is conjectured, but not proved, that breaking **RSA** is polynomially equivalent to factoring.)

By the remarks made above, the **Diffie-Hellman** problem is no more difficult than the **Discrete Log** problem. Although we cannot say precisely how difficult this problem is, we can relate its security to that of another cryptosystem we have already studied, namely the **ElGamal Cryptosystem**.

THEOREM 8.2
Breaking the **ElGamal Cryptosystem** *is equivalent to solving the* **Diffie-Hellman** *problem.*

PROOF First we recall how **ElGamal** encryption and decryption work. The key is $K = (p, \alpha, a, \beta)$, where $\beta = \alpha^a \bmod p$ (a is secret and p, α, and β are public). For a (secret) random number $k \in \mathbb{Z}_{p-1}$,

$$e_K(x, k) = (y_1, y_2),$$

where

$$y_1 = \alpha^k \bmod p$$

and

$$y_2 = x\beta^k \bmod p.$$

For $y_1, y_2 \in \mathbb{Z}_p{}^*$,

$$d_K(y_1, y_2) = y_2(y_1{}^a)^{-1} \bmod p.$$

Suppose we have an algorithm **A** to solve the **Diffie-Hellman** problem, and we are given an **ElGamal** encryption (y_1, y_2). We will apply the algorithm **A** with inputs p, α, y_1, and β. Then, we obtain the value

$$\begin{aligned}\mathbf{A}(p, \alpha, y_1, \beta) &= \mathbf{A}(p, \alpha, \alpha^k, \alpha^a)\\ &= \alpha^{ka} \bmod p\\ &= \beta^k \bmod p.\end{aligned}$$

Then, the decryption of (y_1, y_2) can easily be computed as

$$x = y_2(\beta^k)^{-1} \bmod p.$$

Conversely, suppose we have an algorithm **B** that performs **ElGamal** decryption. That is, **B** takes as inputs p, α, β, y_1, and y_2, and computes the quantity

$$x = y_2({y_1}^{\log_\alpha \beta})^{-1} \bmod p.$$

Now, given inputs p, α, β, and γ for the **Diffie-Hellman** problem, it is easy to see that

$$\begin{aligned}\mathbf{B}(p, \alpha, \beta, \gamma, 1)^{-1} &= 1((\gamma^{\log_\alpha \beta})^{-1})^{-1} \bmod p\\ &= \gamma^{\log_\alpha \beta} \bmod p,\end{aligned}$$

as desired. ∎

8.3 Kerboros

In the key predistribution methods we discussed in the previous section, each pair of users can compute one fixed key. If the same key is used for a long period of time, there is a danger that it might be compromised. Thus it is often preferable to use an on-line method in which a new session key is produced every time a pair of users want to communicate (this property is called *key freshness*).

If on-line key distribution is used, there is no need for any network user to store keys to communicate with other users (each user will share a key with the TA, however). Session keys will be transmitted on request by the TA. It is the responsibility of the TA to ensure key freshness.

Kerboros is a popular key serving system based on private-key cryptography. In this section, we give an overview of the protocol for issuing session keys in **Kerboros**. Each user U shares a secret **DES** key K_U with the TA. In the most recent version of **Kerboros** (version V), all messages to be transmitted are encrypted using cipher block chaining (CBC) mode, as described in Section 3.4.1.

Previous Chapter Previous Page Home Next page Next Chapter

FIGURE 8.4
Transmission of a session key using Kerboros

1. U asks the TA for a session key to communicate with V.
2. The TA chooses a random session key K, a timestamp T, and a lifetime L.
3. The TA computes
$$m_1 = e_{K_U}(K, \mathrm{ID(V)}, T, L)$$
and
$$m_2 = e_{K_V}(K, \mathrm{ID(U)}, T, L)$$
and sends m_1 and m_2 to U.
4. U uses the decryption function d_{K_U} to compute K, T, L, and ID(V) from m_1. She then computes
$$m_3 = e_K(\mathrm{ID(U)}, T)$$
and sends m_3 to V along with the message m_2 she received from the TA.
5. V uses the decryption function d_{K_V} to compute K, T, L and ID(U) from m_2. He then uses d_K to compute T and ID(U) from m_3. He checks that the two values of T and the two values of ID(U) are the same. If so, then V computes
$$m_4 = e_K(T + 1)$$
and sends it to U.
6. U decrypts m_4 using e_K and verifies that the result is $T + 1$.

As in Section 8.2.2, ID(U) will denote public identification information for user U. When a request for a session key is sent to the TA, the TA will generate a new random session key K. Also, the TA will record the time at which the request is made as a *timestamp*, T, and specify the *lifetime*, L, during which K will be valid. That is, the session key K is to be regarded as a valid key from time T to time $T + L$. All this information is encrypted and transmitted to U and (eventually) to V. Before going into more details, we will present the protocol in Figure 8.4.

Previous Chapter Previous Page Home Next page Next Chapter

The information transmitted in the protocol is illustrated in the following diagram:

$$\begin{tikzcd}[column sep=huge]
\mathrm{TA} \arrow[r, "\substack{e_{K_{\mathrm{U}}}(K,\mathrm{ID}(\mathrm{V}),T,L)\\ e_{K_{\mathrm{V}}}(K,\mathrm{ID}(\mathrm{U}),T,L)}"] & \mathrm{U} \arrow[r, shift left, "\substack{e_K(\mathrm{ID}(\mathrm{U}),T)\\ e_{K_{\mathrm{V}}}(K,\mathrm{ID}(\mathrm{U}),T,L)}"] & \mathrm{V} \arrow[l, shift left, "e_K(T+1)"]
\end{tikzcd}$$

We will now explain what is going on in the various steps of the protocol. Although we have no formal proof that **Kerboros** is "secure" against an active adversary, we can at least give some informal motivation of the features of the protocol.

As mentioned above, the TA generates K, T, and L in step 2. In step 3, this information, along with ID(V), is encrypted using the key K_{U} shared by U and the TA to form m_1. Also, K, T, L, and ID(U) are encrypted using the key K_{V} shared by V and the TA to form m_2. Both these encrypted messages are sent to U.

U can use her key to decrypt m_1, and thus obtain K, T, and L. She will verify that the current time is in the interval from T to $T + L$. She can also check that the session key K has been issued for her desired communicant V by verifying the information ID(V) decrypted from m_1.

Next, U will relay m_2 to V. As well, U will use the new session key K to encrypt T and ID(U) and send the resulting message m_3 to V.

When V receives m_2 and m_3 from U, he decrypts m_2 to obtain T, K, L and ID(U). Then he uses the new session key K to decrypt m_3 and he verifies that T and ID(U), as decrypted from m_2 and m_3, are the same. This ensures V that the session key encrypted within m_2 is the same key that was used to encrypt m_3. Then V uses K to encrypt $T + 1$, and sends the result back to U as message m_4.

When U receives m_4, she decrypts it using K and verifies that the result is $T + 1$. This ensures U that the session key K has been successfully transmiited to V, since K was needed in order to produce the message m_4.

It is important to note the different functions of the messages transmitted in this protocol. The messages m_1 and m_2 are used to provide secrecy in the transmission of the session key K. On the other hand, m_3 and m_4 are used to provide *key confirmation*, that is, to enable U and V to convince each other that they possess the same session key K. In most key distribution schemes, (session) key confirmation can be included as a feature if it is not already present. Usually this is done in a similar fashion as it is done in **Kerboros**, namely by using the new session key K to encrypt known quantities. In **Kerboros**, U uses K to encrypt ID(U) and T, which are already encrypted in m_2. Similarly, V uses K to encrypt $T + 1$.

The purpose of the timestamp T and lifetime L is to prevent an active adversary from storing "old" messages for retransmission at a later time (this is called a *replay attack*). This method works because keys are not accepted as valid once they have expired.

FIGURE 8.5
Diffie-Hellman Key Exchange

1. U chooses a_U at random, $0 \leq a_U \leq p-2$.
2. U computes $\alpha^{a_U} \bmod p$ and sends it to V.
3. V chooses a_V at random, $0 \leq a_V \leq p-2$.
4. V computes $\alpha^{a_V} \bmod p$ and sends it to U.
5. U computes
$$K = (\alpha^{a_V})^{a_U} \bmod p$$
and V computes
$$K = (\alpha^{a_U})^{a_V} \bmod p.$$

One of the drawbacks of **Kerboros** is that all the users in the network should have synchronized clocks, since the current time is used to determine if a given session key K is valid. In practice, it is very difficult to provide perfect synchronization, so some amount of variation in times must be allowed.

8.4 Diffie-Hellman Key Exchange

If we do not want to use an on-line key server, then we are forced to use a key agreement protocol to exchange secret keys. The first and best known key agreement protocol is **Diffie-Hellman Key Exchange**. We will assume that p is prime, α is a primitive element of $\mathbb{Z}_p$, and that the values p and α are publicly known. (Alternatively, they could be chosen by U and communicated to V in the first step of the protocol.) **Diffie-Hellman Key Exchange** is presented in Figure 8.5.

At the end of the protocol, U and V have computed the same key

$$K = \alpha^{a_U a_V} \bmod p.$$

This protocol is very similar to **Diffie-Hellman Key Predistribution** described earlier. The difference is that the exponents a_U and a_V of users U and V (respectively) are chosen anew each time the protocol is run, instead of being fixed. Also, in this protocol, both U and V are assured of key freshness, since the session key depends on both random exponents a_U and a_V.

Previous Chapter Previous Page Home Next page Next Chapter

8.4.1 The Station-to-station Protocol

Diffie-Hellman Key Exchange is supposed to look like this:

$$\begin{tikzcd}
\mathrm{U} \arrow[rr, shift left=2, "\alpha^{a_\mathrm{U}}"] & & \mathrm{V} \arrow[ll, shift left=2, "\alpha^{a_\mathrm{V}}"']
\end{tikzcd}$$

Unfortunately, the protocol is vulnerable to an active adversary who uses an *intruder-in-the-middle* attack. There is an episode of *The Lucy Show* in which Vivian Vance is having dinner in a restaurant with a date, and Lucille Ball is hiding under the table. Vivian and her date decide to hold hands under the table. Lucy, trying to avoid detection, holds hands with each of them and they think they are holding hands with each other.

An intruder-in-the-middle attack on the **Diffie-Hellman Key Exchange** protocol works in the same way. W will intercept messages between U and V and substitute his own messages, as indicated in the following diagram:

$$\begin{tikzcd}
\mathrm{U} \arrow[rr, shift left=2, "\alpha^{a_\mathrm{U}}"] & & \mathrm{W} \arrow[ll, shift left=2, "\alpha^{a'_\mathrm{V}}"'] \arrow[rr, shift left=2, "\alpha^{a'_\mathrm{U}}"] & & \mathrm{V} \arrow[ll, shift left=2, "\alpha^{a_\mathrm{V}}"']
\end{tikzcd}$$

At the end of the protocol, U has actually established the secret key $\alpha^{a_\mathrm{U} a'_\mathrm{V}}$ with W, and V has established a secret key $\alpha^{a'_\mathrm{U} a_\mathrm{V}}$ with W. When U tries to encrypt a message to send to V, W will be able to decrypt it but V will not. (A similar situation holds if V sends a message to U.)

Clearly, it is essential for U and V to make sure that they are exchanging messages with each other and not with W. Before exchanging keys, U and V might carry out a separate protocol to establish each other's identity, for example by using one of the identification schemes that we will describe in Chapter 9. But this offers no protection against an intruder-in-the-middle attack if W simply remains inactive until after U and V have proved their identities to each other. Hence, the key agreement protocol should itself authenticate the participants' identities at the same time as the key is being established. Such a protocol will be called *authenticated key agreement*.

We will describe an authenticated key agreement protocol which is a modification of **Diffie-Hellman Key Exchange**. The protocol assumes a publicly known prime p and a primitive element α, and it makes use of certificates. Each user U will have a signature scheme with verification algorithm ver_U and signing algorithm sig_U. The TA also has a signature scheme with public verification algorithm ver_TA. Each user U has a certificate

$$C(\mathrm{U}) = (\mathrm{ID}(\mathrm{U}), ver_\mathrm{U}, sig_\mathrm{TA}(\mathrm{ID}(\mathrm{U}), ver_\mathrm{U})),$$

where ID(U) is identification information for U.

FIGURE 8.6
Simplified Station-to-station Protocol

1. U chooses a random number a_U, $0 \leq a_U \leq p-2$.
2. U computes
$$\alpha^{a_U} \bmod p$$
and sends it to V.
3. V chooses a random number a_V, $0 \leq a_V \leq p-2$.
4. V computes
$$\alpha^{a_V} \bmod p.$$
Then he computes
$$K = (\alpha^{a_U})^{a_V} \bmod p$$
and
$$y_V = sig_V(\alpha^{a_V}, \alpha^{a_U}).$$
4. V sends $(C(V), \alpha^{a_V}, y_V)$ to U.
5. U computes
$$K = (\alpha^{a_V})^{a_U} \bmod p.$$
She verifies y_V using ver_V and she verifies $C(V)$ using ver_{TA}.
6. U computes
$$y_U = sig_U(\alpha^{a_U}, \alpha^{a_V})$$
and she sends $(C(U), y_U)$ to V.
7. V verifies y_U using ver_U and he verifies $C(U)$ using ver_{TA}.

The authenticated key agreement known as the **Station-to-station Protocol** (or **STS** for short) is due to Diffie, Van Oorschot, and Wiener. The protocol we present in Figure 8.6 is a slight simplification; it can be used in such a way that it is conformant with the ISO 9798-3 protocols.

Previous Chapter Previous Page Home Next page Next Chapter

The information exchanged in the simplified **STS protocol** (excluding certificates) is illustrated as follows:

$$\begin{tikzcd}[column sep=huge]
\mathrm{U} \arrow[r, shift left=6, "\alpha^{a_U}"] \arrow[r, leftarrow, "{\alpha^{a_V}, sig_V(\alpha^{a_V}, \alpha^{a_U})}"] \arrow[r, shift right=6, "{sig_U(\alpha^{a_U}, \alpha^{a_V})}"'] & \mathrm{V}
\end{tikzcd}$$

Let's see how this protects against an intruder-in-the-middle attack. As before, W will intercept α^{a_U} and replace it with $\alpha^{a'_U}$. W then receives $\alpha^{a_V}, sig_V(\alpha^{a_V}, \alpha^{a'_U})$ from V. He would like to replace α^{a_V} with $\alpha^{a'_V}$, as before. However, this means that he must also replace $sig_V(\alpha^{a_V}, \alpha^{a'_U})$ by $sig_V(\alpha^{a'_V}, \alpha^{a_U})$. Unfortunately for W, he cannot compute V's signature on $(\alpha^{a'_V}, \alpha^{a_U})$ since he doesn't know V's signing algorithm sig_V. Similarly, W is unable to replace $sig_U(\alpha^{a_U}, \alpha^{a'_V})$ by $sig_U(\alpha^{a'_U}, \alpha^{a_V})$ because he does not know U's signing algorithm.

This is illustrated in the following diagram:

$$\begin{tikzcd}[column sep=huge]
\mathrm{U} \arrow[r, shift left=6, "\alpha^{a_U}"] \arrow[r, leftarrow, "{\alpha^{a'_V}, sig_V(\alpha^{a'_V}, \alpha^{a_U}) = \ ?}"] \arrow[r, shift right=6, "{sig_U(\alpha^{a_U}, \alpha^{a'_V})}"'] & \mathrm{W} \arrow[r, shift left=6, "\alpha^{a'_U}"] \arrow[r, leftarrow, "{\alpha^{a_V}, sig_V(\alpha^{a_V}, \alpha^{a'_U})}"] \arrow[r, shift right=6, "{sig_U(\alpha^{a'_U}, \alpha^{a_V}) = \ ?}"'] & \mathrm{V}
\end{tikzcd}$$

It is the use of signatures that thwarts the intruder-in-the-middle attack.

The protocol, as described in Figure 8.6, does not provide key confirmation. However, it is easy to modify so that it does, by defining

$$y_V = e_K(sig_V(\alpha^{a_V}, \alpha^{a_U}))$$

in step 4 and defining

$$y_U = e_K(sig_U(\alpha^{a_U}, \alpha^{a_V}))$$

in step 6. (As in **Kerboros**, we obtain key confirmation by encrypting a known quantity using the new session key.) The resulting protocol is known as the **Station-to-station Protocol**. We leave the remaining details for the interested reader to fill in.

8.4.2 MTI Key Agreement Protocols

Matsumoto, Takashima, and Imai have constructed several interesting key agreement protocols by modifying **Diffie-Hellman Key Exchange**. These protocols, which we call **MTI** protocols, do not require that U and V compute any signatures. They are *two-pass protocols* since there are only two separate transmissions of information performed (one from U to V and one from V to U). In contrast, the **STS** protocol is a three-pass protocol.

FIGURE 8.7
Matsumoto-Takashima-Imai Key Agreement Protocol

1. U chooses r_U at random, $0 \leq r_U \leq p-2$, and computes
$$s_U = \alpha^{r_U} \bmod p.$$
2. U sends $(C(U), s_U)$ to V.
3. V chooses r_V at random, $0 \leq r_V \leq p-2$, and computes
$$s_V = \alpha^{r_V} \bmod p.$$
4. V sends $(C(V), s_V)$ to U.
5. U computes
$$K = {s_V}^{a_U} {b_V}^{r_U} \bmod p,$$
where she obtains the value b_V from $C(V)$; and V computes
$$K = {s_U}^{a_V} {b_U}^{r_V} \bmod p,$$
where he obtains the value b_U from $C(U)$.

We present one of the **MTI** protocols. The setting for this protocol is the same as for **Diffie-Hellman Key Predistribution**. We assume a publicly known prime p and a primitive element α. Each user U has an ID string, ID(U), a secret exponent a_U $(0 \leq a_U \leq p-2)$, and a corresponding public value
$$b_U = \alpha^{a_U} \bmod p.$$
The TA has a signature scheme with a (public) verification algorithm ver_{TA} and a secret signing algorithm sig_{TA}.

Each user U will have a certificate
$$C(U) = (ID(U), b_U, sig_{TA}(ID(U), b_U)),$$
where b_U is formed as described above.

We present the MTI key agreement protocol in Figure 8.7. At the end of the protocol, U and V have both computed the same key
$$K = \alpha^{r_U a_V + r_V a_U} \bmod p.$$

We give an example to illustrate this protocol.

Example 8.3
Suppose $p = 27803$ and $\alpha = 5$ are publicly known. Assume U chooses $a_U = 21131$; then she will compute

$$b_U = 5^{21131} \bmod 27803 = 21420$$

which is placed on her certificate. As well, assume V chooses $a_V = 17555$. Then he will compute

$$b_V = 5^{17555} \bmod 27803 = 17100$$

which is placed on his certificate.
Now suppose that U chooses $r_U = 169$; then she will send the value

$$s_U = 5^{169} \bmod 27803 = 6268$$

to V. Suppose that V chooses $r_V = 23456$; then he will send the value

$$s_V = 5^{23456} \bmod 27803 = 26759$$

to U.
Now U can compute the key

$$\begin{aligned} K_{U,V} &= {s_V}^{a_U} {b_V}^{r_U} \bmod p \\ &= 26759^{21131} 17100^{169} \bmod 27803 \\ &= 21600, \end{aligned}$$

and V can compute the key

$$\begin{aligned} K_{U,V} &= {s_U}^{a_V} {b_U}^{r_V} \bmod p \\ &= 6268^{17555} 21420^{23456} \bmod 27803 \\ &= 21600, \end{aligned}$$

Thus U and V have computed the same key. $\square$

The information transmitted during the protocol is depicted as follows:

$$\begin{tikzcd} U \arrow[r, shift left, "C(U),\, \alpha^{r_U} \bmod p"] & V \arrow[l, shift left, "C(V),\, \alpha^{r_V} \bmod p"] \end{tikzcd}$$

Let's look at the security of the scheme. It is not too difficult to show that the security of the **MTI** protocol against a passive adversary is exactly the same

Previous Chapter Previous Page Home Next page Next Chapter

as the **Diffie-Hellman** problem — see the exercises. As with many protocols, proving security in the presence of an active adversary is problematic. We will not attempt to prove anything in this regard, and we limit ourselves to some informal arguments.

Here is one threat we might consider: Without the use of signatures during the protocol, it might appear that there is no protection against an intruder-in-the-middle attack. Indeed, it is possible that W might alter the values that U and V send each other. We depict one typical scenario that might arise, as follows:

$$\begin{tikzcd}[column sep=huge]
\mathrm{U} \arrow[r, shift left, "\mathrm{C(U)}, \alpha^{r_{\mathrm{U}}}"] & \mathrm{W} \arrow[l, shift left, "\mathrm{C(V)}, \alpha^{r'_{\mathrm{V}}}"] \arrow[r, shift left, "\mathrm{C(U)}, \alpha^{r'_{\mathrm{U}}}"] & \mathrm{V} \arrow[l, shift left, "\mathrm{C(V)}, \alpha^{r_{\mathrm{V}}}"]
\end{tikzcd}$$

In this situation, U and V will compute different keys: U will compute

$$K = \alpha^{r_{\mathrm{U}} a_{\mathrm{V}} + r'_{\mathrm{V}} a_{\mathrm{U}}} \bmod p.$$

while V will compute

$$K = \alpha^{r'_{\mathrm{U}} a_{\mathrm{V}} + r_{\mathrm{V}} a_{\mathrm{U}}} \bmod p.$$

However, neither of the key computations of U or V can be carried out by W, since they require knowledge of the secret exponents a_{U} and a_{V}, respectively. So even though U and V have computed different keys (which will of course be useless to them), neither of these keys can be computed by W (assuming the intractibility of the **Discrete Log** problem). In other words, both U and V are assured that the other is the only user in the network that could compute the key that they have computed. This property is sometimes called *implicit key authentication.*

8.4.3 Key Agreement Using Self-certifying Keys

In this section, we describe a method of key agreement, due to Girault, that does not require certificates. The value of a public key and the identity of its owner implicitly authenticate each other.

The **Girault Scheme** combines features of **RSA** and discrete logarithms. Suppose $n = pq$, where $p = 2p_1 + 1$, $q = 2q_1 + 1$, and p, q, p_1, and q_1 are all large primes. The multiplicative group $\mathbb{Z}_n{}^*$ is isomorphic to $\mathbb{Z}_p{}^* \times \mathbb{Z}_q{}^*$. The maximum order of any element in $\mathbb{Z}_n{}^*$ is therefore the least common multiple of $p - 1$ and $q - 1$, or $2p_1q_1$. Let α be an element of order $2p_1q_1$. Then the cyclic subgroup of $\mathbb{Z}_n{}^*$ generated by α is a suitable setting for the **Discrete Logarithm** problem.

In the **Girault Scheme**, the factorization of n is known only to the TA. The values n and α are public, but p, q, p_1, and q_1 are all secret. The TA chooses a public **RSA** encryption exponent, which we will denote by e. The corresponding decryption exponent, d, is secret (recall that $d = e^{-1} \bmod \phi(n)$).

Each user U has an ID string ID(U), as in previous schemes. A user U obtains a *self-certifying public key*, p_{U}, from the TA as indicated in Figure 8.8. Observe

FIGURE 8.8
Obtaining a self-certifying public key from the TA

1. U chooses a secret exponent a_U, and computes
$$b_U = \alpha^{a_U} \bmod n.$$
2. U gives a_U and b_U to the TA.
3. The TA computes
$$p_U = \left(b_U - \mathrm{ID}(U)\right)^d \bmod n.$$
4. The TA gives p_U to U.

that U needs the help of the TA to produce p_U. Note also that

$$b_U = {p_U}^e + \mathrm{ID}(U) \bmod n$$

can be computed from p_U and ID(U) using publicly available information.

The **Girault Key Agreement Protocol** is presented in Figure 8.9. The information transmitted during the protocol is depicted as follows:

$$\begin{tikzcd} U \arrow[r, shift left, "\mathrm{ID}(U), p_U, \alpha^{r_U} \bmod n"] & V \arrow[l, shift left, "\mathrm{ID}(V), p_V, \alpha^{r_V} \bmod n"] \end{tikzcd}$$

At the end of the protocol, U and V each have computed the key

$$K = \alpha^{r_U a_V + r_V a_U} \bmod n.$$

Here is an example of key exchange using the **Girault Scheme**.

Example 8.4
Suppose $p = 839$ and $q = 863$. Then $n = 724057$ and $\phi(n) = 722356$. The element $\alpha = 5$ has order $2p_1q_1 = \phi(n)/2$. Suppose the TA chooses $d = 125777$ as the **RSA** decryption exponent; then $e = 84453$.

Suppose U has $\mathrm{ID}(U) = 500021$ and $a_U = 111899$. Then $b_U = 488889$ and $p_U = 650704$. Suppose also that V has $\mathrm{ID}(V) = 500022$ and $a_V = 123456$. Then $b_V = 111692$ and $p_V = 683556$.

Now, U and V want to exchange a key. Suppose U chooses $r_U = 56381$, which means that $s_U = 171007$. Further, suppose V chooses $r_V = 356935$, which means that $s_V = 320688$.

Then both U and V will compute the same key $K = 42869$. □

FIGURE 8.9
Girault Key Agreement Protocol

1. U chooses r_U at random and computes
$$s_U = \alpha^{r_U} \bmod n.$$
2. U sends ID(U), p_U and s_U to V.
3. V chooses r_V at random and computes
$$s_V = \alpha^{r_V} \bmod n.$$
4. V sends ID(V), p_V and s_V to U.
5. U computes
$$K = s_V{}^{a_U} \left(p_V{}^e + \mathrm{ID}(V)\right)^{r_U} \bmod n;$$
and V computes
$$K = s_U{}^{a_V} \left(p_U{}^e + \mathrm{ID}(U)\right)^{r_V} \bmod n.$$

Let's consider how the self-certifying keys guard against one specific type of attack. Since the values b_U, p_U, and ID(U) are not signed by the TA, there is no way for anyone else to verify their authenticity directly. Suppose this information is forged by W (i.e., it is not produced in cooperation with the TA), who wants to masquerade as U. If W starts with ID(U) and a fake value b'_U, then there is no way for her to compute the exponent a'_U corresponding to b'_U if the **Discrete Log** problem is intractible. Without a'_U, a key computation cannot be performed by W (who is pretending to be U).

The situation is similar if W acts as an intruder-in-the-middle. W will be able to prevent U and V from computing a common key, but W is unable to duplicate the computations of either U or V. Thus the scheme provides implicit key authentication, as did the **MTI** protocol.

An attentive reader might wonder why U is required to supply the value a_U to the TA. Indeed, the TA can compute p_U directly from b_U, without knowing a_U. Actually, the important thing here is that the TA should be convinced that U knows the value of a_U before the TA computes p_U for U.

We illustrate this point by showing how the scheme can be attacked if the TA indiscriminately issues public keys p_U to users without first checking that they possess the value a_U corresponding to their b_U. Suppose W chooses a fake value

a'_U and computes the corresponding value

$$b'_U = \alpha^{a'_U} \bmod .$$

Here is how he can determine the corresponding public key

$$p'_U = \left(b'_U - \mathrm{ID(U)}\right)^d \bmod n.$$

W will compute

$$b'_W = b'_U - \mathrm{ID(U)} + \mathrm{ID(W)}$$

and then give b'_W and ID(W) to the TA. Suppose the TA issues the public key

$$p'_W = \left(b'_W - \mathrm{ID(W)}\right)^d \bmod n$$

to W. Using the fact that

$$b'_W - \mathrm{ID(W)} \equiv b'_U - \mathrm{ID(U)} \pmod n,$$

it is immediate that

$$p'_W = p'_U.$$

Now, at some later time, suppose U and V execute the protocol, and W substitutes information as follows:

$$\begin{tikzcd}
\mathrm{U} \arrow[r, shift left, "\mathrm{ID(U)}, p_U, \alpha^{r_U} \bmod n"] & \mathrm{W} \arrow[l, shift left, "\mathrm{ID(V)}, p_V, \alpha^{r_V} \bmod n"] \arrow[r, shift left, "\mathrm{ID(U)}, p'_U, \alpha^{r'_U} \bmod n"] & \mathrm{V} \arrow[l, shift left, "\mathrm{ID(V)}, p_V, \alpha^{r_V} \bmod n"]
\end{tikzcd}$$

Now V will compute the key

$$K' = \alpha^{r'_U a_V + r_V a'_U} \bmod n,$$

whereas U will compute the key

$$K = \alpha^{r_U a_V + r_V a_U} \bmod n.$$

W can compute K' as

$$K' = {s_V}^{a'_U} \left({p_V}^e + \mathrm{ID(V)}\right)^{r'_U} \bmod n.$$

Thus W and V share a key, but V thinks he is sharing a key with U. So W will be able to decrypt messages sent by V to U.

8.5 Notes and References

Blom presented his key predistribution scheme in [BL85]. Generalizations can be found in Blundo *et al.* [BDSHKVY93] and Beimel and Chor [BC94].

Diffie and Hellman presented their key exchange algorithm in [DH76]. The idea of key exchange was discovered independently by Merkle [ME78]. The material on authenticated key exchange is taken from Diffie, van Oorschot, and Wiener [DVW92].

Version V of **Kerbobos** is described in [KN93]. For a recent descriptive article on **Kerboros**, see Schiller [SC94].

The protocols of Matsumoto, Takashima, and Imai can be found in [MTI86]. Self-certifying key distribution was introduced by Girault [GIR91]. The scheme he presented was actually a key predistribution scheme; the modification to a key agreement scheme is based on [RV94].

Two recent surveys on key distribution and key agreement are Rueppel and Van Oorschot [RV94] and van Tilburg [vT93].

Exercises

8.1 Suppose the **Blom Scheme** with $k = 1$ is implemented for a set of four users, U, V, W and X. Suppose that $p = 7873$, $r_U = 2365$, $r_V = 6648$, $r_W = 1837$ and $r_X = 2186$. The secret g polynomials are as follows:

$$g_U(x) = 6018 + 6351x$$
$$g_V(x) = 3749 + 7121x$$
$$g_W(x) = 7601 + 7802x$$
$$g_X(x) = 635 + 6828x.$$

(a) Compute the key for each pair of users, verifying that each pair of users obtains a common key (that is, $K_{U,V} = K_{V,U}$, etc.).

(b) Show how W and X together can compute $K_{U,V}$.

8.2 Suppose the **Blom Scheme** with $k = 2$ is implemented for a set of five users, U, V, W, X and Y. Suppose that $p = 97$, $r_U = 14$, $r_V = 38$, $r_W = 92$, $r_X = 69$ and $r_Y = 70$. The secret g polynomials are as follows:

$$g_U(x) = 15 + 15x + 2x^2$$
$$g_V(x) = 95 + 77x + 83x^2$$
$$g_W(x) = 88 + 32x + 18x^2$$
$$g_X(x) = 62 + 91x + 59x^2$$
$$g_X(x) = 10 + 82x + 52x^2.$$

(a) Show how U and V each will compute the key $K_{U,V} = K_{V,U}$.

FIGURE 8.10
The MTI problem

Problem Instance $I = (p, \alpha, \beta, \gamma, \delta, \epsilon)$, where p is prime, $\alpha \in \mathbb{Z}_p{}^*$ is a primitive element, and $\beta, \gamma, \delta, \epsilon \in \mathbb{Z}_p{}^*$.

Objective Compute $\beta^{\log_\alpha \gamma} \delta^{\log_\alpha \epsilon} \bmod p$.

(b) Show how W, X and Y together can compute $K_{U,V}$.

8.3 Suppose that U and V carry out the **Diffie-Hellman Key Exchange** with $p = 27001$ and $\alpha = 101$. Suppose that U chooses $a_U = 21768$ and V chooses $a_V = 9898$. Show the computations performed by both U and V, and determine the key that they will compute.

8.4 Suppose that U and V carry out the **MTI Protocol** where $p = 30113$ and $\alpha = 52$. Suppose that U has $a_U = 8642$ and chooses $r_U = 28654$, and V has $a_V = 24673$ and chooses $r_V = 12385$. Show the computations performed by both U and V, and determine the key that they will compute.

8.5 If a passive adversary tries to compute the key K constructed by U and V by using the **MTI** protocol, then he is faced with an instance of what we might term the **MTI** problem, which we present in Figure 8.10. Prove that any algorithm that can be used to solve the **MTI** problem can be used to solve the **Diffie-Hellman** problem, and vice versa.

8.6 Consider the **Girault Scheme** where $p = 167$, $q = 179$, and hence $n = 29893$. Suppose $\alpha = 2$ and $e = 11101$.

(a) Compute d.

(b) Given that $\text{ID}(U) = 10021$ and $a_U = 9843$, compute b_U and p_U. Given that $\text{ID}(V) = 10022$ and $a_V = 7692$, compute b_V and p_V.

(c) Show how b_U can be computed from p_U and $\text{ID}(U)$ using the public exponent e. Similarly, show how b_V can be computed from p_V and $\text{ID}(V)$.

(d) Suppose that U chooses $r_U = 15556$ and V chooses $r_V = 6420$. Compute s_U and s_V, and show how U and V each compute their common key.