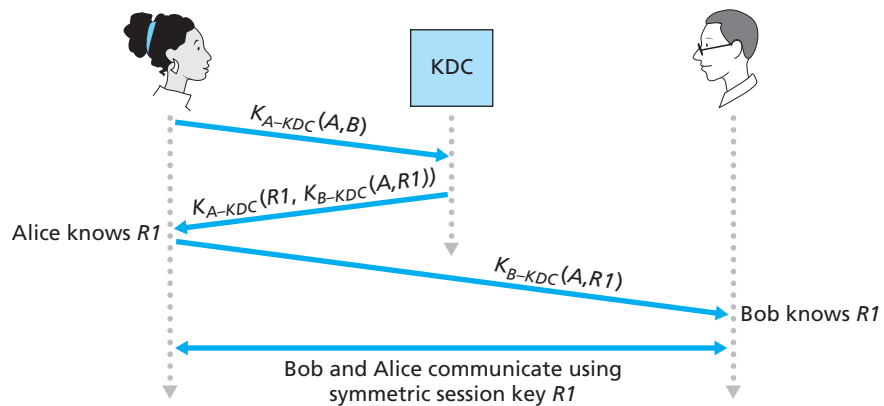


### 8.5.1 The Key Distribution Center

Suppose once again that Bob and Alice want to communicate using symmetric key cryptography. Suppose they have never met and thus have not established a shared secret key in advance. How can they now agree on a secret key, given that they can communicate with each other only over the network? A solution often adopted in practice is to use a trusted KDC.

The KDC is a server that shares a unique secret symmetric key with each registered user. This key might be manually installed at the server when a user first registers. The KDC knows the secret key of each user, and each user can communicate securely with the KDC using this key. Let's see how knowledge of this one key allows a user to obtain a key securely for communicating with any other registered user. Suppose that Alice and Bob are users of the KDC; they know only their individual keys,  $K_{A-KDC}$  and  $K_{B-KDC}$ , respectively, for communicating securely with the KDC. Alice takes the first step, and they proceed as illustrated in Figure 8.19.

1. Using  $K_{A-KDC}$  to encrypt her communication with the KDC, Alice sends a message to the KDC saying she ( $A$ ) wants to communicate with Bob ( $B$ ). We denote this message,  $K_{A-KDC}(A, B)$ .
2. The KDC, knowing  $K_{A-KDC}$ , decrypts  $K_{A-KDC}(A, B)$ . The KDC then generates a random number,  $R1$ . This is the shared key value that Alice and Bob will use to perform symmetric encryption when they communicate with each other. This key is referred to as a **one-time session key**, because Alice and Bob will use this key for only this one session. The KDC now needs to inform Alice and



**Figure 8.19** ♦ Setting up a one-time session key using a key distribution center

Bob of the value of  $RI$ . The KDC thus sends back a message to Alice, encrypted using  $K_{A-KDC}$ , containing the following.

- ◆  $RI$ , the one-time session key that Alice and Bob will use to communicate.
- ◆ A pair of values,  $A$  and  $RI$ , encrypted by the KDC using Bob's key,  $K_{B-KDC}$ . We denote this  $K_{B-KDC}(A, RI)$ . It is important to note that KDC is sending Alice not only the value of  $RI$  for her own use, but also an encrypted version of  $RI$  and Alice's name, encrypted using Bob's key. Alice can't decrypt this pair of values in the message (she doesn't know Bob's encryption key), but then she doesn't really need to. We'll see shortly that Alice will simply forward this encrypted pair of values to Bob, who will be able to decrypt them.

The KDC puts these items into a message, encrypts them using Alice's shared key, and sends them to Alice. The message from the KDC to Alice is thus  $K_{A-KDC}(RI, K_{B-KDC}(A, RI))$ .

3. Alice receives the message from the KDC, decrypts it, and extracts  $RI$  from the message. Alice now knows the one-time session key,  $RI$ . Alice also extracts  $K_{B-KDC}(A, RI)$  and forwards this to Bob.
4. Bob decrypts the received message,  $K_{B-KDC}(A, RI)$  using  $K_{B-KDC}$  and extracts  $A$  and  $RI$ . Bob now knows the one-time session key,  $RI$ , and the person with whom he is sharing this key,  $A$ . Of course, he takes care to authenticate Alice using  $RI$  before proceeding any further.

### 8.5.2 Public Key Certification

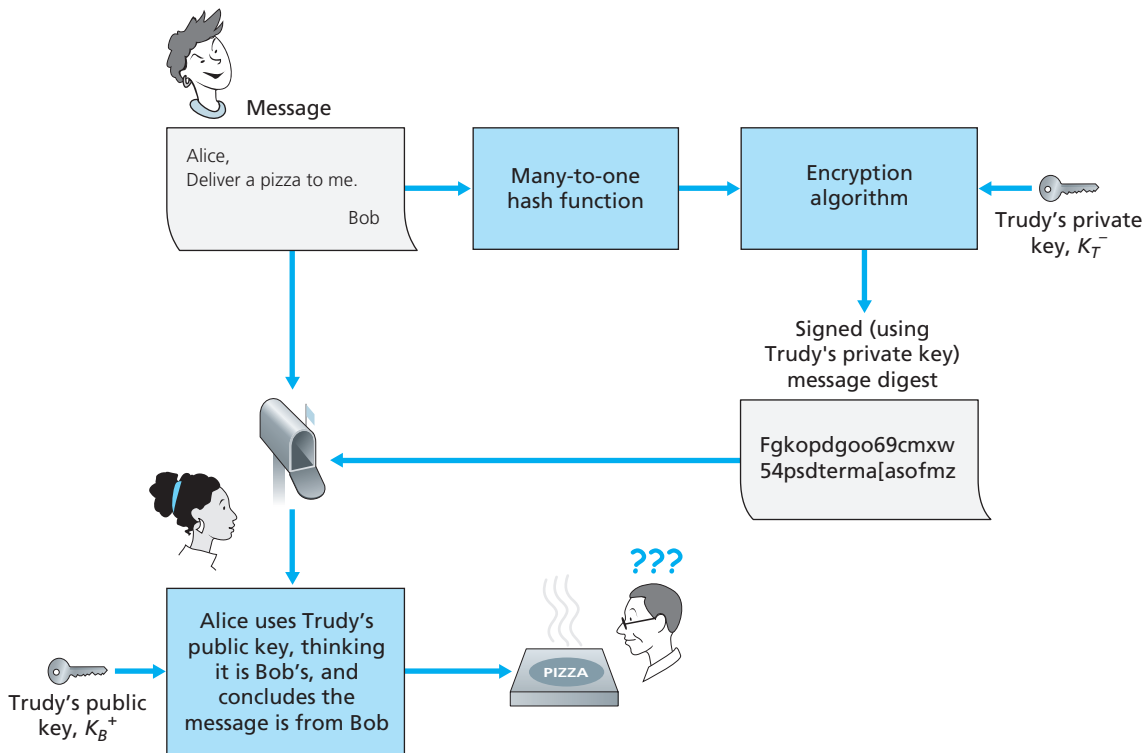
One of the principal features of public key cryptography is that it is possible for two entities to exchange secret messages without having to exchange secret keys. For example, when Alice wants to send a secret message to Bob, she simply encrypts the message with Bob's public key and sends the encrypted message to Bob; she doesn't need to know Bob's private key, nor does Bob need to know her private key. Thus, public key cryptography obviates the need for a KDC infrastructure.

Of course, with public key cryptography, the communicating entities still have to exchange public keys. A user can make its public key publicly available in many ways, for example, by posting the key on the user's personal Web page, by placing the key in a public key server, or by sending the key to a correspondent by e-mail. A Web commerce site can place its public key on its server in a manner such that browsers automatically download the public key when connecting to the site. Routers can place their public keys on public key servers, thereby allowing other network entities to retrieve them.

There is, however, a subtle, yet critical, problem with public key cryptography. To gain insight into this problem, let's consider an Internet commerce example. Suppose that Alice is in the pizza delivery business and she accepts orders over the Internet. Bob, a pizza lover, sends Alice a plaintext message that includes his home

address and the type of pizza he wants. In this message, Bob also includes a digital signature (that is, a signed message digest for the original plaintext message). As discussed in Section 8.4, Alice can obtain Bob's public key (from his personal Web page, a public key server, or an e-mail message) and verify the digital signature. In this manner she makes sure that Bob, rather than some adolescent prankster, placed the order.

This all sounds fine until clever Trudy comes along. As shown in Figure 8.20, Trudy decides to play a prank. Trudy sends a message to Alice in which she says she is Bob, gives Bob's home address, and orders a pizza. She also attaches a digital signature, but she attaches the signature by signing the message digest with her (that is, Trudy's) private key. Trudy also masquerades as Bob by sending Alice Trudy's public key but saying that it belongs to Bob. In this example, Alice will apply Trudy's public key (thinking that it is Bob's) to the digital signature and conclude that the plaintext message was indeed created by Bob. Bob will be very surprised when the delivery person brings a pizza with everything on it to his home!



**Figure 8.20** ♦ Trudy masquerades as Bob using public key cryptography.

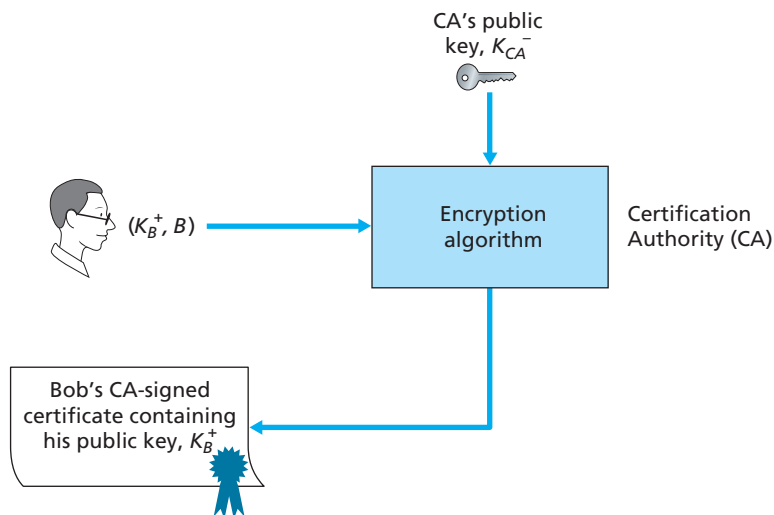
We see from this example that in order for public key cryptography to be useful, entities (users, browsers, routers, and so on) need to know *for sure* that they have the public key of the entity with whom they are communicating. For example, when Alice is communicating with Bob using public key cryptography, she needs to know for sure that the public key that is supposed to be Bob's is indeed Bob's. We had similar concerns in our authentication protocols in Figures 8.12 and 8.13.

Binding a public key to a particular entity is typically done by a CA, whose job is to validate identities and issue certificates. A CA has the following roles.

1. A CA verifies that an entity (a person, a router, and so on) is who it says it is. There are no mandated procedures for how certification is done. When dealing with a CA, one must trust the CA to have performed a suitably rigorous identity verification. For example, if Trudy were able to walk into the Fly-by-Night CA and simply announce "I am Alice" and receive certificates associated with the identity of Alice, then one shouldn't put much faith in public keys certified by the Fly-by-Night certificate authority. On the other hand, one might (or might not!) be more willing to trust a CA that is part of a federal—or state—program (for example, Utah licenses CAs in that state [Utah 2004]). You can trust the identity associated with a public key only to the extent that you can trust a CA and its identity verification techniques. What a tangled web of trust we spin!
2. Once the CA verifies the identity of the entity, the CA creates a **certificate** that binds the public key of the entity to the identity. The certificate contains the public key and globally unique identifying information about the owner of the public key (for example, a human name or an IP address). The certificate is digitally signed by the CA. These steps are shown in Figure 8.21.

Let us now see how certificates can be used to combat pizza-ordering pranksters, like Trudy, and other undesirables. When Alice receives Bob's order, she gets Bob's certificate, which may be on his Web page, in an e-mail message, or in a certificate server. Alice uses the CA's public key to check the validity of Bob's CA-signed certificate. If we assume that the public key of the CA itself is known to all (for example, it could be published in a trusted, public, and well-known place, such as the *New York Times*, so that it is known to all and cannot be spoofed), then Alice can be sure that she is indeed dealing with Bob. Figure 8.21 illustrates the steps involved in CA-mediated public key encryption. You can view the CA certificates stored in your Netscape browser by choosing Communicator, Tools, Security Info, Certificates; in Internet Explorer, choose Tools, Internet Options, Content, Certificates.

Both the International Telecommunication Union (ITU) and the IETF have developed standards for certificate authorities. ITU X.509 [ITU 1993] specifies an authentication service as well as a specific syntax for certificates. [RFC 1422] describes CA-based key management for use with secure Internet e-mail. It is compatible with X.509 but goes beyond X.509 by establishing procedures and



**Figure 8.21** ♦ Bob obtains a certificate from the CA.

conventions for a key management architecture. Table 8.3 describes some of the important fields in a certificate.

With the recent boom in e-commerce and the consequent widespread need for secure transactions, there has been increased interest in certificate authorities.

Field Name	Description
Version	Version number of X.509 specification
Serial Number	CA-issued unique identifier for a certificate
Signature	Specifies the algorithm used by CA to sign this certificate
Issuer Name	Identity of CA issuing this certificate, in distinguished name (DN) [RFC 2253] format
Validity period	Start and end of period of validity for certificate
Subject name	Identity of entity whose public key is associated with this certificate, in DN format
Subject public key	The subject's public key as well as an indication of the public key algorithm (and algorithm parameters) to be used with this key

**Table 8.3** ♦ Selected fields in a X.509 and RFC 1422 public key

Among the companies providing CA services are Digital Signature Trust Company [Digital Signature 2004] and Verisign [Verisign 2004].

A certificate issued by Thawte Consulting to Netfarmers Enterprises, Inc., as viewed through a Netscape browser, is shown in Figure 8.22.

## 8.6 Access Control: Firewalls

We've seen throughout this chapter that the Internet is not a very safe place—bad guys are out there, wreaking all sorts of havoc. From a network administrator's point of view, the world divides quite neatly into two camps—the good guys (who belong to the organization administering the network, and who should be able to access resources inside the administrator's network in a relatively unconstrained manner) and the bad guys (everyone else, whose access to network resources must be carefully scrutinized). In many organizations, ranging from medieval castles to modern corporate office buildings, there is a single point of entry/exit where both good guys and bad guys entering and leaving the organization are security-checked. In a castle, this was done at a gate at one end of the drawbridge; in a corporate building, this is done at the security/reception desk. In a computer network, when traffic entering/leaving a network is security-checked, logged, dropped, and/or forwarded, it is done at a device known as a firewall.



**Figure 8.22** ♦ A certificate issued by Thawte Consulting to Netfarmers Enterprises, Inc.