CHAPTER NINE

SECURE SHOPPING SETUP

Ultimately, the Web store deals with money that needs to get safely from your customer to you. Thus, it is imperative that security is implemented in any online store. There are two main areas of security to be concerned about. First, we want to make sure that the user's financial information/credit cards do not travel over the Internet to the Web server unencrypted and second, once we have the user's financial information, we want to make sure that it stays secure on our end.

Once users have chosen the items that they wish to order, they need to actually pay for them. Unfortunately, traditional methods of payment such as check, money order, or cash cannot be used over the Internet unless the customer finished the deal by using physical mail to send the money. The concept of sending real cash over the Internet is like faxing a pizza: it cannot be done, or worse, the process will be messy.

Thus, credit cards are a natural means of paying for items relatively instantaneously on the Web. However, with this option comes some responsibility. We should assume that customers do not want their credit card information transmitted over the Internet as plain text from the browser to the Web server where the store is housed. The first section of this chapter discusses solutions to this problem.

Finally, once the order has been completed, most store administrators like to have the order actually emailed to them. Unfortunately, because so much attention is focused on security in submitting the credit card information in the first place, little attention is usually placed on what happens with the information when it is

submitted. If your store owner's email address, for example, resides on another Internet service provider (ISP) or on another machine, emailing the order can be very insecure. Remember, email is sent as plain text. Thus, if the email includes credit card information, it can be picked up as plain text. Figure 9.1 illustrates the pipeline involved in ordering a product over the Internet and where something can go wrong. The second part of this chapter discusses how to encrypt your orders so that they will not be subject to prying eyes on the Internet.

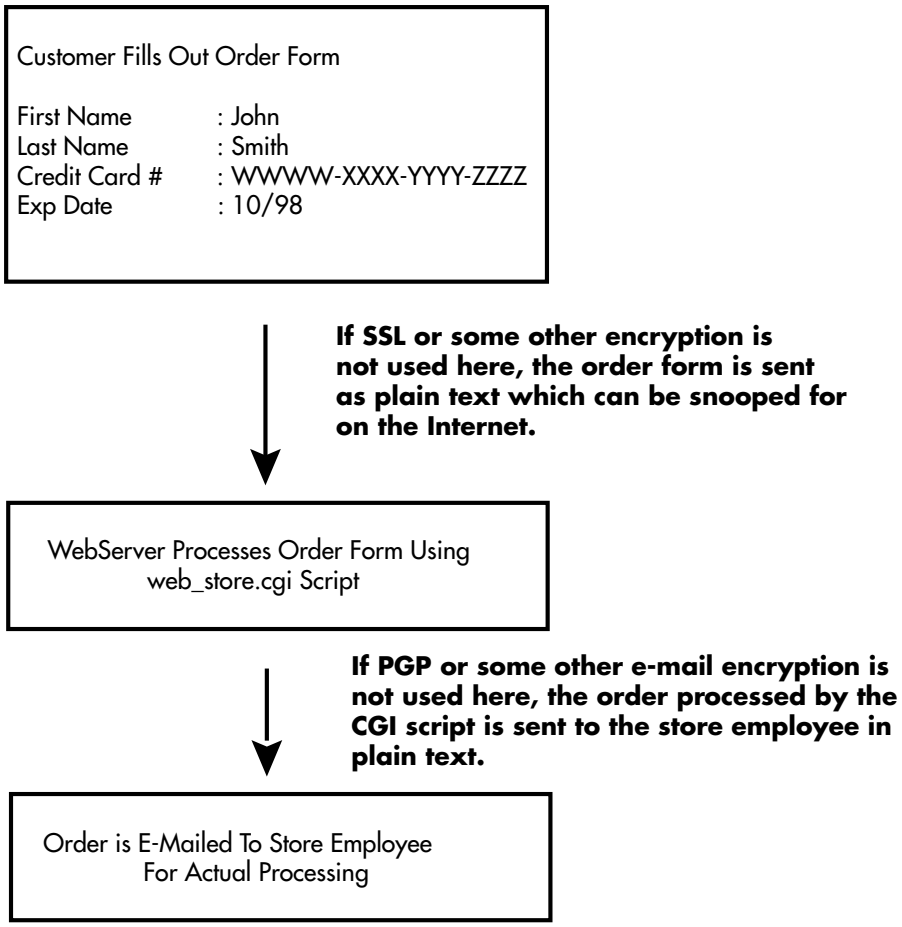

Figure 9.1 Security problems with ordering over the Internet.

Using SSL and Other Secure Protocols

Getting the credit card information from the user's Web browser to your Web store script is not that hard. All you need is a "secure" server that supports a popular Web encryption protocol such as SSL (Secure Sockets Layer). Most commercial Web servers, as well as many public domain Web servers support SSL or some other Web-related encryption protocol.

If your Web store is being hosted by an ISP, you will need to piggyback onto their existing secure server—assuming they have one. If your ISP lacks a secure server, then you are not entirely out of luck, but your options become more limited. Basically, you will need to find a third-party secure server that will accept the orders on your behalf. Sometimes, you will want this option anyway, because if you are unable to accept credit cards as a vendor, then this third party can be used to charge the customer's credit card for you and then send the money directly to your bank account. To use a third party in this way, you will need to turn on the setup variable **$sc_order_with_hidden_fields** so that the cart contents will be stored in hidden fields that can be submitted to the third party. The reason you need to use hidden fields to store cart information if a third party is involved is because they probably will not have direct access to the cart or database information. Therefore the order form must contain all the cart information as hidden fields so that all the ordering information will be sent to the third party as form data.

The problem with this approach is that it introduces a third party who may or may not be reputable, and your third party will not be able to double-check the orders against a live database of items. Remember, the Web store, data, and cart files all reside on your server, not on the third party's server. If your customers figure out that they can create their own form with "fake" items and submit it to your third party, they may be short-changing you by also faking the price and subtotal calculations that get sent to the third party as hidden variables. You may have to reconcile all your orders closely. For example, a user might easily change a price from 17.95 to 15.95 and order 100 of them at their new "discount". If you have a lot of products, it can be time consuming and difficult to reconcile these prices.

Even if your ISP has a secure server, there is no guarantee that it will be on the same physical machine as the Web server you are renting space on. If this is the case, you will need to make sure that the Web store is set up on the secure server as well, just for handling orders. Additionally, all the data-related directories from the original Web store setup should be available on the secure server as directories mounted over the network. If your ISP is using a UNIX server, making the data related directories accessable from the secure server will be done using NFS (network file system) or AFS (Andrew file system). Figure 9.2 shows an example of how your ISP might be set up if the secure server is physically located on a machine other than where the Web store is set up.

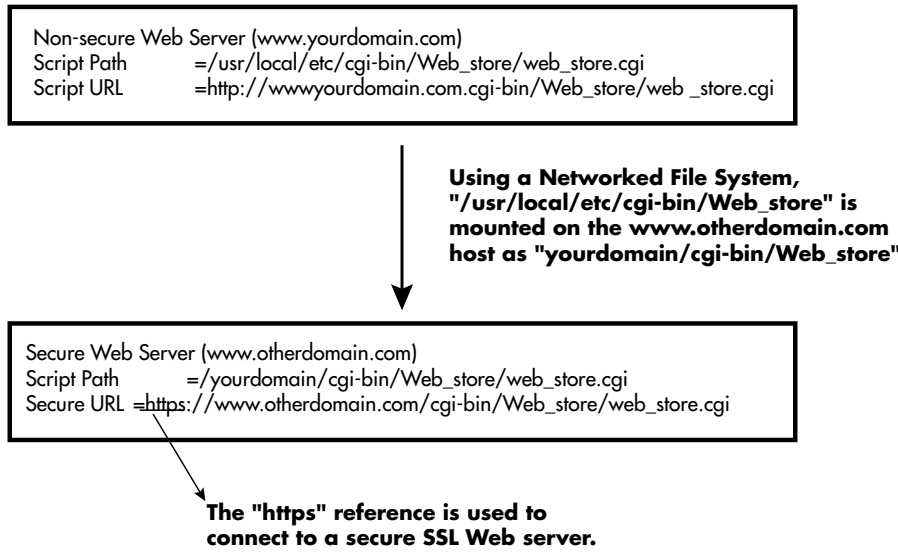

Figure 9.2 Network diagram of physically separate Web servers.

While it is more convenient to have the secure server located on the same machine as the nonsecure Web server, it usually costs more money to maintain a secure server. For example, the certificates that authorize a secure server for action have the DNS name (descriptive Internet name) registered to the certificate. Thus, even if you were running **www.buysneakers.com**, your order form would need to post to **www.yourisp.com** unless you specifically purchased a certificate for your domain name that your ISP would set up for you. Depending on how deep your pockets are, the extra cost of getting your domain name registered with a secure

certificate may or may not be a problem. Most people opt to just use the secure server that an ISP has for secure orders, and use the regular Web server for all the other Web store functions.

If you are running your own Web server, you will need to obtain a version of your Web server software that handles encryption. For example, if you are using Netscape server, you will need to obtain either Netscape Commerce Server or Netscape Enterprise Server. If you are using the freeware Apache server, you will need to obtain the special patch that lets it run the SSL protocol. Next, you will need to apply for a "secure certificate" yourself from a signature authority such as Verisign and set it up on your new secure server. Detailed instructions for obtaining a secure certificate are generally distributed with your Web server documentation.

Regardless of how you are going to access a secure server, it is important to understand what you must do in order to enable secure server processing. First of all, you do not need to run the whole Web store in secure mode. Doing so will only slow down the user's browser, since it will be encrypting their entire browsing experience. In addition, the actual order form that the user fills out does not have to be secure. When the empty form is transmitted to the customer's browser, the user enters data locally on the form. No financial information has been exchanged.

However, when the customer actually presses the **Submit Order** button, the server that the form information is posted to must be secure because now the customer's information really *is* being transmitted to the Web server. The URL that is used by the Web store to post data is in the setup file and is called **$sc_order_script_url**. If your Web server is using the SSL protocol, this variable will typically be set to the same URL as before except with an HTTPS instead of HTTP protocol signature. For example, if **$sc_main_script_url** is:

```
http://www.yourdomain.com/cgi-bin/Web_store/web_store.cgi"
```

then $sc_order_script_url should be set to something like: https://www.yourdomain.com/cgi-bin/Web_store/web_store.cgi

Of course, if your secure server is located on another machine or name, you will need to make the appropriate modifications to the **$sc_order_script_url** variable. Figure 9.3 illustrates how the hand-off works from the nonsecure order form to the secure order form processing.

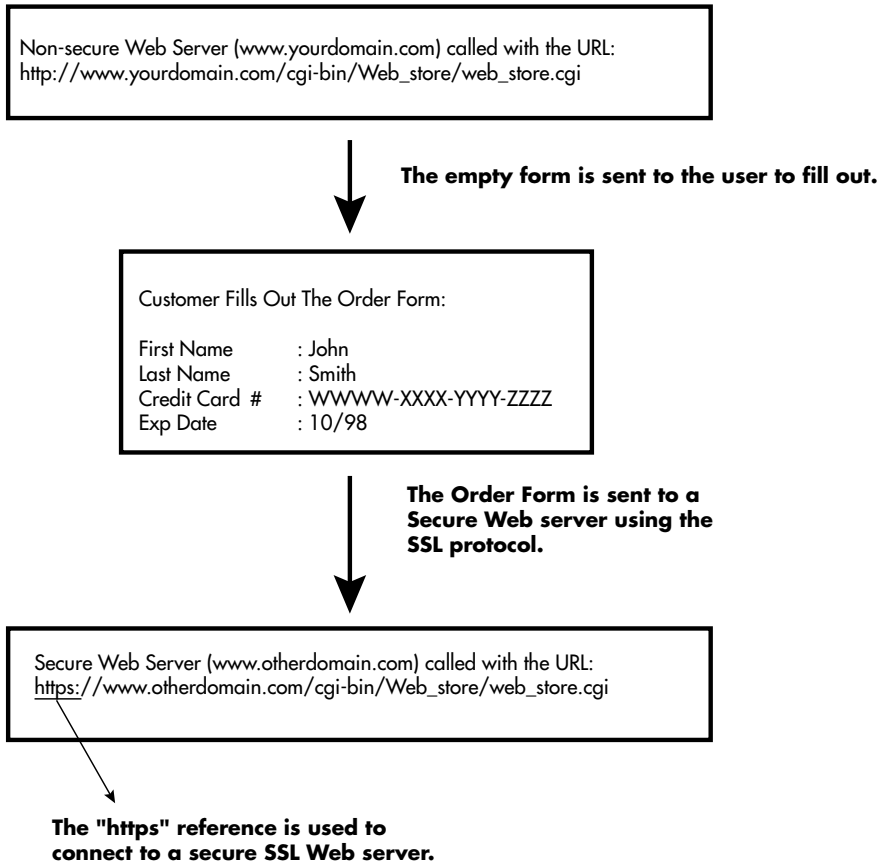

Figure 9.3 The process of moving from nonsecure to secure Web server.

In summary, to set up a secure server you must pay attention to the Setup variable **$sc_order_script_url** and make sure that it points to the secure server. Second, the secure server must have access to Web store cgi script and the related data files. If you are using a third party secure server, then **$sc_order_script_url** should point to that third-party secure server and **$sc_order_with_hidden_fields** should be set to **yes** so that the cart contents will be embedded in the order form as hidden fields that will be sent to the third-party secure server.

Setting up PGP For Emailing Orders Securely

Once the order has been sent to the Web store script, the script has several choices about how to handle the data. The script can either log the data to a file, email the data, or both. In addition, the data can be logged or emailed either encrypted (secure) or as plain text (insecure).

Logging the data to a file is not recommended, especially on a shared Web server. The likelihood is high that other people will be able to figure out how to get into the log file and get information about the orders. E-Mailing the orders is reasonable, but if that email is traveling to a machine other than where the Web server is located, then there is a possibility that someone else may be able to intercept and snoop into the email packets. Such a scenario like this would basically undo all the protection you have been trying to set up by using a secure Web server. The solution to security problems in either case is to use an encryption tool to encrypt the data.

One of the best and most widely available encryption tools is called PGP (Pretty Good Privacy), written by Phil Zimmerman. Keep in mind though, that PGP is only freeware for noncommercial use. You will have to obtain an official license to use it if you are using it for running a for-profit online store.


NOTE

The main PGP tool is useful only if you are inside the boundaries of the United States or Canada. At the time of this writing, the U.S. government is very particular about the exportation of encryption technology, so none of the sites that carry PGP allow people to download it outside this continent.

If you are in Europe, Australia, or some other country, then you will need to get PGP 2.6ui. The "ui" version is developed outside of the U.S. and is fully compatible with the regular PGP v2.6 inside the United States. Yes, it does seem kind of silly that the United States does not allow the exportation of technology already widely available outside of our boundaries. For more details regarding the massive policy debate surrounding the export of cryptography, see http://www.eff.org/pub/Privacy/.

Why do we want to use PGP instead of another encryption tool such as "crypt"? The UNIX crypt tool was simply not made for high security and its encryption can easily be broken. In addition, to use crypt and most other encryption tools, you have only one password, which is used as a key for both encrypting and decrypting. If the same key is used on your Web server to encrypt the messages, then it is possible that someone snooping around may find that key and use it to decrypt your log file or emails. It is inherently insecure to use the exact same key to both encrypt and decrypt messages.

PGP uses a different mechanism. It actually breaks up the key into two parts: a public key and a private key. You keep the private key on your local machine. However, you take your public key and configure the Web server so that it encrypts the email using your public key. Even if someone else gets a hold of your public key, they will not be able to decrypt any of your messages. Once something is encrypted with the public key, it can be decrypted only by using the private key. Figure 9.4 illustrates the use of public/private keys.

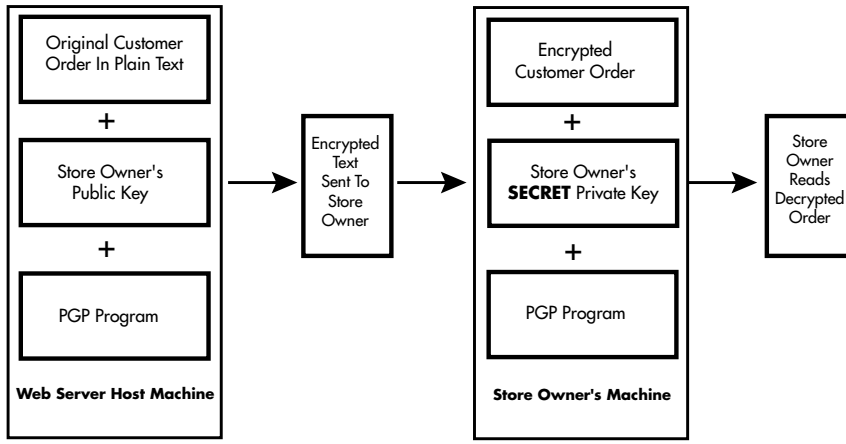

Figure 9.4 The use of public/private keys.

In fact, when you create a private key for yourself using PGP, you typically want as many people as possible to get your public key. Then, all your friends can send you messages that are encrypted so that only you can decrypt them with your private key. Likewise, you want to obtain your friend's public keys so that you can send them encrypted replies that only they can decrypt. Without the special mechanism

of having two-part keys, you would always have to keep all your keys secret. With public/private key encryption, you only need to keep your one private key a secret. A detailed discussion of how private/public key encryption works can be found in O'Reilly & Associates, Inc., PGP: Pretty Good Privacy, by Simson Garfinkel.

To set up the Web store with PGP, you need to generate a private key for yourself and the corresponding public key that the Web server will use to encrypt email. In addition, you will need to set up the PGP related variables and do some editing to **pgp-lib.pl** in order to set it up for interfacing with pgp on your system.

Setting up a PGP Receiver (Private-Key)

The first thing that you need to do to use PGP is set up your private key on the machine you expect to be receiving email on. If you are using Windows/DOS at home to dial in to your ISP, then the first thing you need to do is download the DOS version of PGP. As of this writing, the latest copy of PGP is located at http://web.mit.edu/network/pgp-form.html. This is a form that makes you acknowledge that you are a United States citizen or someone else who is legally allowed to download PGP. The International version of PGP can be found at http://www.ifi.uio.no/~staalesc/PGP/ if you are outside the United States. As of this writing, if you are planning on using PGP for commercial purposes, you can purchase it from Pretty Good Privacy, Inc. which is located at http://www.pgp.com/.

Although you can obtain a Windows GUI version of PGP, the instructions here work for the command-line MS-DOS and UNIX versions. If you prefer using the Windows GUI, the concepts are the same, except that you use menus and buttons to do the same thing as command-line options.

When you download PGP for DOS, you will receive it as a normal zip file. You should unzip this file inside a subdirectory called **pgp262**. Unzipping this file will reveal another zip file. If you are using the DOS version of PKUNZIP, use the **-d** option to unzip the file with full directory names. A sample session for installing PGP on a DOS machine appears below. The characters that you type appear in bold:

```
C:\>mkdir pgp262
C:\>cd pgp262
C:\pgp262\>pkunzip ..\pgp262.zip
  Exploding: setup.doc
```

```
 Extracting: pgp262i.asc
 Extracting: pgp262i.zip
C:\pgp262\>pkunzip -d pgp262i.zip
  Exploding: config.txt
  Exploding: doc/pgformat.doc
  Exploding: doc/keyserv.doc
  Exploding: doc/setup.doc
  Exploding: doc/pgpdoc2.txt
  Exploding: doc/pgpdoc1.txt
  Exploding: doc/politic.doc
  Exploding: doc/appnote.doc
  Exploding: doc/changes.doc
  Exploding: doc/blurb.txt
  Exploding: es.hlp
  Exploding: fr.hlp
  Exploding: keys.asc
  Exploding: language.txt
  Exploding: mitlicen.txt
  Exploding: pgp.exe
  Exploding: pgp.hlp
  Exploding: readme.doc
  Exploding: rsalicen.txt
C:\pgp262\>dir

 Volume in drive C is MAIN
 Volume Serial Number is 1F3A-09FE
 Directory of C:\pgp262
.              <DIR>        12-15-96  2:31p .
..             <DIR>        12-15-96  2:31p ..
SETUP    DOC        16,253  10-22-94  6:53p SETUP.DOC
PGP262I  ASC           293  10-22-94  7:38p PGP262I.ASC
PGP262I  ZIP       275,146  10-22-94  7:38p PGP262I.ZIP
CONFIG   TXT         4,042  10-11-94  5:26p CONFIG.TXT
DOC            <DIR>        12-15-96  2:33p DOC
ES       HLP         4,379  05-06-94  3:58p ES.HLP
FR       HLP         4,467  05-06-94  3:58p FR.HLP
KEYS     ASC         5,895  09-03-94 12:52a KEYS.ASC
LANGUAGE TXT        70,744  05-23-94  6:40p LANGUAGE.TXT
MITLICEN TXT         2,589  05-24-94 11:56a MITLICEN.TXT
PGP      EXE       243,097  10-22-94  7:37p PGP.EXE
PGP      HLP         3,983  06-19-94  5:15p PGP.HLP
README   DOC         6,768  10-18-94  5:38p README.DOC
RSALICEN TXT         7,630  05-23-94 10:39p RSALICEN.TXT
        13 file(s)        645,286 bytes
         3 dir(s)      49,987,584 bytes free
C:\pgp262>
```

Now, you should be ready to use PGP on your DOS machine. The installation for UNIX is generally more difficult and involves compiling PGP from source-code files. The installation for UNIX also differs depending on the flavor of UNIX you have. It is recommended that your ISP should have previously installed PGP for you if you feel uncomfortable with compiling public-domain programs on your UNIX server.

The next step is to actually generate your very own private key. PGP contains a list of "k" command line options used to maintain your keys. Typing pgp -k at the command line will give you a list of these options. The sample output from **pgp -k** appears below:

```
C:\pgp262\pgp -k
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1996/12/15 22:45 GMT

Key management functions:
To generate your own unique public/secret key pair:
   pgp -kg
To add a key file's contents to your public or secret key ring:
   pgp -ka keyfile [keyring]
To remove a key or a user ID from your public or secret key ring:
   pgp -kr userid [keyring]
To edit your user ID or pass phrase:
   pgp -ke your_userid [keyring]
To extract (copy) a key from your public or secret key ring:
   pgp -kx userid keyfile [keyring]
To view the contents of your public key ring:
   pgp -kv[v] [userid] [keyring]
To check signatures on your public key ring:
   pgp -kc [userid] [keyring]
To sign someone else's public key on your public key ring:
   pgp -ks her_userid [-u your_userid] [keyring]
To remove selected signatures from a userid on a keyring:
   pgp -krs userid [keyring]
```

The first option (**-kg**) is what you want to use in order to generate a private/public key pair. The following is a sample session of making a key. The characters that you type appear in bold:

```
C:\pgp262\>pgp -kg
WARNING: Environmental variable TZ is not defined, so GMT timestamps
may be wrong.  See the PGP User's Guide to properly define TZ in
```

```
AUTOEXEC.BAT file.
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct
94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security,
Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1996/12/15 22:47 GMT
Pick your RSA key size:
             1)  512 bits- Low commercial grade, fast but less
                     secure
             2)  768 bits- High commercial grade, medium speed,
                     good security
       3)  1024 bits- "Military" grade, slow, highest
                     security
Choose 1, 2, or 3, or enter desired number of bits: 3
Generating an RSA key with a 1024-bit modulus.

You need a user ID for your public key.  The desired form for this
user ID is your name, followed by your E-mail address enclosed in
<angle brackets>, if you have an E-mail address.
For example:  John Q. Smith <12345.6789@compuserve.com>
Enter a user ID for your public key:
yourname

You need a pass phrase to protect your RSA secret key.
Your pass phrase can be any sentence or phrase and may have many
words, spaces, punctuation, or any other printable characters.

Enter pass phrase:yourpassword
Enter same pass phrase again:yourpassword
Note that key generation is a lengthy process.

We need to generate 1016 random bits.  This is done by measuring the
time intervals between your keystrokes.  Please enter some random text
on your keyboard until you hear the beep:
1016[RANDOM TEXT ENTERED]

key generation completed.
C:\pgp262\>
```

That's all there is to it. You are now ready to receive and decrypt email. If you receive something that is encrypted, use the command that follows. Let's assume that you received something that was encrypted. The following code

outlines steps for decrypting the file along with a quick example as well as a display of what the encrypted data actually looks like before the decryption is done. As before, the characters you type appear in bold. The main step is simply to type **pgp** followed by the filename that you wish to decrypt. But first, the encrypted file sample is displayed using the DOS **type** command:

```
C:\pgp262\>type order.txt
-----BEGIN PGP MESSAGE-----
Version: 2.6.2

hEwDS3ppBLlMyp0BAf90zBPMdkJ3IxEOCrUqnfb7im/U1ilDr2
1s2apl61RnXSvS5k+SrCszHlpE5FHgflsLYoI2tspgAAAbvm7/
+jyMdI8Ob99e06GrjecBSA2ZBz8tYbcYI19OqxJApbmeFE/E/D
XzdBxFko7uCZZCOGr8yW5gU/tPhuqDRTDOoejRJQUf/Z+IH/+W
bwkDyXB+zw4OLWY5Jaest59fTlzaaLAUmKk3s1Mcb7ZiPzZtAo
lakcnlNnRqmM9H5n2z5OYT0i/bcCQgT7WCwwAW1+2rnsa16f44
5eFJaTw7aF6O4ou0B4vlRRu44UQqCwnsOgIdP+++nVRW2r1Nnz
6PgB3Gu2YdmQpHyM4hD3JXjxGoSS9ndZm2xUkOpBkiek1frXpp
X8zcuuBJIVehFwb4J50ptoAVg5F1Mu4nohq0/Va5zBKGZE23Td
6/UGEyj9p1Gz7ex/m3VAU2Uspdy0NAGmvGHI0P2proKW/h6sa0
02TB8gZTO0eCgNTBr/G4uFXYVs8b8y7JO5gok+NSXtKU/2sQIt
C5FwhrxCZND+q74y0oiycacJD8QHFkJ9yMrZFX+WO9lAT1oyoa
3XuJAD0GWAbdzbkXcUTw===jUwS
-----END PGP MESSAGE-----
```

The step below actually performs the decryption:

```
C:\pgp262\>pgp order.txt
WARNING: Environmental variable TZ is not defined, so GMT timestamps
may be wrong.  See the PGP User's Guide to properly define TZ in
AUTOEXEC.BAT file.
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct
94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security,
Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1996/12/15 23:01 GMT

File is encrypted.  Secret key is required to read it.
Key for user ID: yourname
1024-bit key, Key ID C669795D, created 1996/12/15
```

```
You need a pass phrase to unlock your RSA secret key.
Enter pass phrase: yourpassword
Pass phrase is good.  Just a moment......
Plaintext filename: order
```

After the file has been decrypted, the **type** command is used to look at the resulting file:

```
C:\pgp262>type order
Description         = The letter A
Options             = Times New Roman 0.00, Red 0.00
Price After Options = $15.98

Description         = The letter E
Options             = Times New Roman 0.00, Red 0.00
Price After Options = $12.98

Subtotal:           = $171.74

Shipping:           = $5.00

Discount:           = $1.00

Sales Tax:          = $8.60

Grand Total:        = $184.34
C:\pgp262>
```

At this point you would have your decrypted order in hand. The next step is to set up your account on the Web server so that it can encrypt files using your public key. Before we configure this, we need to export the public key from your "key ring" on the DOS machine. This is done using the command line option (**-kxa**). **kx** extracts the key and **a** tells pgp to export the key as ASCII text to make it easier to transfer from machine to machine:

```
C:\pgp262\pgp -kxa
WARNING: Environmental variable TZ is not defined, so GMT timestamps
may be wrong.  See the PGP User's Guide to properly define TZ in
AUTOEXEC.BAT file.
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct
94
```

```
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security,
Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1996/12/15 23:10 GMT

A user ID is required to select the key you want to extract.
Enter the key's user ID:yourname

Extracting from key ring: 'pubring.pgp', userid "yourname".

Key for user ID: yourname
1024-bit key, Key ID C669795D, created 1996/12/15

Extract the above key into which file? yourname.asc

Transport armor file: yourname.asc

Key extracted to file 'yourname.asc'.
C:\pgp262\>type yourname.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.2
mQCNAzK0gWEAAAEEAO5CCH8cYH99dECtptItvPEyHUmbo8DLtbrG7rfMuphlxW5j58eYYT
LInVUSLpQQXAea6k7f6uuWYhT8ofbKWKT0YNb1x8YC++EoR6tLxYYsEY11Yj2cbZTng4b9
z/oXLL1BCHkrvXjD5D3dtuBUH/9BwOC55yK0U0VGhonGaXldAAURtAh5b3VybmFtZQ===7
BFj
-----END PGP PUBLIC KEY BLOCK-----
C:\pgp262\>
```

Now that you have your public key extracted from your key ring, all you need to do is transfer it to the machine that is housing your Web server. At this point, we will switch to assuming that your Web server is running on a UNIX-based machine and has PGP configured for it. If you are using a Windows NT based server, the steps to take would be similar with changes made that are unique to the DOS version of PGP.

Setting Up a PGP Sender (Public-Key)

On your UNIX system, you need to initially set up your PGP files. First, set up a separate dummy private key on your UNIX machine account. This will let your UNIX account act as a real PGP user that can encrypt messages with

other user's public keys. You set up the private account by using the (**-kg**) option in the same manner that was explained before. After you have set up a private account, you are ready to import the ASCII public key into your UNIX server account's key ring. You do this using the (**-ka**) parameter:

```
UNIX:/home/yourhome>pgp -ka yourname.asc
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct
94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security,
Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1996/12/15 20:19 GMT

Looking for new keys...
pub  1024/C669795D 1996/12/15  yourname

Checking signatures...

Keyfile contains:
   1 new key(s)

One or more of the new keys are not fully certified.
Do you want to certify any of these keys yourself (y/N)?y
Key for user ID: yourname
1024-bit key, Key ID C669795D, created 1996/12/15
Key fingerprint =  3E D9 D0 89 A3 53 5F 44  3C DE AD 43 36 70 C2 9F
This key/userID association is not certified.

Do you want to certify this key yourself (y/N)?y

READ CAREFULLY:  Based on your own direct first-hand knowledge, are
you absolutely certain that you are prepared to solemnly certify that
the above public key actually belongs to the user specified by the
above user ID (y/N)?y
You need a pass phrase to unlock your RSA secret key.
Key for user ID "gunther"

Enter pass phrase: Pass phrase is good.  Just a moment....
Key signature certificate added.

Make a determination in your own mind whether this key actually
```

```
belongs to the person whom you think it belongs to, based on available
evidence.  If you think it does, then based on your estimate of that
person's integrity and competence in key management, answer
the following question:

Would you trust "yourname"
to act as an introducer and certify other people's  public keys to
you?
(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ?4
UNIX:/home/yourhome>
```

The next step is to test your account to make sure that you can encrypt files with your public key. In this case, assume we have made a dummy text file called **foobar.txt**.

```
UNIX:/home/yourhome>pgp -feat <foobar.txt >foobar.enc
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct
94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security,
Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1996/12/15 20:29 GMT


A user ID is required to select the recipient's public key.
Enter the recipient's user ID: yourname


Key for user ID: yourname
1024-bit key, Key ID C669795D, created 1996/12/15
UNIX:/home/yourhome>cat foobar.enc
-----BEGIN PGP MESSAGE-----
Version: 2.6.2


hIwDRUaGicZpeV0BBACZ3ek41DmwQF3lhWJMWetf+lO9YOU2Y9Q86f27b1GSfjX+HsDh6l
jzz/TqunTFsAXY2H1vTIwCi+M8P4z72Z2t7Bpfw0YnN52lW2E31JDXk6xOjsjF8FRkNGfo
ViIPN4iSIQ13KmKK9twles1qkHGsluFnfu7IFuFoGzsXDEeTj6YAAAAtF4tdHVH0cNtgqi
uYurM3qeNiUNj0suEzwjAciNl342It5qfIjX5PhG02Y2gB=/+bR
-----END PGP MESSAGE-----
UNIX:/home/yourhome>
```

If printing out **foobar.enc** reveals a message similar to what is shown immediately above, then your environment is set up correctly for encrypting files. The next step is to set up your Web server so that it can encrypt files using the PGP information that you just set up.

Setting up the PGP Sender Files (Public-Key) on Your Web Server

By default, your PGP-related files are stored in a hidden subdirectory under your UNIX home directory called **.pgp**. Your UNIX home directory is typically the directory you start in when you first telnet into your UNIX server.


NOTE

The directory name for the PGP files is called **.pgp** by default. The period in front of the directory name is important because it tells UNIX to hide the file when you do a normal directory listing. If you wish to see hidden files, use the (**-a**) command-line parameter **ls** to list all files.

Under the **Web_store** distribution, there is a directory called **Pgpfiles**. Take the resulting files from your **.pgp** directory and copy them into the **Pgpfiles** directory. To do this on UNIX, make sure you are currently in the **Pgpfiles** directory under **Web_store** and issue the command **cp ~/.pgp/* .** The **cp** command copies all files under the **.pgp** directory under your home directory (indicated by a tilde (~) into the current directory (indicated by a single period).

The files that can be found in the **.pgp** directory are **config.txt**, **pubring.bak**, **pubring.pgp**, **randseed.bin**, and **secring.pgp**. Although you are going to copy all these files to the **Pgpfiles** directory, recall that the setup described here only specified making your real private key on your DOS machine. You export only the public key from your DOS machine and import it to your UNIX account. Your UNIX account should not have your private key in it when you copy the files to the **Pgpfiles** directory.

Make sure that the Web server has permission to read these files using the instructions previously illustrated in Chapter 1. Finally, edit the **pgp-lib.pl** file to make sure that the settings are correct for your Web server. Specifically, you must edit the following variables:

- **$pgp_path** is the path and filename where the PGP executable is located.
- **$pgp_public_key_user_id** is the username of the public key you are going to use to encrypt the data. To be consistent with the examples given here, we would make **$pgp_public_key_user_id = "yourname"**.
- **$pgp_config_files** is the full path to where the Web_store/Pgpfiles directory is located on your UNIX server.

Finally, you must edit the Setup file and change the PGP-related variables to suit your environment. The following is a list of PGP-related variables in the Setup file:

- **$sc_pgp_lib_path** is the path and filename where **pgp-lib.pl** is located.
- **$sc_use_pgp** is set to yes if you wish to start using PGP encryption.
- **$sc_pgp_temp_file_path** is set to the path where you wish to store PGP files temporarily. You must set this to a directory that is writable by the Web server. The default is to store the PGP temporary files inside the **Admin_files** directory.

Summary

After all these steps, you should finally be able to use PGP to encrypt your orders. Keep in mind that the Web server account, not your account, is running the CGI scripts. Your Web server must have permission set up so that it can access the PGP executable as well as the PGP configuration files.