Note: Section 3.4 is based on the first edition of "Cryptography Theory and Practice", which was published in 1995, and it has not been updated for this course.

3.4 Kerberos

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.

Kerberos 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 **Kerberos**. Each user U shares a secret **DES** key K_{U} with the TA. In the most recent version of **Kerberos** (version V), all messages to be transmitted are encrypted using cipher-block chaining (CBC) mode, as described in Section **??**.

As in Section 3.3, $\mathrm{ID}(\mathrm{U})$ 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. As well, 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 3.3.

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

$$\mathrm{TA} \quad \xrightarrow{\substack{e_{K_{\mathrm{U}}}(K, \mathrm{ID}(\mathrm{V}), T, L) \\ e_{K_{\mathrm{V}}}(K, \mathrm{ID}(\mathrm{U}), T, L)}} \quad \mathrm{U} \quad \begin{array}{c} \xrightarrow{\substack{e_K(\mathrm{ID}(\mathrm{U}), T) \\ e_{K_{\mathrm{V}}}(K, \mathrm{ID}(\mathrm{U}), T, L)}} \\ \xleftarrow{\quad e_K(T+1) \quad} \end{array} \quad \mathrm{V}$$

We will now explain what is going on in the various steps of the protocol. Although we have no formal proof that **Kerberos** 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 $\mathrm{ID}(\mathrm{V})$, is encrypted using the key K_{U} shared by U and the TA to form m_1. Also, K, T, L and $\mathrm{ID}(\mathrm{U})$ 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 $\mathrm{ID}(\mathrm{V})$ 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 $\mathrm{ID}(\mathrm{U})$ 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 $\mathrm{ID}(\mathrm{U})$. Then he uses the new session key K to decrypt m_3 and he verifies that T and $\mathrm{ID}(\mathrm{U})$, 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 **Kerberos**, namely by using the new session key K to encrypt known quantities. In **Kerberos**, U uses K to encrypt $\mathrm{ID}(\mathrm{U})$ 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.

One of the drawbacks of **Kerberos** 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 varaiation in times must be allowed.

FIGURE 3.3
Transmission of a session key using Kerberos

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

$$m_1 = e_{K_{\mathrm{U}}}(K, \mathrm{ID}(\mathrm{V}), T, L)$$

and

$$m_2 = e_{K_{\mathrm{V}}}(K, \mathrm{ID}(\mathrm{U}), T, L)$$

and sends m_1 and m_2 to U.
4. U uses the decryption function $d_{K_{\mathrm{U}}}$ to compute K, T, L and $\mathrm{ID}(\mathrm{V})$ from m_1. She then computes

$$m_3 = e_K(\mathrm{ID}(\mathrm{U}), T)$$

and sends m_3 to V along with the message m_2 she received from the TA.
5. V uses the decryption function $d_{K_{\mathrm{V}}}$ to compute K, T, L and $\mathrm{ID}(\mathrm{U})$ from m_2. He then uses d_K to compute T and $\mathrm{ID}(\mathrm{U})$ from m_3. He checks that the two values of T and the two values of $\mathrm{ID}(\mathrm{U})$ 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 $T + 1$.