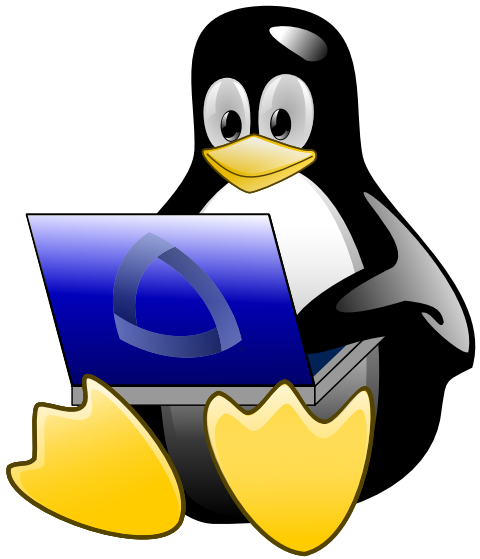
PGP, GPG, and Enigmail... Oh My!

Jack Rosenthal

25 January 2016



Colorado School of Mines
Linux Users Group

Pretty Good Privacy

- PGP is a data encryption, decryption, and signing program. It follows the OpenPGP standard.
- People use PGP to sign, encrypt, and decrypt emails, files, folders, and even whole disk partitions.
- PGP allows you to specify a recipient to encrypt a message for given only their public key, you can even encrypt to multiple recipients given only their public keys.

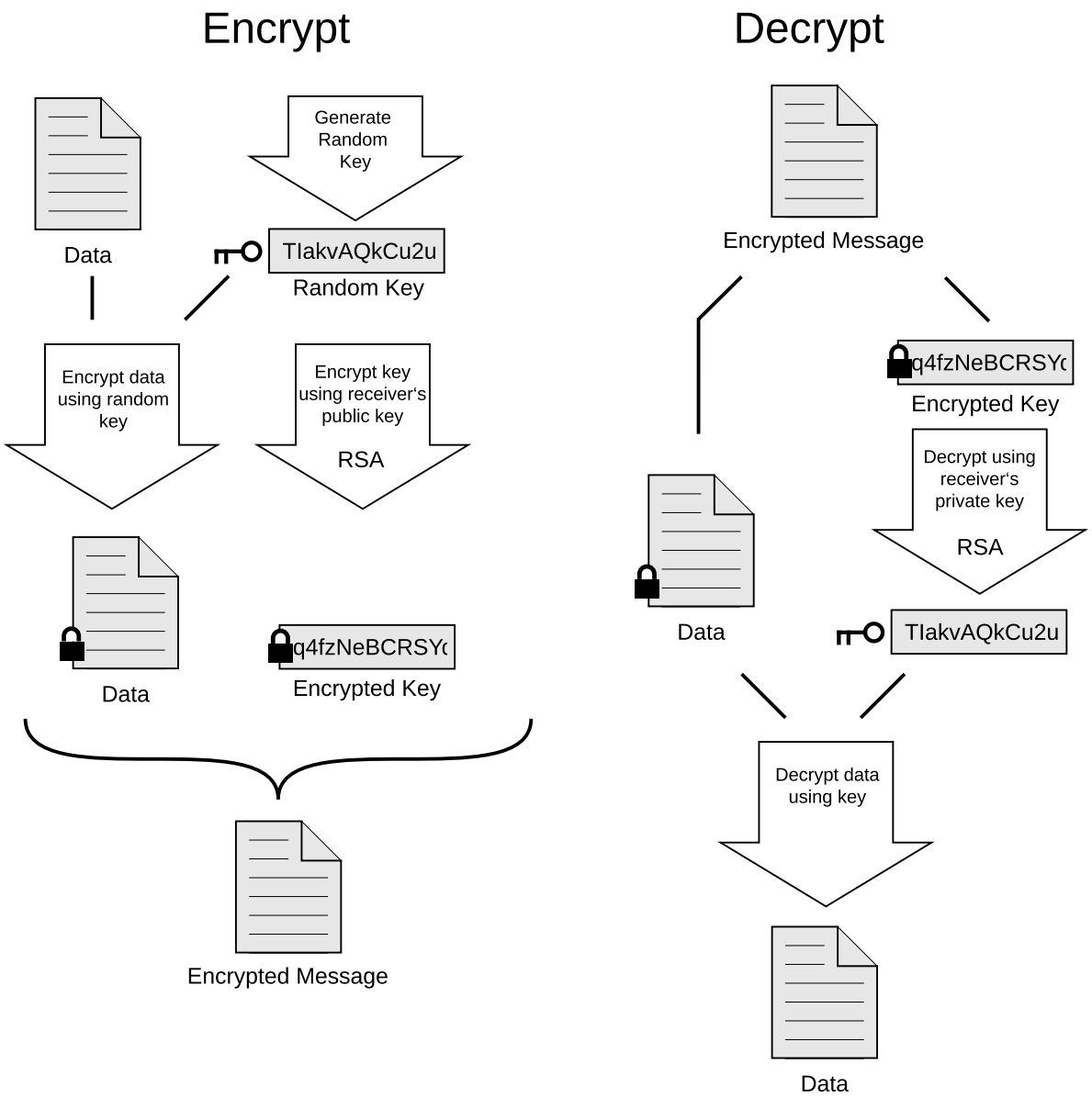
Pretty Good Privacy

- PGP is a data encryption, decryption, and signing program. It follows the OpenPGP standard.
- People use PGP to sign, encrypt, and decrypt emails, files, folders, and even whole disk partitions.
- PGP allows you to specify a recipient to encrypt a message for given only their public key, you can even encrypt to multiple recipients given only their public keys.

Pretty Good Privacy

- PGP is a data encryption, decryption, and signing program. It follows the OpenPGP standard.
- People use PGP to sign, encrypt, and decrypt emails, files, folders, and even whole disk partitions.
- PGP allows you to specify a recipient to encrypt a message for given only their public key, you can even encrypt to multiple recipients given only their public keys.

How it Works


Encrypt
Data
Generate Random Key
TIakvAQkCu2u
Random Key
Encrypt data using random key
Encrypt key using receiver's public key
RSA
Data
q4fzNeBCRSYc
Encrypted Key
Encrypted Message
Decrypt
Encrypted Message
q4fzNeBCRSYc
Encrypted Key
Decrypt using receiver's private key
RSA
Data
TIakvAQkCu2u
Decrypt data using key
Data

GNU Privacy Guard

- Free implementation of OpenPGP standard
- Typically interfaced through gpg command line tool, but various GUI's are available
- Probably already on your system as most package managers (`pacman`, `apt`, etc) require it for package verification
- See `man gpg` for all of its options

GNU Privacy Guard

- Free implementation of OpenPGP standard
- Typically interfaced through gpg command line tool, but various GUI's are available
- Probably already on your system as most package managers (`pacman`, `apt`, etc) require it for package verification
- See `man gpg` for all of its options

GNU Privacy Guard

- Free implementation of OpenPGP standard
- Typically interfaced through gpg command line tool, but various GUI's are available
- Probably already on your system as most package managers (`pacman`, `apt`, etc) require it for package verification
- See `man gpg` for all of its options

GNU Privacy Guard

- Free implementation of OpenPGP standard
- Typically interfaced through gpg command line tool, but various GUI's are available
- Probably already on your system as most package managers (`pacman`, `apt`, etc) require it for package verification
- See `man gpg` for all of its options

Web of Trust

How can I trust PGP content someone sends me?

Phil Zimmermann, PGP creator

As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.

- Meet with them in person and verify the key fingerprints match, then sign their key.
- Or, verify someone you trust has signed their key.

Web of Trust

How can I trust PGP content someone sends me?

Phil Zimmermann, PGP creator

As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.

- Meet with them in person and verify the key fingerprints match, then sign their key.
- Or, verify someone you trust has signed their key.

Web of Trust

How can I trust PGP content someone sends me?

Phil Zimmermann, PGP creator

As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.

- Meet with them in person and verify the key fingerprints match, then sign their key.
- Or, verify someone you trust has signed their key.

Web of Trust

How can I trust PGP content someone sends me?

Phil Zimmermann, PGP creator

As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers. And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures. This will cause the emergence of a decentralized fault-tolerant web of confidence for all public keys.

- Meet with them in person and verify the key fingerprints match, then sign their key.
- Or, verify someone you trust has signed their key.

Making your own PGP key

1. `$ gpg --gen-key`
2. gpg will ask you a number of questions... answer them
3. After making a secure passphrase, you will have to wait a few minutes for it to generate a key. During this time, you may want to browse the web, or do something else to increase system entropy, as this will speed up the prime number generation.
4. `$ gpg --list-keys` and make sure your new key is there
5. Upload your key to a keyserver like `pgp.mit.edu`
   `$ gpg --keyserver pgp.mit.edu --send-keys` *`B20E73F7`*

Making your own PGP key

1. `$ gpg --gen-key`
2. gpg will ask you a number of questions... answer them
3. After making a secure passphrase, you will have to wait a few minutes for it to generate a key. During this time, you may want to browse the web, or do something else to increase system entropy, as this will speed up the prime number generation.
4. `$ gpg --list-keys` and make sure your new key is there
5. Upload your key to a keyserver like `pgp.mit.edu`
   `$ gpg --keyserver pgp.mit.edu --send-keys B20E73F7`

Making your own PGP key

1. `$ gpg --gen-key`
2. gpg will ask you a number of questions... answer them
3. After making a secure passphrase, you will have to wait a few minutes for it to generate a key. During this time, you may want to browse the web, or do something else to increase system entropy, as this will speed up the prime number generation.
4. `$ gpg --list-keys` and make sure your new key is there
5. Upload your key to a keyserver like `pgp.mit.edu`
   `$ gpg --keyserver pgp.mit.edu --send-keys B20E73F7`

Making your own PGP key

1. `$ gpg --gen-key`
2. gpg will ask you a number of questions... answer them
3. After making a secure passphrase, you will have to wait a few minutes for it to generate a key. During this time, you may want to browse the web, or do something else to increase system entropy, as this will speed up the prime number generation.
4. `$ gpg --list-keys` and make sure your new key is there
5. Upload your key to a keyserver like `pgp.mit.edu`

   `$ gpg --keyserver pgp.mit.edu --send-keys B20E73F7`

Making your own PGP key

1. `$ gpg --gen-key`
2. gpg will ask you a number of questions... answer them
3. After making a secure passphrase, you will have to wait a few minutes for it to generate a key. During this time, you may want to browse the web, or do something else to increase system entropy, as this will speed up the prime number generation.
4. `$ gpg --list-keys` and make sure your new key is there
5. Upload your key to a keyserver like `pgp.mit.edu`
   `$ gpg --keyserver pgp.mit.edu --send-keys` *B20E73F7*

Generating a Revocation Certificate

1. `gpg --output revoke.asc --gen-revoke` *`B20E73F7`*
2. "If you forget your passphrase or if your private key is compromised or lost, this revocation certificate may be published to notify others that the public key should no longer be used."
3. revoked public keys
   - can still verify signatures made by you in the past
   - cannot be used to encrypt future messages to you
   - do not affect ability to decrypt previously received messages (because that depends on your private key)

Generating a Revocation Certificate

1. `gpg --output revoke.asc --gen-revoke` *`B20E73F7`*
2. "If you forget your passphrase or if your private key is compromised or lost, this revocation certificate may be published to notify others that the public key should no longer be used."
3. revoked public keys
   1. can still verify signatures made by you in the past
   2. cannot be used to encrypt future messages to you
   3. do not affect ability to decrypt previously received messages (because that depends on your private key)

Generating a Revocation Certificate

1. `gpg --output revoke.asc --gen-revoke` *`B20E73F7`*
2. "If you forget your passphrase or if your private key is compromised or lost, this revocation certificate may be published to notify others that the public key should no longer be used."
3. revoked public keys
    1. can still verify signatures made by you in the past
    2. cannot be used to encrypt future messages to you
    3. do not affect ability to decrypt previously received messages (because that depends on your private key)

Generating a Revocation Certificate

1. `gpg --output revoke.asc --gen-revoke` *`B20E73F7`*
2. "If you forget your passphrase or if your private key is compromised or lost, this revocation certificate may be published to notify others that the public key should no longer be used."
3. revoked public keys
   1. can still verify signatures made by you in the past
   2. cannot be used to encrypt future messages to you
   3. do not affect ability to decrypt previously received messages (because that depends on your private key)

Generating a Revocation Certificate

1. `gpg --output revoke.asc --gen-revoke` *`B20E73F7`*
2. "If you forget your passphrase or if your private key is compromised or lost, this revocation certificate may be published to notify others that the public key should no longer be used."
3. revoked public keys
   1. can still verify signatures made by you in the past
   2. cannot be used to encrypt future messages to you
   3. do not affect ability to decrypt previously received messages (because that depends on your private key)

Generating a Revocation Certificate

1. `gpg --output revoke.asc --gen-revoke` *B20E73F7*
2. "If you forget your passphrase or if your private key is compromised or lost, this revocation certificate may be published to notify others that the public key should no longer be used."
3. revoked public keys
   1. can still verify signatures made by you in the past
   2. cannot be used to encrypt future messages to you
   3. do not affect ability to decrypt previously received messages (because that depends on your private key)

Signing other's keys

1. Get their key from a keyserver

   ```
   $ gpg --keyserver pgp.mit.edu --recv B20E73F7
   ```

2. `$ gpg --sign-key B20E73F7`
3. Have them run `$ gpg --fingerprint` in person and verify the fingerprint matches the one on your terminal
4. Send their key (now signed) back to the keyserver.

   ```
   $ gpg --keyserver pgp.mit.edu --send-keys B20E73F7
   ```

Signing other's keys

1. Get their key from a keyserver

   ```
   $ gpg --keyserver pgp.mit.edu --recv B20E73F7
   ```

2. `$ gpg --sign-key B20E73F7`
3. Have them run `$ gpg --fingerprint` in person and verify the fingerprint matches the one on your terminal
4. Send their key (now signed) back to the keyserver.

   ```
   $ gpg --keyserver pgp.mit.edu --send-keys B20E73F7
   ```

Signing other's keys

1. Get their key from a keyserver

   `$ gpg --keyserver pgp.mit.edu --recv` *B20E73F7*
2. `$ gpg --sign-key` *B20E73F7*
3. Have them run `$ gpg --fingerprint` in person and verify the fingerprint matches the one on your terminal
4. Send their key (now signed) back to the keyserver.

   `$ gpg --keyserver pgp.mit.edu --send-keys` *B20E73F7*

Signing other's keys

1. Get their key from a keyserver

   `$ gpg --keyserver pgp.mit.edu --recv` *B20E73F7*
2. `$ gpg --sign-key` *B20E73F7*
3. Have them run `$ gpg --fingerprint` in person and verify the fingerprint matches the one on your terminal
4. Send their key (now signed) back to the keyserver.

   `$ gpg --keyserver pgp.mit.edu --send-keys` *B20E73F7*

Integrating GnuPG with your email client

- Thunderbird: Install the Enigmail addon. The GUI is pretty self explanatory.
- Mutt: See `http://dev.mutt.org/trac/wiki/MuttGuide/UseGPG`
- Android: Use K9 mail and install APG
- Other email clients: Google it

When setting up your email client, please use PGP/MIME, as *inline PGP considered harmful*.