OpenSSL Commonly Used Commands for Class

Cryptography

Encoding

- Base64 encoding is a popular technique for **encoding** data
- Commonly used to represent binary characters
- It uses characters and numbers to encode 6 bits (2^6 = 64) at a time
- Let's see a simple example:

```
$ echo "secret" | base64
c2VjcmV0Cg==
```

- **Question**: Did we have to specify a key to encode this message?

Very Important: Encoding is not encryption! It is just a way of representing data.

- We use encoding such as ASCII and UTF-8 to represent characters in binary
- **Optional**: Let's do a simple example by hand: **https://en.wikipedia.org/wiki/Base64 (https://en.wikipedia.org/wiki/Base64)**
- Let's decode this on the command line:

```
echo "c2VjcmV0Cg==" | base64 -d
```

- We can use base64 to store keys and to transmit keys as well

Encryption - Symmetric

Note: GCM is not supported on command line.

- Let's actually encrypt a file. (**Remember: Encoding != Encryption!!!**)
- To generate a key, run the following command: `openssl enc -e -aes-256-cbc -pbkdf2 -salt -P`
- Type in a password, and then copy the key and save it in a file, for example, given the below output:

```
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
salt=D5281DB0E6DD504D
key=D74B2819B8CD6ABE5A930F21B861B7010D66F4C83198643F07DFB8D07389D8E9
iv =ED0519BEB326FEA76C48022A99597C6D
```

- You would save it in a file running the below command:
  - `echo "D74B2819B8CD6ABE5A930F21B861B7010D66F4C83198643F07DFB8D07389D8E9" > key`
  - Ideally, we would save the salt and use that as well
- Now, create a secret message and store it in a file called secret:
  - `echo "Spartie" > secret`
- We can encrypt the file secret and save the output in a new file called secret.enc:
  - `openssl enc -e -aes-256-cbc -pbkdf2 -kfile key -in secret -out secret.enc`
- Print out the contents of secret.enc using `cat` , do you see anything readable?
  - Try running the `strings` on it
  - Try base64 decoding the file
- You can decrypt the file using the same key and a similar command:
  - `openssl enc -d -aes-256-cbc -pbkdf2 -kfile key -in secret.enc -out secret.dec`
- If you `cat secret.dec`, do you see the message?

Encryption Asymmetric

- Sometimes we want to allow anyone to send us a message or we don't want to transmit a private symmetric key
- This is what asymmetric encryption is good for as it uses two keys: public and private
  - public - You provide this key to people, so they know what to use to send you data
    - This only encrypts the message when someone sends it to you
  - private - You keep this private, this is used to decrypt the messages sent to you
- Generate the private key:
  - `openssl genrsa -out private.pem 2048`
  - We can view the file by running the following: `cat private.pem`
- If we want to get the public key, we can do that:
  - `openssl rsa -in private.pem -pubout > public.pem`
  - Then we can print it out: `cat public.pem`
- Now, let's use the public to encrypt a message, but first, make a message:
  - `echo 'hello' > secret.msg`
  - Then encrypt it:
    - `openssl pkeyutl -encrypt -inkey public.pem -pubin -in secret.msg -out secret.enc`
    - Notice: You do not have to enter a password
    - If you cat out the file: `cat secret.enc` you will see a lot of garbled text
- Now lets decrypt it using our private key

```
$ openssl pkeyutl -decrypt -inkey private.pem -in secret.enc
Enter pass phrase for private.pem:
Hello!
```

- If we wanted, we can give anyone our public key and then use our private key