EE487: Applications of Cyber Engineering Name:
SX-14: Asymmetric Encryption

Discussion: There are relatively few completely transformative technologies that come along from time to time (due to the work of very smart people). The transistor is one of those, and another is clearly public-private key encryption, more precisely known as asymmetric encryption. Diffie and Hellman first published an asymmetric scheme in 1976. Without this mathematical break-through, the modern World Wide Web probably couldn't exist in its current form. You use asymmetric encryption many times every day and probably don't even realize it.

We learned in the last lab that symmetric encryption has a fatal flaw. Even if we have an unconditionally secure cipher, we still have to find a way to get the key to the receiver without anyone else intercepting it. If the person is across the room, it's doable. If they're across the world or somewhere out on the Internet, it's a real problem. Enter asymmetric encryption. Now we create two mathematically related keys. One we keep very secret and protected (usually we actually encrypt that key again with another password just to keep it really secure). The other we publish to the entire world. Anyone can encrypt anything they want with the public key – only the person who holds the private key can decrypt it. As in all engineering, however, there is a trade-off. This method of encryption is much more complicated and consumes much more processor cycles to perform. Next week we'll learn how we solve that problem.

Today we're going to use the gpg program as last time, but now we're going to use asymmetric keys (public-private key pairs). It's called asymmetric because the two parties (sender and receiver) do not use the same key. However, the keys are mathematically related and must be generated using a special algorithm. The folks who dreamt up these algorithms are generally very smart and probably wealthy now.

Setup: Open your personal Ubuntu Virtual Machine (VM) using the VSphere Client. You will have sudo NOPASSWD privileges on your machine, so be careful! You will also want to use a web browser from your Windows host operating system and navigate here: http://rona.cs.usna.edu/~si110/resources/rsa/

Reading: Erickson pages 400 to page 404 (sections 0x740 and 0x741).

EE487: Applications of Cyber Engineering Name:
SX-14: Asymmetric Encryption

Section 1: **RSA**

1. Let's first explore some RSA key pairs. Go to the website referenced above and navigate to the given hyperlink. This is a web page that does the calculations that Erickson tells you about in section 0x741.

    a. Hit the "Generate Key" button. You should get a key pair in hex format. One is public and one is private. Which one is which?

    b. What would happen if you reversed them? Would it still work?

    c. How many bits are in each?

    d. Is RSA unconditionally secure? Why or why not?

    e. What is the only way that we can make RSA more secure, and why?

2. Now paste the **public key** (don't mix them up) into the "*Key*" block and type a message of your choice, then hit "*Encrypt Input*".

    a. What happened? How many bits is the output? How does that compare to your original message?

    b. Do you think you could use an analysis method (like counting the number of certain letters) to break this code?

c. If you intercepted this cipher-text and wanted to crack it with brute-force, how many possible keys would you have to try?

d. Now paste the cipher-text into the message block, and replace the key with the **private key** and hit "*Decrypt Input*". Did it work?

e. Try decrypting this message with the other (**public**) key and see what happens. Can you find a way to decrypt the message without using the **private key**? What does this tell you about the **private key**?

Section 2: **GNU Privacy Guard (GPG)**

3. GPG can be used in many modes and has several possible encryption ciphers. Remember that GPG is just a program that implements ciphers invented by various other people or entities.

   a. Usually, the cipher itself is not kept secret at all; it actually makes the cipher more secure. Can you figure out why? Hint: think "crowd-sourcing" and also consider that Microsoft's source code is NOT open-source (publicly available), while Linux source code is.

   b. What definitely has to be kept secret?

   c. Use the command
      ```
      gpg --version
      ```
      What are the possible <u>asymmetric</u> (public key) ciphers? Don't let this confuse you. We're just showing that there is more than one cipher that can be used depending on the situation.

d. Could we use AES for asymmetric encryption? Can GPG use RSA?

e. What is usually done is that each person gets a public-private key pair just for their personal use. Actually, any object can have a key pair, including a computer for example. That's how we can identify a machine or a web site from across the Internet. Your classmate next to you will generate his own unique key pair, and you **must not** ever share the **private key** with anyone. Do you remember how long it took for your CAC to be created? Take a guess at what is embedded in that chip on the card.

f. Let's generate a key pair for your use. Don't worry about the names of the algorithms – just realize that there are several that can be used depending on the situation. Type the following command to generate key pairs following the prompts and using the default values (hitting enter). Use your first and last name as the Real name. Enter your USNA email address, when prompted. We will continue to use this key for class, so follow these instructions and make sure you **remember your passphrase**! ***Write down the public key fingerprint when done***.

```
gpg --gen-key
```

FINGERPRINT:

g. What effect does the key size have and what is the trade-off (why not always use the largest possible key)?

h. Some very confusing things just happened, but hopefully you see that the program used a cipher algorithm to generate two key pairs. These are actually a modern variation of the original Diffie-Hellman cipher, but it's now using DSA and ElGamal ciphers. It also had you type a passphrase to keep the private key very private (so the private key actually gets encrypted again). These keys have to be stored somewhere. Type the command

```
gpg -k
```

and note what you see. Where is the public key stored (full path)?

i. Can you figure out where the private key is stored?

4. Now let's export your public key.

   a. Use the command
      `gpg --armor --export mXXXXXX`
      where XXXXXX is your alpha code. The "armor" switch encodes the output in ASCII format for display; otherwise it would produce a pure bit-stream of gibberish (to you at least).

   b. Let's use gpg to output to a file instead:
      `gpg --output mXXXXXX.gpg --export mXXXXXX`
      You should now have a new file in that directory.

   c. Upload it to your webroot folder and run your server so that everyone can have access to it.

   d. Once your intended message recipient lab partner does the same thing, find his gpg file and download it to your machine.

   e. Navigate to the directory where you downloaded his key, then import his public key to your "keyring" (*pubring.gpg*) using this command:
      `gpg --import mYYYYYY.gpg`
      where YYYYYY is your recipient's alpha code. You could do this for all of your classmates if you had the time. Now do
      `gpg -k`
      again and you should see both your public key and his.

   f. Why do you need to have the recipient's public key?

   g. Do they need yours to access a file you send to them?

5. Now we have to <u>validate</u> the key. This is a CRUCIAL step to build trust in the identity behind the key. If you fail to do this, the validity of the entire scheme is compromised.

   a. If we don't do this, what would prevent a hacker from quickly deleting your classmate's public key and replacing it with his own?

b. This is where the fingerprint comes into play. You need to verify with your classmate that the fingerprint is the same on both ends. To do this, you have to just ask them, over the phone, in person, etc., in some manner that assures you that you are actually talking to them and not an imposter. Type
   `gpg --edit-key mXXXXXX`
   then type
   `fpr`
   at the prompt. Now verify with the intended recipient that the fingerprint is correct. If it is, type
   `sign`
   at the prompt and follow the instructions. This puts this key in your "trusted" keyring, meaning that you are CERTAIN that it belongs to the recipient. Now type "**q**" to return to your normal prompt.

c. What would happen if you put a public key in your keyring that was actually an imposter's public key?

d. Do you have to keep the fingerprint secret?

6. Now let's pass a message to a classmate like we did last class. Create a message file using nano. The command necessary to encrypt a message is:
   `gpg -e -r mYYYYYY` ***`message`*** **(Replace 'message' with name of file)**

   a. Was it easier or harder to set up this asymmetric scheme as compared to symmetric? Why?

   b. Is it going to be easier or harder to pass the message as compared to symmetric? Why?

   c. On your intructor's command, go ahead and run the exercise just like you did on the last lab. The command to decrypt a message is:
      `gpg -d` ***`message`*****`.gpg`**

EE487: Applications of Cyber Engineering Name:
SX-14: Asymmetric Encryption

Conclusion and Results:

Your typed lab report will consist of two paragraphs, in the first paragraph:
• Briefly describe what you did in the lab in your own words.
• Discuss something new that you learned.

In the second paragraph, answer the questions:
• How could an adversary use this knowledge or these tools for malicious purposes?
• How could you use your new understanding to protect your systems and personnel from attack?

Staple the completed report to the back of your original lab and turn it in to your instructor at the beginning of the next class.