Laboratory_09: GPG Encryption: A Comprehensive Guide to Securing Data Transfers

Step 1: Setting Up GPG Keys

Generating Keys on the Server

To encrypt data, the server needs a key pair:

```
$ gpg --full-generate-key
```

- Follow the prompts to:

Select key type: choose “RSA and RSA”.

```
$ gpg --full-generate-key
Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? (1)
```

- Specify key size: a 4096-bit key is recommended for maximum security.

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (4096)
```

- Set an expiration date: optional but enhances security.

```
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (365)
```

- Enter user details: provide an email address for key identification.

```
GnuPG needs to construct a user ID to identify your key.
```

Real name: The Flying Gibbon
Email address: the-flying-gibbon@example.com
Comment: Medium
You selected this USER-ID:
"The Flying Gibbon (Medium) <the-flying-gibbon@example.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?

Now, you'll be prompted to set a passphrase to secure the private key: use a strong and memorable passphrase to ensure security.

After generation, export the server's public key to share with the client:

```
gpg --export --armor the-flying-gibbon@example.com > server-public.key
```

Key Management Best Practices

- Backup private keys: store them securely in an offline location.
- Use strong passphrases: protect the private key with a robust passphrase.
- Key revocation: generate a revocation certificate to disable the key if it's compromised.

```
gpg --gen-revoke the-flying-gibbon@example.com > revocation-certificate.asc
```

Step 2: Encrypting the File on the Client

Importing the Server's Public Key

The client imports the server's public key to encrypt data:

```
gpg --import server-public.key
```

Encrypting a File

Suppose the client needs to send a file named sensitive_data.txt:

```
gpg --encrypt --recipient the-flying-gibbon@example.com sensitive_data.txt
```

This generates an encrypted file, sensitive_data.txt.gpg: the original file remains untouched.

File Encryption Options

To include the original filename in the encrypted file (for easier identification or to hide original extension of the file):

```
gpg --encrypt --recipient the-flying-gibbon@example.com --output sensitive_data.gpg
sensitive_data.txt
```

For added security, compress the file before encryption:

```
tar -czf sensitive_data.tar.gz sensitive_data.txt gpg --encrypt --recipient the-flying-
gibbon@example.com sensitive_data.tar.gz
```

Step 3: Transferring the File via SFTP

The encrypted file can now be transferred securely:

```
sftp user@server.com <<EOF
put sensitive_data.txt.gpg
bye
EOF
```

SFTP Security Considerations

- SSH configuration: use strong authentication methods like SSH keys instead of passwords.
- Limit permissions: restrict SFTP users to specific directories using chroot.
- Audit logs: monitor file transfer activity for suspicious behavior.

Step 4: Decrypting the File on the Server

Once the file is transferred, the server decrypts it using the private key:

```
gpg --decrypt sensitive_data.txt.gpg > sensitive_data.txt
```

If you specified a passphrase during the key pair generation, GPG will prompt you to enter the passphrase during the decryption process with a message similar to this:

```
gpg: key 1234ABCD: public key decryption failed: bad passphrase
Enter passphrase:
```

If the file is correctly encrypted, the passphrase is correct and the private key matches, the server will recover the original file.