Secure Web Surfing and Email

Steve Revilak

2013 Digitial Media Conference

Oct. 26, 2013

Outline

- Secure web surfing with Tor
  - Around 30 minutes
- Email Encryption with GnuPG
  - 60 minutes (or whatever time we have left)

Part I: Tor

What is Tor?

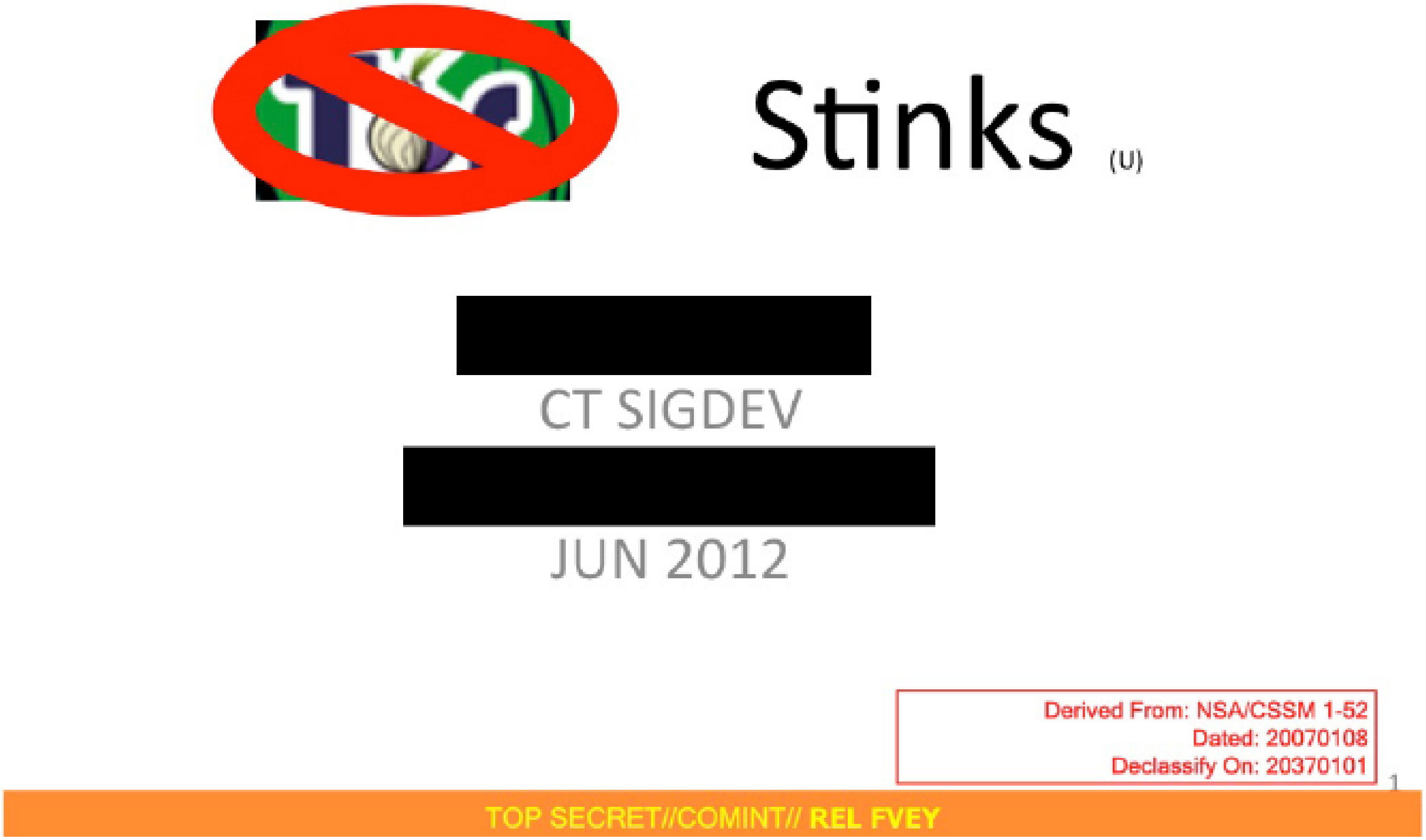
Tor is . . .

> *“free software and an open network that helps you defend against traffic analysis, a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security.”*

(quote from `https://www.torproject.org/`)

In short, Tor conceals the source of web traffic, along with the content of that web traffic.

Why use Tor? (1)



Stinks (U)

[redacted]

CT SIGDEV

[redacted]

JUN 2012

Derived From: NSA/CSSM 1-52
Dated: 20070108
Declassify On: 20370101

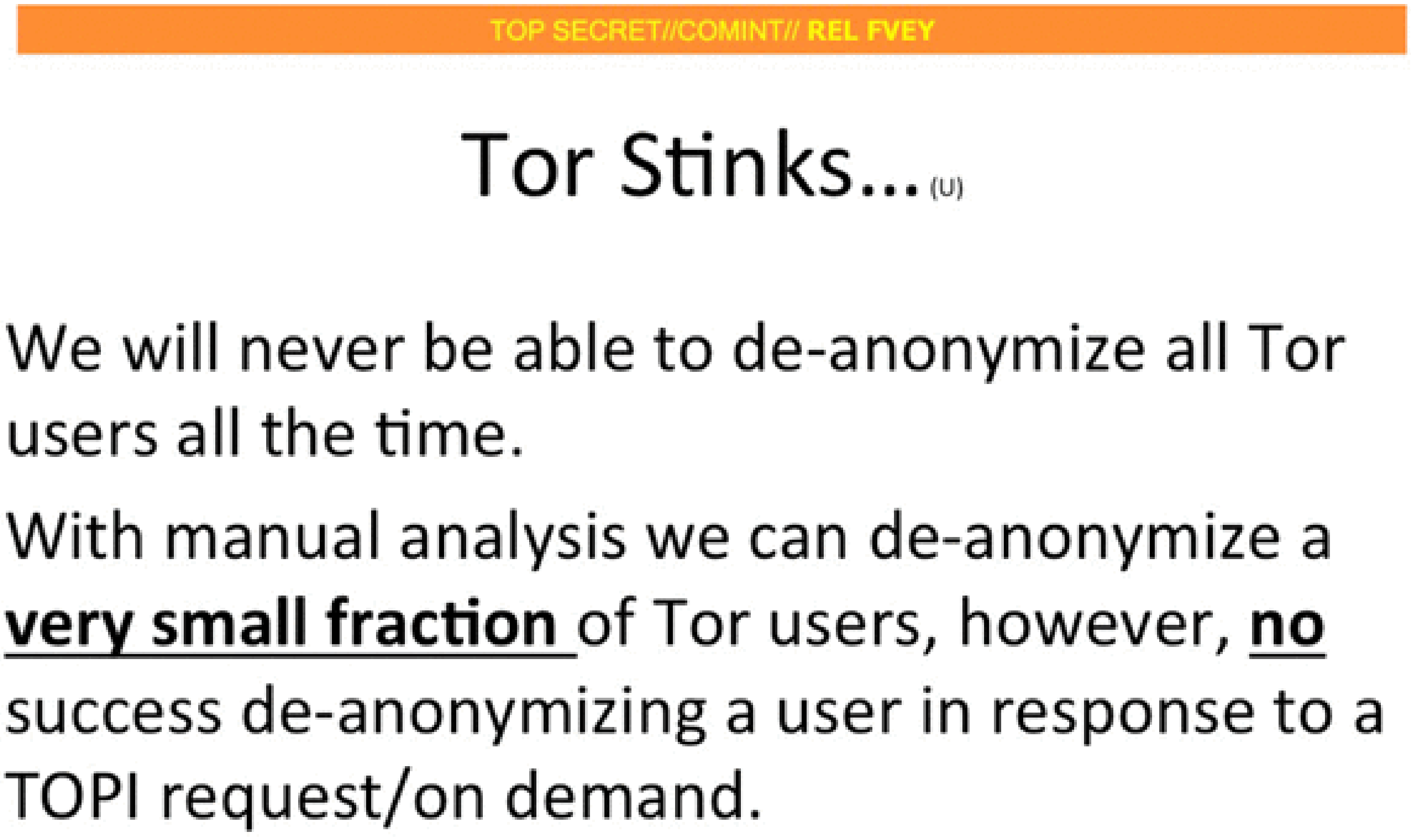
1

TOP SECRET//COMINT// REL FVEY

Source: NSA *Tor Stinks* presentation.

`https://www.documentcloud.org/documents/801434-doc2.html`

Why Use Tor? (2)



TOP SECRET//COMINT// **REL FVEY**

Tor Stinks... (U)

- We will never be able to de-anonymize all Tor users all the time.
- With manual analysis we can de-anonymize a **very small fraction** of Tor users, however, **no** success de-anonymizing a user in response to a TOPI request/on demand.

Source: *ibid*

Why Use Tor? (3)

I see two ways to interpret the NSA's *Tor Stinks* presentation:

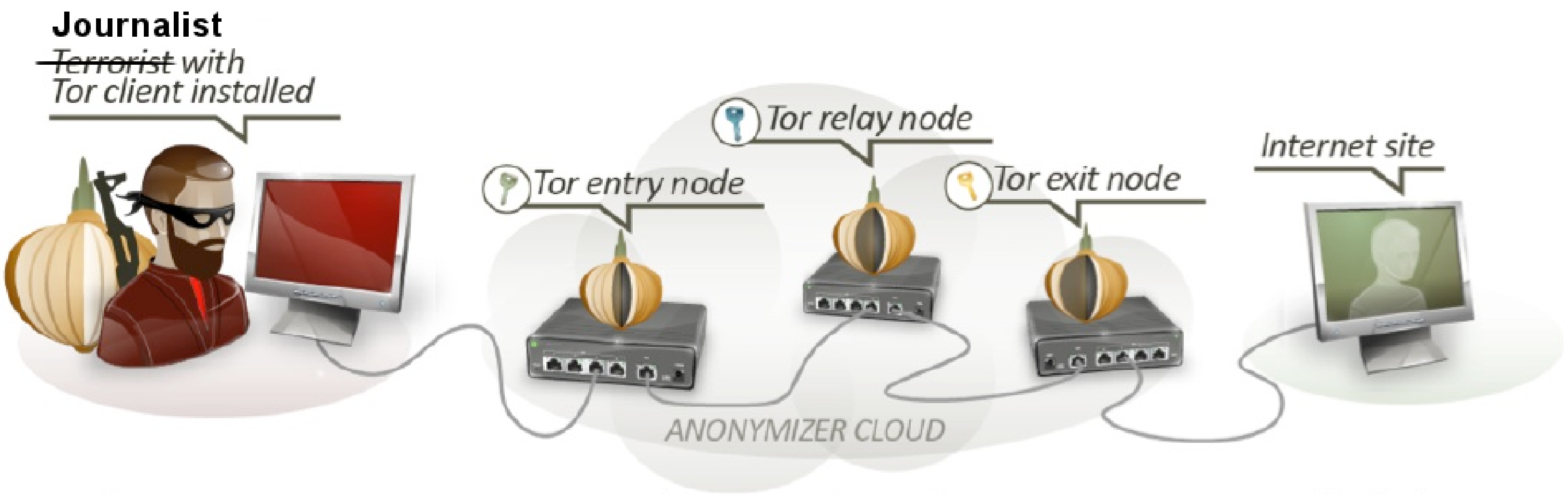
1. Tor is a good system for preserving anonymity on the web, and the NSA has trouble "breaking" it.
2. Tor is horribly broken, so the NSA wants us all to use it.

Personally speaking, I'm more inclined to believe the former.

Installing and Running Tor

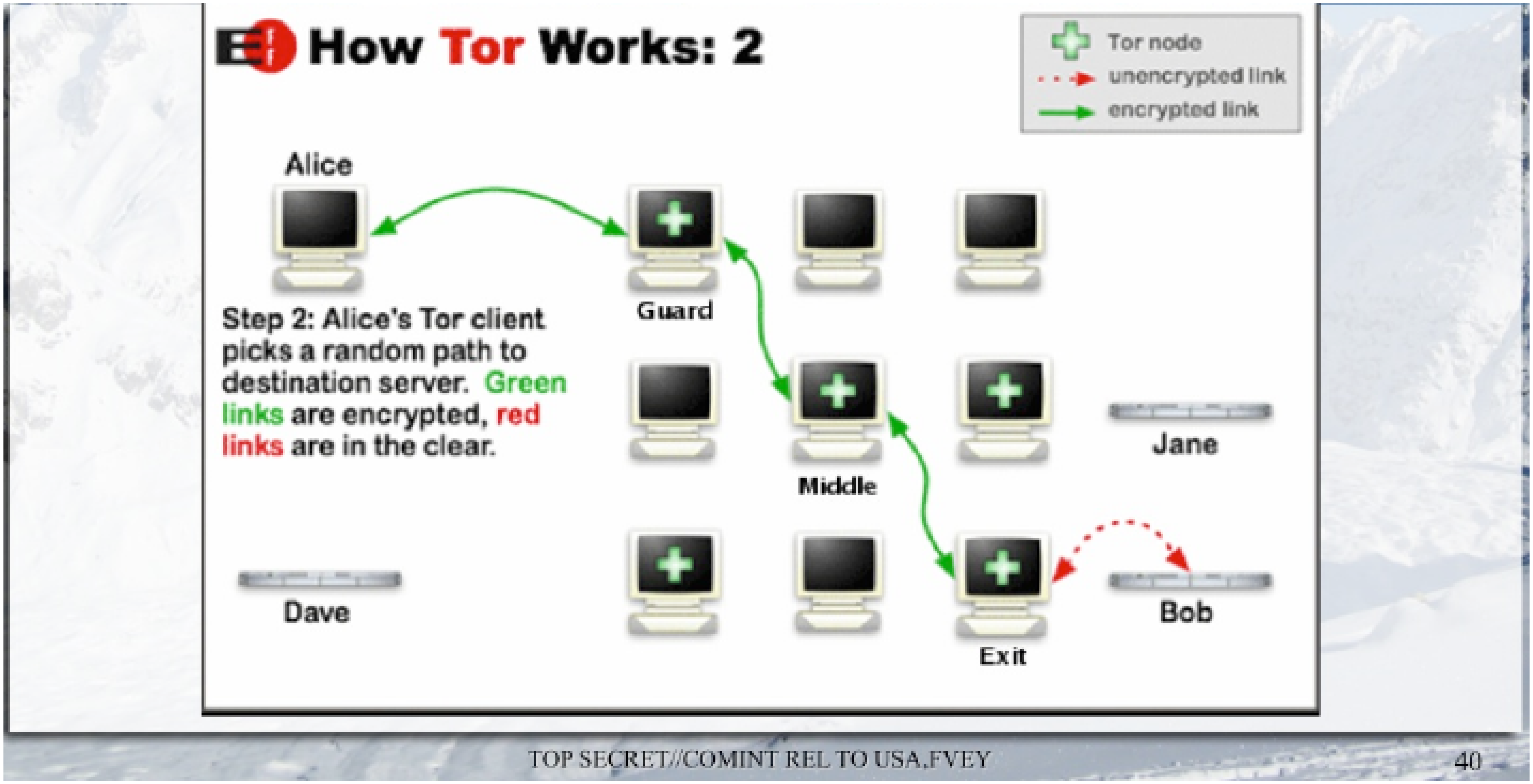
- You can download Tor from the Tor Project's web site `https://www.torproject.org`
- Tor Browser should look familiar to Firefox users – it's a customized version of Firefox.
- Browse to `https://www.maxmind.com/en/geoip_demo` with Tor. What do you see? Try again with a regular web browser, and compare the results.

How Tor Works (1)



Source: My EFF-inspired tweaks to a *Tor Stinks* graphic.

How Tor Works (2)



How Tor Works: 2

Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.

TOP SECRET//COMINT REL TO USA,FVEY

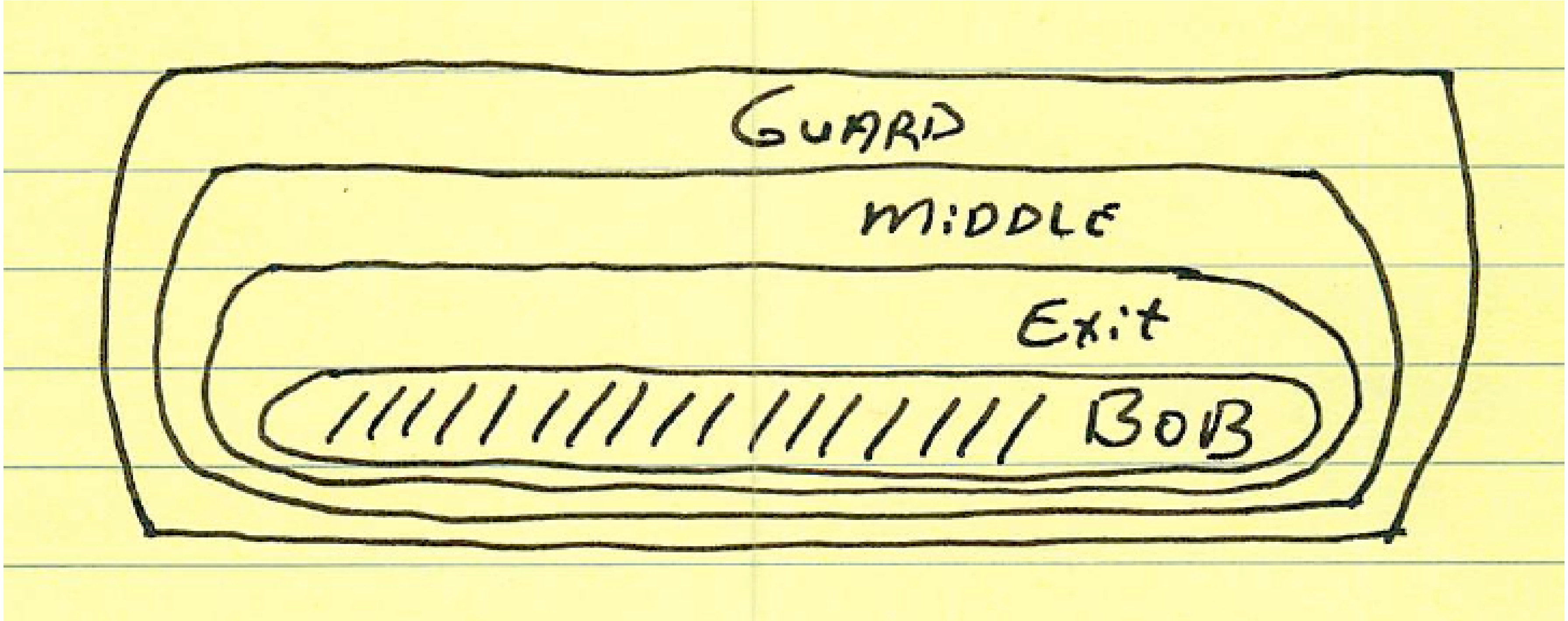
40

Source: NSA *Advanced Open Source Multi-Hop* presentation

`https://www.documentcloud.org/documents/801435-doc3.html`

How Tor Works (3)

Here's what Alice's Tor request looks like:



These layers of encryption constitute an *onion*.

Tor Caveats

- Tor is slower than using an ordinary web browser.
- Web sites that rely heavily on Geolocation (e.g., Google) don't play well with Tor.
- You may need to change your browsing habits (see `https://www.torproject.org/download/download-easy.html#warning` for elaboration).

Tor Wrap Up

- Using Tor is just as easy as using any other web browser.
- Tor protects you from surveillance by proxying web traffic through a network of Tor nodes.
- The network of Tor nodes conceals your location; encryption protects the content of your traffic.

Part II: GnuPG

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 file.
- PGP can take a little work to set up. After that, it's easy to use.

Encryption and Signing

Encryption The purpose is to ensure that a message is readable only by someone possessing a specific key.

Signing Guarantees that a message was sent by someone with a specific key (and wasn't subsequently altered).

(Here I'm using 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.

Public vs Private Keys

Keys exist as a pair (a *keypair*):

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

During *encryption*, the sender encrypts the message with the recipient's public key. The recipient uses their private key to decrypt the message.

During *signing*, the sender signs the message with their private key. The signature can be verified by anyone with the corresponding public key.

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.)

Generating a Keypair

We can do these things with GUI tools. For reference, these are command-line equivalents.

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

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.

Configuring your MUA (GMail)

GMail:

- Enable IMAP or POP in Gmail's web interface.
- Sending: smtp.gmail.com, port 587, use SSL
- Receiving: imap.gmail.com, port 993, use SSL; OR pop.gmail.com, port 995, use SSL
- For help, see `https://support.google.com/mail/troubleshooter/1668960?hl=en&ref_topic=1669040`

Configuring your MUA (Hotmail)

Hotmail:

- Enable POP/IMAP in outlook.com's web interface
- Sending: smtp-mail.outlook.com, port 587, use TLS
- Receiving: imap-mail.outlook.com, Port 993, use SSL; OR pop-mail.outlook.com, port 995, SSL
- For help, see `http://windows.microsoft.com/en-us/windows/outlook/send-receive-from-app`

Configuring your MUA (Yahoo)

Yahoo:

- POP is only available for Yahoo Plus Accounts
- Sending: smtp.mail.yahoo.com, port 587, use SSL
- Receving: pop.mail.yahoo.com, port 995, use SSL; OR imap.mail.yahoo.com, port 993, use SSL
- For help, see `http://help.yahoo.com/kb/index?page=content&y=PROD_MAIL_ML&locale=en_US&id=SLN4075`

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.

Trusting and Signing Keys (1)

How do you verify 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
```

If the fingerprint matches, you've got the right key.

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.

Backing up your keys

If you lose your private key, then forget about decryption. Lost private keys cannot be recovered!

- 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.

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. Remember: it's hard 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/`