GPG Lab: Due Wednesday, 11:59pm

Your Name Here

October 26, 2016

1. Use SSH to log in to `vmwardrobe`

If you use Windows, you will have to download and install PuTTy. You need `putty.exe` and `puttygen.exe`, so the MSI installer is probably the right thing to use.

If you use OSX or Linux, you probably already have `ssh`. You can use it in the terminal application.

Use `ssh` to log in to `vmwardrobe.westmont.edu`. Your username is your Westmont username (without the `@westmont.edu` part) and your password is your 7-digit ID number. The first thing you should do is use the `passwd` command to **change your password.**

Once you have changed your password, set up your SSH keys on `vmwardrobe` so you don't need a password to log in. To do this, you will need to generate a key pair on your local machine. This will be an RSA key pair.

- Generate a key pair with OSX or Linux
- Generate a key pair with Putty

To use these keys for authentication, the public key needs to go in the right place on `vmwardrobe`, and the private key needs to go in the right place on your local machine. Paste your public SSH key into the code block below.

```
AAAAB3NzaC1yc2EAAAADAQABAAABAQDzWCC0nTIS38K5u9XrMLXQP8RKrKstQabsmxV7o22gNzvR/kOtYYhpgEhDsQVGaQ5FKiZwG2p

The ‘vmwardrobe‘ server will use this public key to encrypt a test message $m$ and send $E(m)$ to you. 

You should now be able to log in to ‘vmwardrobe‘ without a password. Try it. Once you are logged in, tr
```

youruserid@vmwardrobe:~$ mail youruserid@westmont.edu

```
After you enter a subject you can type a message. When you are done, type CTRL-D to send.

## 2. Complete the Getting Started section

Follow the instructions in the [Gnu Privacy Handbook, Chapter 1](https://www.gnupg.org/gph/en/manual/c1

1. Generate a public/private key pair (both for encryption and for signing).
2. Make a revocation certificate.
3. Export your public key as ‘XYZ_public_key.gpg‘ (where ‘XYZ‘ are your initials) and email it to me.
```

mail -A XYZ_public_key.gpg dhunter@westmont.edu

```
4. Download my public key: ‘wget http://math.westmont.edu/macs150/djh_public_key.gpg‘
5. Import my public key and check its fingerprint. Paste the fingerprint below.
```

Key fingerprint = 63B9 22C3 A296 E076 F2C1 2040 6000 1173 773E 8FE3

```
6. Send me an encrypted message. Make sure you use the appropriate key.
7. Send me a signed message. Except, before you do, change the behavior of GPG so that it uses the SHA-
8. Below are four signed messages from me, some of which are forged. Determine which of these messages
```

——BEGIN PGP SIGNED MESSAGE—— Hash: SHA1

Roses are red, ——BEGIN PGP SIGNATURE—— Version: GnuPG v1

iQEcBAEBAgAGBQJYC++TAAoJEGAAEXN3Po/je5gH/R/HMOW+43/Oqx4TDH2vkqTN rQvLE2m1l5HOC8IWBdYX6b
OUTlgeN6RlJd6N7uyWH36Ne3/UuLsqpGA4CgneTMVZdYsaCav6xbmm3VwBu4onmj WqW23jd0XfkIsLUQ3cgNklN+Dhjb+
ObyBkIniahDVitB+CUAWMB4YOtghPnxE8czWmw8yViLtD31kzCjHebt6AlepTqon 5KDAsvqjisx9aqsmYmQ9HKUIG3wgw+
=iM4W ——END PGP SIGNATURE——

——BEGIN PGP SIGNED MESSAGE—— Hash: SHA1

Violets are blue, ——BEGIN PGP SIGNATURE—— Version: GnuPG v1

iQEcBAEBAgAGBQJZC++7AAoJEGAAEXN3Po/jpT8IANIH2ETocS2QqL7oTNyo32UK GY2slIX0UxLgktnKH9qpKKktvk
/SJ9S6GCg3x+inl37d/1arNh6tylNtyF+6la7O2Kwxn+P0/bDf/DXk3rIXweRwUN 8cBquNj8u71k6/3o6Z0mAh4lLNMiE7DDap
sVU1DqYG1b+CX/cgkp+sa8Yd5e9QCwSH50+bj8zqZOpd9j5JoTmE9RVajsTeppCV e/31fsjltSMKE9CO9hbnUf1jCE/tuIoM9
=civA ——END PGP SIGNATURE——

——BEGIN PGP SIGNED MESSAGE—— Hash: SHA1

Encryption is fun, ——BEGIN PGP SIGNATURE—— Version: GnuPG v1

iQEcBAEBAgAGBQJYC/A7AAoJEGAAEXN3Po/jOdgH+we7NSzf2jenFQeyXepun+Na B2RICisMicGIm+nC19X1vwxOYep
qIVu69Z3k0Rlad/SmGc7XRzbyajxFVu75U4tKMwqD6xlWKFJ3h6CJkDyzf+NNjqV I0gPhbSEM4l896jlMJ1ok9kHuD0zzNfxW
kwHw/oqyHYjC95n+XYITMu+2TCbw9o8/OmuNU0h2SW/28Z0Wpy5rOwASBL0ayg/G LeqJDsFb15WAPim8bVkuPatr4QF
=Z2km ——END PGP SIGNATURE——

——BEGIN PGP SIGNED MESSAGE—— Hash: SHA1

But what is Gnu? ——BEGIN PGP SIGNATURE—— Version: GnuPG v1

iQEcBAEBAgAGBQJYC/BQAAoJEGAAEXN3Po/jiSwH/2ep2Sw3dz6cq6JSHmXevvFV gz9BJYToQaHvJ6tCdBrn7dOO1tN
qHa1YGHnhQH4Zkjd/omy41df51eLO2bjuj06c3tUbWwUZCpDohNr7wT6Dmd+uU3y 5ceRlLJTe8BM5epYLd0YbHn6G76Ozr
ObCJS6WprbwOmgcN981sUEvlWxcMyQCQXl7IUDns4BcdneBDfe7RDkgQ1r4m635H tTBUkzLd3i0Y/J12I29PnY8DTj86CJm
=Um/3 ——END PGP SIGNATURE—— “‘ 1. Good 2. Bad CRC 3. Good 4. Bad Signature

3. “Playstation” Attack on ElGamal

Suppose that Alice signs two documents using the ElGamal signature scheme with

$$p = 1267650600228229401496703205653$$

$\alpha = 2$, and $\beta = 479366713173960022956350873704$. The two signed messages are

$$(m_1, r_1, s_1) = (73646, 544051462776724778073434116661, 914404324671027799264463858401)$$

and

$$(m_2, r_2, s_2) = (63513, 544051462776724778073434116661, 1236987333514898966758089443580).$$

1. Why is it obvious that Alice used the same value of k for both signatures?

   The r values are the same. r is found by rasing α^k.

2. Find this value of k, and also the secret value of a such that $\beta = \alpha^a$ in $U(p)$.

$$a * r = m_1 - k * s_1 = m_2 - k * s_2$$

```r
p <- as.bigz("1267650600228229401496703205653")
m1 <- as.bigz("73646")
m2 <- as.bigz("63513")
s1 <- as.bigz("914404324671027799264463858401")
s2 <- as.bigz("1236987333514898966758089443580")
r <- as.bigz("544051462776724778073434116661")
s3 <- s2 - s1
m3 <- m2-m1

k <-mod.bigz(inv.bigz(s3, (p-1))*m3, (p-1))

print(k)
```

```
Big Integer ('bigz') :
[1] 123000000000001
```

```r
a <- mod.bigz(inv.bigz(r, (p-1))*(m1-k*s1), (p-1))

print(a)
```

```
Big Integer ('bigz') :
[1] 19451945194519451945
```