Cryptography Basics
Concepts and fundamentals

Davis Claiborne

LUG @ NC State

April 17, 2019

Linux Users Group
at NC State University

Types of cryptography

There are two main types of ways to encrypt messages:

- Symmetric encryption
- Asymmetric encryption

Symmetric encryption is based on a shared, secret key

In **asymmetric encryption**, each person has two keys, a public key and private key [1]

Digital signatures are also an important part of cryptography, ensuring that the message being sent has not been altered

[1] This is a bit of an oversimplification - symmetric and asymmetric refer to the information available to both parties

Symmetric cryptography

A shared, secret key is used to encrypt and decrypt messages [3]

Pros:

- Fast and not mathematically intensive
- Simple to implement

Cons:

- Need a separate way to communicate secret key
- Anybody that guesses or knows the key can decrypt your messages and impersonate another person
- Number of keys required for pair-wise communications grows at at rate of n^2

E.g. a password-encrypted file

Simple symmetric encryption example

Choose a password to be your 'key' to encrypt a message

'*dog*' will be the key and '*this is a test*' will be the message

In our simple encryption scheme, a message is encrypted as follows:

- Convert the password to using a trapdoor function: [2] [3]
  - Replace characters with place in alphabet
  - Square each number then sum them
  - E.g. 'dog' $\rightarrow 4^2 + 15^2 + 7^2 = 290$

[2] A trapdoor function is a function that is easy to convert a group of inputs to, but it is hard to determine the original numbers given one or more of the inputs

[3] The methods used in these examples are **not** trapdoor functions, nor are they particularly secure; they are only for demonstration purposes

Simple symmetric encryption example - continued

- Convert the message to a number using the key:
  - Convert letters in message to a number: [4]
  - "this is a test" $\rightarrow$ 20 8 9 19 0 9 19 0 1 0 20 5 19 20
- In this encryption method, multiply each letter by the key, then pad zeros to make the number 4 digits long:
  - $20 \cdot 290 = 5800$, etc.
  - Message becomes '5800232026105510000026105510000002900000580014505510580 0'

[4] Space is being encoded as 0

Asymmetric cryptography

Each person communicating has two keys: a public key and a private key [3]

Pros:

- Can guarantee privacy and sender authenticity [5]

Cons:

- Requires either a central authority or prior communication to guarantee public key
- Current implementations are mathematically intensive and slow compared to symmetric cryptography

[5] This is different from *validating* the sender's authenticity

Simple asymmetric encryption example

In our chosen scheme:

- A message encrypted using your public or private key should be evenly divisible by the other key
- You are the only person who can have that key

For our example, assume the following:

- Your public key and private key should sum to 100
- Your public key is '*linux*' $\rightarrow 82$ [6]
- This makes your private key $100 - 82 = 18$

[6] '*linux*' $\rightarrow 12 + 9 + 14 + 21 + 26 = 82$

Simple asymmetric encryption example - continued

To encrypt the message using your private key, multiply each letter by $100 - pr$ [7]

If our message can be represented as *message*,

- $(100 - pr) \cdot message = 82 \cdot message$
- Because our encrypted message is divisible by our public key, it can be proven that the message was sent by us

[7] *pr* represents the *private key*

Encryption using your private key

If you encrypt a message with your private key, it can be guaranteed to have come from you by using your public key to decrypt it [8] [4]

Problem: Everybody knows your public key, so anybody can decrypt your message

[8] The act of encrypting something using your private key is generally known as *signing*

Encryption using the recipient's public key

If you encrypt a message with the recipient's public key, it can be guaranteed that only they will be able to read it [4]

Problem: Everybody knows the recipient's public key, so anybody could have sent the message

Public-private key encryption

If you encrypt a message using your private key then the recipient's public key, this guarantees: [4]

- You sent the message, because it can be decrypted using your public key
- Only your recipient can read the message, because it can only be further decrypted using their private key

Combining symmetric and asymmetric encryption

Because symmetric encryption is simple, but not extremely strong, it is not ideal for most cases where security is important

Because asymmetric encryption is mathematically intensive and relatively slow, it is not very well-suited for most cases

Solution: Use both - asymmetric encryption to establish a shared *secret key*, which can be done over an unsecure channel, then use the secret key to communicate privately

Diffie-Hellman key exchange

Diffie-Hellman cryptography is a type of public key cryptography that makes it possible to establish a shared secret key even in a public, unencrypted manner

This is accomplished by having each party choose a secret key, which is not shared with anybody, and a common key, which is shared

Combining the common key and your secret key should be practically impossible to undo or replicate, even if the common key and result of common and secret key is known [1]

Diffie-Hellman key exchange

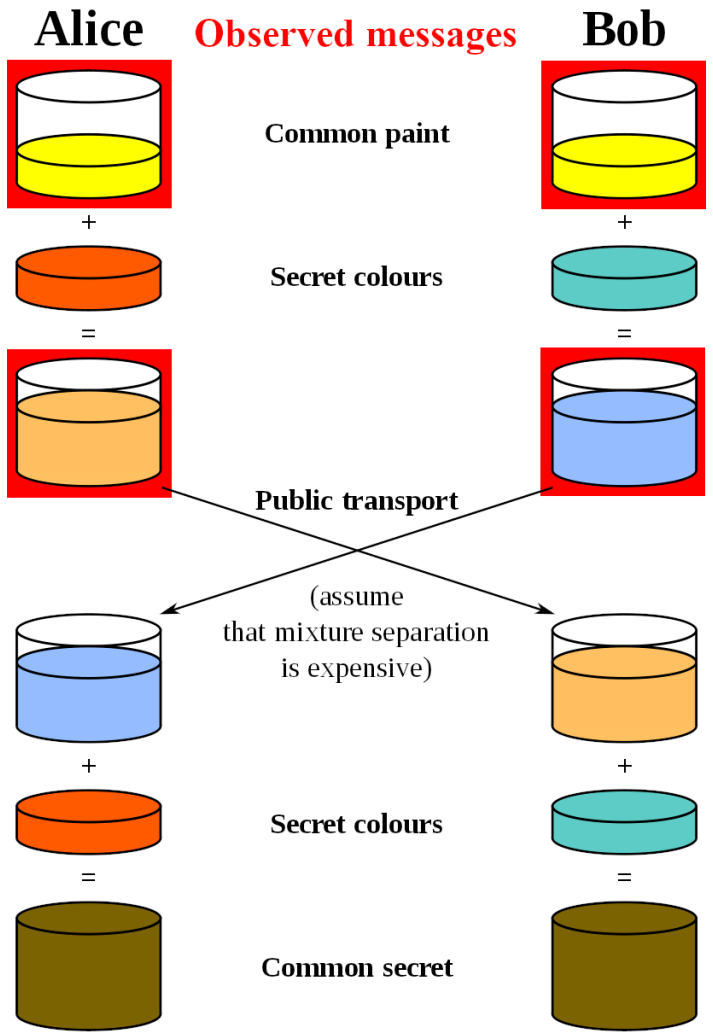

Image modified from Wikipedia [2]

Diffie-Hellman with public key cryptography

Diffie-Hellman can be implemented fairly simply using public key cryptography.

In its simplest form, you can simply choose something to use as your symmetric key, encrypt that with the recipient's public key, then send that to your intended recipient

In practice, this is very basic form is susceptible to MITM attacks, so typically other algorithms, such as RSA or STS, are used

What is GPG

GPG is a command line tool used to encrypt and decrypt data

GPG is the GNU Privacy Guard's, or GnuPG's,[9] implementation of the OpenPGP[10] standard for encryption and decryption

[9] GPG does **not** stand for GnuPG

[10] PGP stands for "Pretty Good Privacy"

Encrypting with your private key

Encrypting a message with your private key is known as "signing"

To encrypt a message with your private key using gpg:

```
echo <message> | gpg --sign
```

Or, to encrypt a text file:

```
gpg --sign <file>
```

You can also choose to have the signature in plain text and detached like so:

```
gpg --sign --clearsign --detach-sig <file>
```

Verifying a signature

To validate a signature using gpg:

```
gpg --verify <signed file>
```

Note that the public key corresponding to the private key used to sign the document must be imported into your keyring
The output will look something like this:

```
gpg: Signature made Mon May 20 18:26:04 2013 CEST using RSA
gpg: key ID D9C15D0D using PGP trust model
gpg: Good signature from "The Tcpdump Group (Package signing
gpg: key) <release@tcpdump.org>"
gpg: WARNING: This key is not certified with a trusted
gpg: signature! gpg: There is no indication that the
gpg: signature belongs to the owner.
```

Note that the third and fourth lines mean that anybody could have signed it

Encrypting with the recipient's public key

To encrypt a message with your recipient's public key, encrypt it like so:

```
gpg --encrypt --recipient <recipient-id> <file>
```

To encrypt your file using plain text, use the `--armor` flag, like so:

```
gpg --encrypt --armor --recipient <recipient-id> <file>
```

Decrypting files

To decrypt a file using gpg, you can use the decrypt flag:

```
gpg --decrypt <file>
```

Assuming the sender's public key is in your key ring, this is all that is required to decrypt

It will also verify the signature if of the message if one is included

References

[1] Cryptography - Public Key Encryption Algorithms
https://condor.depaul.edu/ichu/ds420/lecture/1030/public_key_encryp.htm

[2] Diffie-Hellman Key Exchange
https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

[3] Foundations of Computer Security https://www.cs.utexas.edu/users/byoung/cs361/lecture44.pdf

[4] Public Key Cryptography and RSA https://www.cse.wustl.edu/~jain/cse571-17/ftp/l_09pkc.pdf