Signing and Encryption with GnuPG

Steve Revilak
http://www.srevilak.net/wiki/Talks

Cryptoparty @ Somerville Public Library

January 10, 2015

What is GnuPG?

- GnuPG is a free software implementation of the OpenPGP standard.
  - PGP stands for *Pretty Good Privacy*
- PGP is a system for *encrypting* data, and for creating digital signatures (aka *signing*).
- Commonly used for Email, but can be used with any type of data or file.
- PGP takes a little work to set up. After that, it's easy to use.
- Today, we'll focus on the setup part.

Where do I get GnuPG?

Mac OS https://gpgtools.org/

Windows http://gpg4win.org/

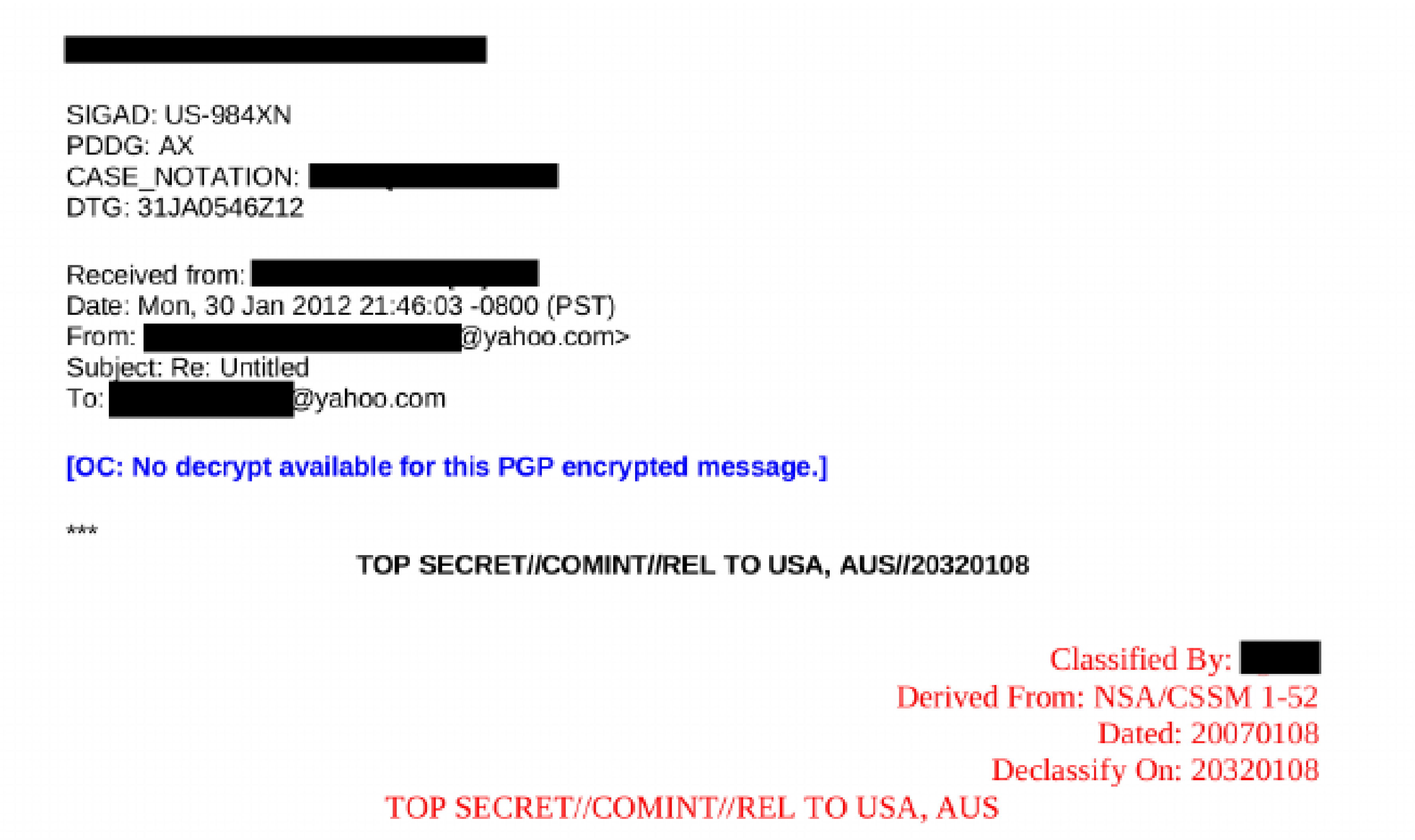
Linux GnuPG may already be installed. If not, use your package manager (yum, apt-get, zypper, synaptic, aptitude, etc.) to install it.

Also useful:

Thunderbird https://www.mozilla.org/en-US/thunderbird/

Enigmail https://www.enigmail.net/home/index.php

Why Use GnuPG?



[redacted]

SIGAD: US-984XN
PDDG: AX
CASE_NOTATION: [redacted]
DTG: 31JA0546Z12

Received from: [redacted]
Date: Mon, 30 Jan 2012 21:46:03 -0800 (PST)
From: [redacted]@yahoo.com>
Subject: Re: Untitled
To: [redacted]@yahoo.com

[OC: No decrypt available for this PGP encrypted message.]

***

TOP SECRET//COMINT//REL TO USA, AUS//20320108

Classified By: [redacted]
Derived From: NSA/CSSM 1-52
Dated: 20070108
Declassify On: 20320108

TOP SECRET//COMINT//REL TO USA, AUS

A brief introduction to keys

Objective: Alice wants to (securely) send a file to Bob.

- Alice encrypts the file with a password
- Alice sends the encrypted file to Bob
- Bob gets the encrypted file, but . . .
- How does Alice (securely) get the *password* to Bob?
- This is the dilemma with password-based encryption.

Public key cryptography avoids this problem entirely. Instead of passwords, you can use public and private keys.

Public and Private Keys

In order to do anything with PGP, you'll need a *key*. Keys exist as a pair, called a *keypair*.

- There's a *public key*. You share this with everyone (because it's public).
- There's a *private key*, sometimes called a *secret key*. Don't share this with anyone (because it's a secret).

The private key will "undo" what the public key does, and vice versa; think of them as inverse functions. If a public key encrypts a message, then the corresponding private key decrypts it.

Now,

- Alice can encrypt the file with Bob's public key.
- Bob decrypts the file with his private key.

What can you do with a key?

Keys allow you to sign and encrypt messages.

- Signing Guarantees that a message was sent by someone with a specific private key (and wasn't subsequently altered).
- Encryption The purpose is to ensure that a message is readable only by someone possessing a specific private key.

(Here I use the term "message" in a very generic sense – it could be an email message, a file, or any arbitrary piece of data).

Leap of faith: You need some level of trust that a particular key belongs to a *particular person*.

Equations!?! Oh NOES!

decrypt(PRIVKEY, encrypt(PUBKEY, MSG)) = MSG

- This is how encryption/decryption works

decrypt(PUBKEY, encrypt(PRIVKEY, MSG)) = MSG

- This is how signing/verification works

Many crypto applications use this technique. For example, PGP, HTTPS, OTR.

Goals for this part of the workshop

- Generate a keypair (if you don't already have one).
  - Upload your public key to a keyserver
  - Download my public key.
- Set up your mail program to send and receive signed and encrypted email.
  (Mail program = Mail User Agent, or MUA)
- Send me a signed and encrypted message. (I should be able to decrypt your message, and verify your signature.)
- I'll respond with a signed and encrypted message. (You should be able to decrypt my message and verify my signature.)

Mail Client Basics

Sending:

- You'll use a protocol called SMTP, or Simple Mail Transfer Protocol.

Receiving:

- Two options: IMAP (Internet Mail Access Protocol), or POP (Post Office Protocol)
- IMAP stores all messages on your ESP's mail server. You can move them to local folders, but you have to do this explicitly.
- POP downloads mail from your ESP's mail server. By default, the server copy is deleted; you can also configure your mail client to leave it on the server.
- If you have a lot of mail on the server, the initial synchronization might take a while, especial with POP.

Generating a Keypair

I'm going to demonstrate with Thunderbird and Enigmail. The command-line equivalents are here for reference.

- Generate a key (if you don't already have one).
  ```
  gpg --gen-key
  ```
  Choose RSA, RSA. Use the longest key possible (4096 bits).
- Upload your key to a keyserver.
  ```
  gpg --send-key KEYID
  ```
- Download my public key.
  `gpg --search steve@srevilak.net` OR
  `gpg --recv-key 28C2A300`

Sending and receiving mail

- We'll take this one step at a time.
- Send me a signed and encrypted message.
- Open your Sent Mail folder. Make sure you can read the encrypted message that you just sent!
- I'll respond. Work on downloading, decrypting, and reading my message. Be sure to verify the signature.

Backing up your keys

If you lose your private key, then forget about decryption. *There is no password recovery for keys!* This is by design.

- Backup your private key

  ```
  gpg -a --export-secret-keys KEYID > private-key.asc
  ```

Store a copy of `private-key.asc` in a safe place. For example, keep electronic and printed copies in a safe deposit box.

Revocation Certificates

What if (say) your laptop is stolen, and you lose your private key? If this happens, you'll want to *revoke* your key.

- Generate a revocation certificate

  ```
  gpg -a --gen-revoke KEYID > pgp-revoke.asc
  ```

Uploading the revocation certificate (to a keyserver) "cancels" your key.

Note: you cannot generate a revocation certificate without a private key! Keep the revocation certificate in a safe place.

Trusting and Signing Keys (1)

How do you know that a given key belongs to a given person? You check the key's *fingerprint*. Here's my fingerprint:

```
gpg --fingerprint 28C2A300
...
Key fingerprint = 6F09 15FF 59CE E093 56F4
                  BEEC E772 7C56 28C2 A300
```

The fingerprint uniquely identifies a PGP key. If the fingerprints match, you've got the right one.

Note: the key id is the last eight digits of the fingerprint.

Trusting and Signing Keys (2)

Signing a key indicates that you trust it.

- `gpg --sign-key 28C2A300` OR
  `gpg --lsign-key 28C2A300`

`--lsign-key` makes a local signature; it's only visible to you.

To distribute a non-local (`--sign-key`) signature:

- Send it to a key server:
  `gpg --send-key 28C2A300`
- Export the key (containing your signature), and send it to the key holder.
  `gpg -a --export 28C2A300 > signed-key.asc`

The key holder will `gpg --import signed-key.asc` to import your signature.

Some Advanced Tips

$HOME/.gnupg/gpg.conf is GnuPG's configuration file. Some things you should consider adding:

```
# Sign keys using SHA256, instead of SHA1
cert-digest-algo SHA256


# Sign messages using SHA256, too
personal-digest-preferences SHA256


# Set stronger preferences on newly-generated keys
# Put this all on one line.
default-preference-list SHA512 SHA384 SHA256 SHA224 \
             AES256 AES192 AES CAST5 ZLIB BZIP2 \
             ZIP Uncompressed
```

More Advanced Tips

Change the preferences of your existing key, to match the `default-preference-list` in the previous slide.

See instructions at `http://www.apache.org/dev/openpgp.html`.

Tip: It doesn't hurt to back up your key before trying this.

GnuPG Wrap Up

- PGP protects your privacy through encryption.
- PGP provides non-repudiation through digital signatures.
- PGP is something that you can (and should!) use every day.
- GnuPG is a free software implementation of a public standard. It's harder to backdoor software when the source code is public.

PGP Resources

- GnuPG: http://gnupg.org/
- GPG4win: http://www.gpg4win.org/
- GPG Tools: http://gpgtools.org/
- Riseup.net's Best practices for OpenPGP: https://we.riseup.net/riseuplabs+paow/openpgp-best-practices
- Cryptoparty handbook: https://www.cryptoparty.in/documentation/handbook
- Surveillance Self-Defense: https://ssd.eff.org/
- Email Self-Defense: https://emailselfdefense.fsf.org/en/