Networking and Internet Services

Intro to networking Security

Presented by
Omid Panahi

Champlain
COLLEGE SAINT-LAMBERT

Problems with TCP

- TCP on it's own without extra security added has no security protecting it:
  - Wide open
  - Everything can be seen openly

General Security Pillars

- Encryption
- Integrity
- Nonrepudiation
- Authentication
- Authorization

- *Encryption*
  - To scramble, mix up, or change data.
  - This scrambled-up data must also be easily descrambled by the receiver of the data.
- *Integrity*
  - The process that guarantees that the data received is the same as originally sent.
  - Integrity is designed to cover situations in which someone intercepts your data and make changes.
- *Nonrepudiation*
  - is the process of making sure data came from the person or entity it was supposed to come from. This prevents others from pretending to be a different entity and doing evil things by impersonation like posting threats or stealing all your money.
- *Authentication*
  - Verification that whoever is trying to access the data is the person you want accessing that data.
  - The most classic form of authentication is username/password.
- *Authorization*
  - Defines what an authenticated person can do with that data.
  - Assign permissions to a user account.
  - An administrator, for example, can do a lot more after being authenticated than a limited user can do.

Encryption

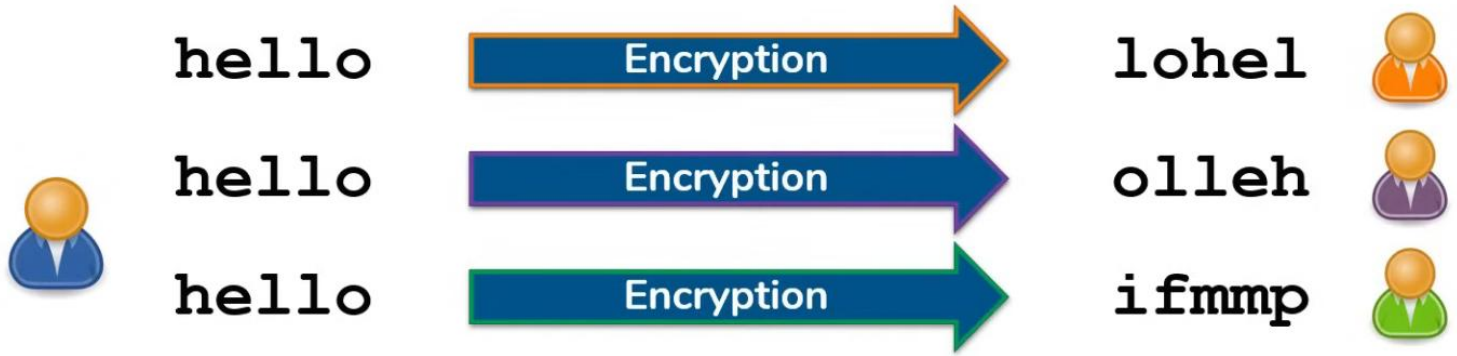
Encryption

- Cleartext
  - Unencrypted data (does NOT mean that is TEXT for sure).
- Cypher/Encrypt
  - The action of encrypting data
- Algorithm
  - The method that the data is encrypted

Encryption need heavy math algorithm otherwise , it can be easily predictable.

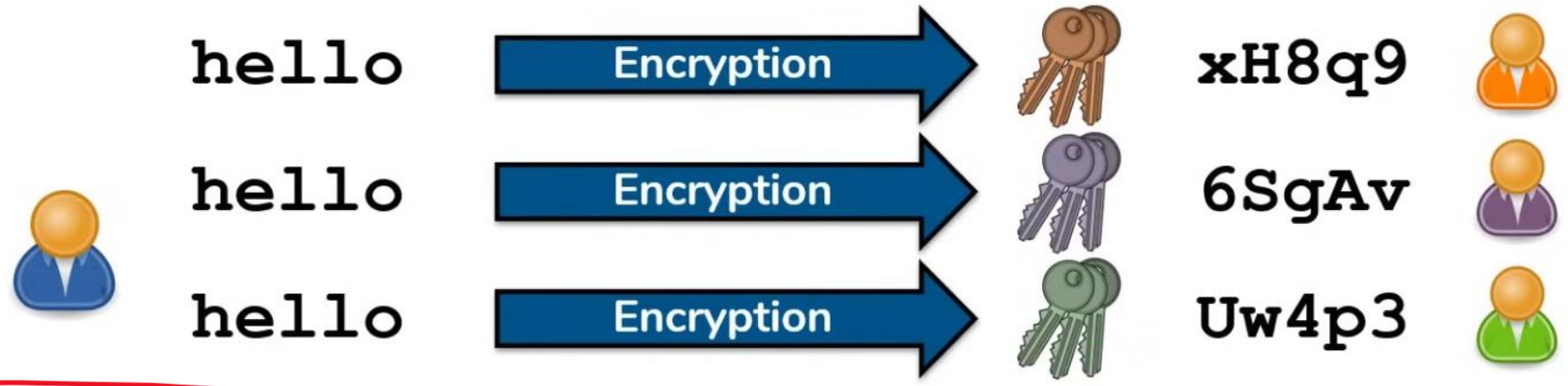
Encryption

- Encryption is used to provide Confidentiality
  - Confidentiality: Only intended recipient can interpret the Data



- Simple Encryption: Transforms Plaintext into Cipher text
  - Doesn't scale
  - Hard to do securely

Secret Key is a random key



- **Key Based** Encryption
  - Combines industry vetted algorithm with a Secret Key
    - Algorithm is created by experts
    - Secret Keys can be randomly generated

2 ways to Encrypt data with Keys

- Two types of **Key Based Encryption**:
  - **Symmetric Encryption**
    - Encrypt and Decrypt using the **same** keys
  - **Asymmetric Encryption**
    - Encrypt and Decrypt using **different** keys

In this example, we shift each character 3
So, our Secret Key =3

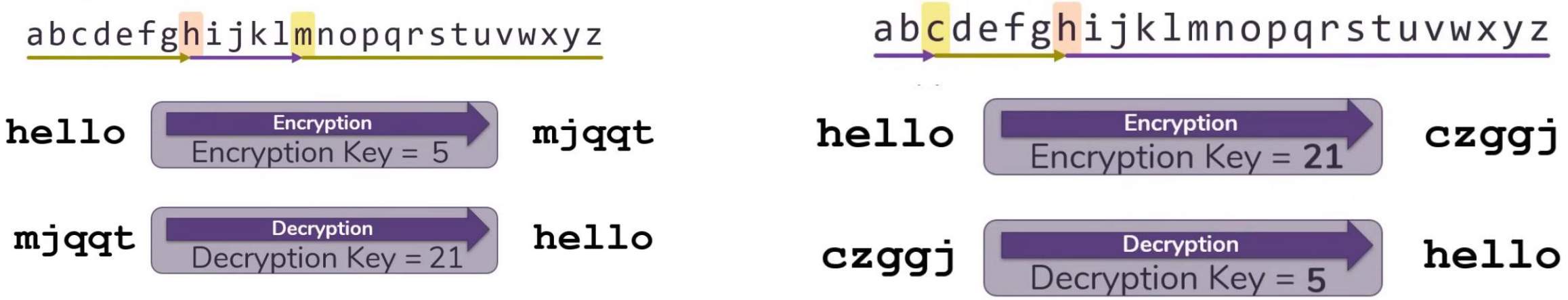
abcdefghijklmnopqrstuvwxyz

- Symmetric Encryption:

Symmetric Key-based algorithm is a 2-way algorithm(forward and backward)

But Asymmetric is 1 way algorithm

- Asymmetric Encryption:



Two different keys that are mathematically related
What one key Encrypts, only the other can Decrypt

These keys are mathematically related. So, that we can switch encryption with 21 and decryption with 5

- One key will be made Public → 
- Other key will be kept Private → 

This type of keys will be shared to others

We never share private keys

- Asymmetric Encryption — *Restricted to Limited Data*
  - Weakness: Slower – Requires much larger key sizes
  - Weakness: Cipher text expansion
  - Strength: Private Key is never shared – More Secure

- Symmetric Encryption — *Ideal for Bulk Data*
  - Strength: Faster – Lower CPU Cost
  - Strength: Cipher text is same size as Plain Text
  - Weakness: Secret key must be shared – Less Secure

- Asymmetric Encryption algorithms: *Restricted to Limited Data*
  - DSA
  - RSA
  - Diffie-Hellman
  - ECDSA
  - ECDH
- Symmetric Encryption algorithms: *Ideal for Bulk Data*
  - ~~DES~~ 56 bit key
  - ~~RC4~~ 128 bit key
  - 3DES 168 bit key
  - AES 128, 192, or 256 bit keys
  - ChaCha20 128 or 256 bit keys

XOR Encryption Example

- XOR:
  - 1 and 0 = 1
  - 0 and 1 = 1
  - 1 and 1 = 0
  - 0 and 0 = 0

Encrypt "HI" using XOR

- Encryption of the string "Hi" using an XOR encryption key.
- Choose a key. Let's use "10101010". Let's call this the "Private key".
- "Hi" = 01001000 01101001 (This is super-confidential text).

1. H → ASCII value 72
   72 in binary → **01001000**
2. i → ASCII value 105
   105 in binary → **01101001**

Example Encrypt

- Text : 01001000 01101001 (Hi)
- Key : 10101010 10101010 (You choose)
- XOR : 11100010 11000011 (This is the result)

Decrypt

1. H → ASCII value **72**
   72 in binary → **01001000**
2. i → ASCII value **105**
   105 in binary → **01101001**

- Enc Text : 11100010 11000011
- Key : 10101010 10101010
- XOR : 01001000 01101001 (Result Back)

Try this

- Private key
  - 11001100
- Encrypted text
  - 10011110 11111110 10001000 11111110
- Algorythm used
  - XOR

Symmetric Keys

- Usually the input is encypted in BLOCKS, not bit by bit.
- Eg: 128bit blocks of data at a time.
- The opposite of block cypher is "stream cypher"

- One standard called "DES" used a 64 bit block, and a 56 bit key.

- "AES" uses 128 bit block size, and 128 to 256 bit key.

Problem with Symmetric Keys

- If someone gets a hold of the key, they can encypt and decrypt data with it.
- A "Certificate" for a website, can be encrypted. If we had this key, we can "impersonate" that website.

Asymmetric-Key Algorithm Standards

- Uses a different key to encrypt and to decrypt.
- Private key : Decrypt
- Public key : Encrypt

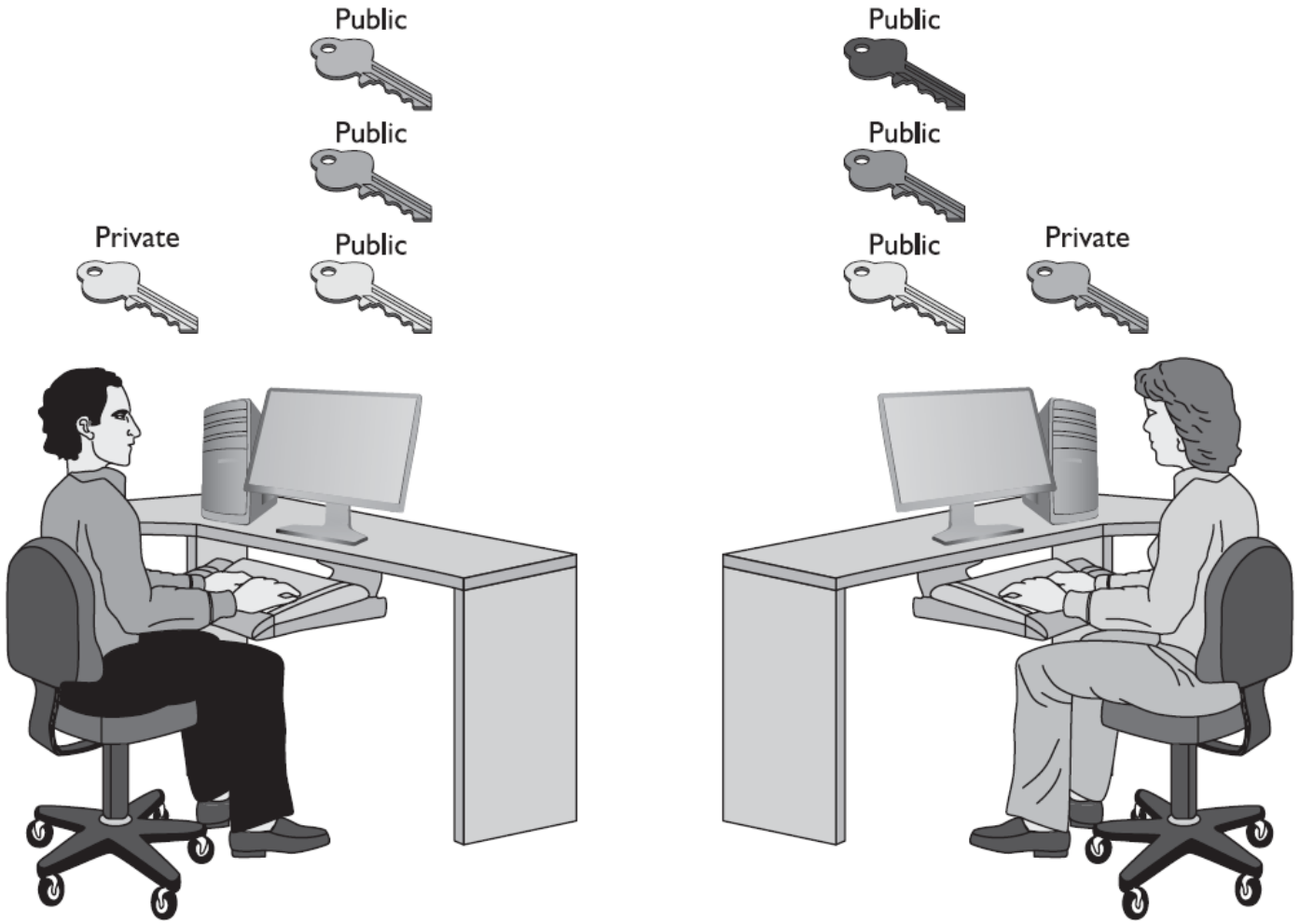
- Diffie, Martin Hellman, and Ralph Merkle are the inventors of this technology

- *RSA: Rivest Shamir Adleman (RSA)* enabled secure digital signatures

Example

- To exchange data, you need 2 keys
- Private key is kept by the person generating it.
- Public key is sent to the recipient.
- The recipient encrypts something using the public key.
- The private key is then used to decrypt.

- Notes
  - The private and public keys are called the "key-pair".
  - Private key can decrypt the message
  - Also public key can decrypt the message
  - There can be more than one different public key!


Public
Public
Private
Public
Public
Public
Public
Private

Integrity

HASHing

- Hash = *cryptographic hash function*
- A cryptographic hash function is a one-way function.
- Often this function returns a VALUE
- One-way means the hash is irreversible.
- You should not be able to re-create the data, even if you know the hashing algorithm and the checksum.

Why use hashing?

- To make sure that a file was downloaded correctly (and not corrupted during transmission).
- Ensure that it's not a "counterfeit" application, for example someone put a virus in your .exe installer.

Hashing

- *Secure Hash Algorithm (SHA) is the primary algorithm*
- It includes SHA-1, SHA-2 (which includes the popular

- HA-256 and SHA-512 variants)
  - Linux: shasum filename

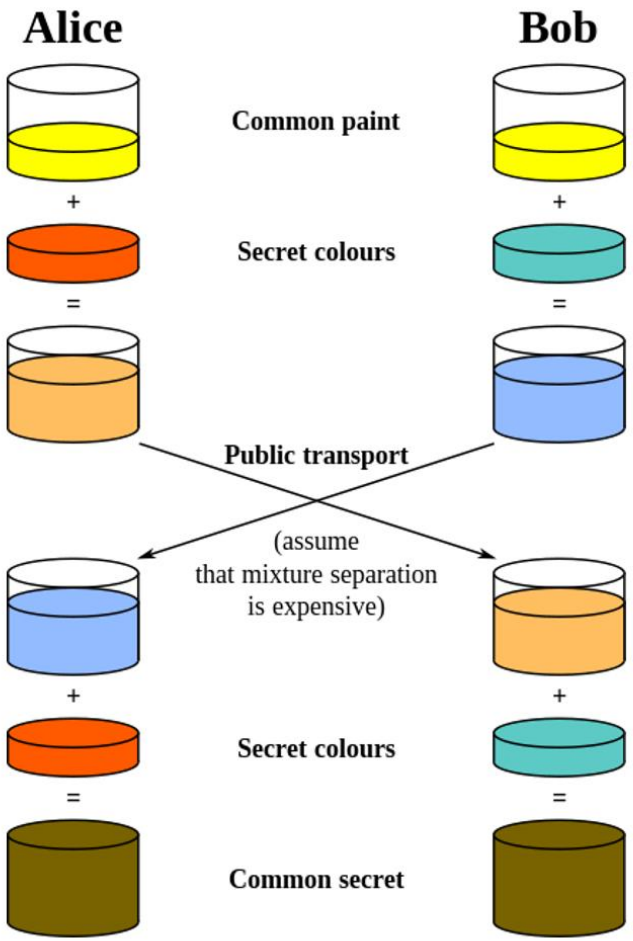
- Another type = MD5
  - Linux: md5sum filename

Encryption Standards

Public and Private Keys

Let’s watch this short Video

https://www.youtube.com/watch?v=cM4mNVUBtHk


Alice
Bob
Common paint
+
Secret colours
=
Public transport
(assume
that mixture separation
is expensive)
+
Secret colours
=
Common secret

Diffie-Hellman

Allows two parties
to **establish a shared secret**
over an **unsecured medium**

Shared Secret is never transmitted
Only values used to derive secret

Others(Internet)

Alice

Private = 5

(6^{5}) MOD 13
(7776) MOD 13
Public = 2

(9^{5}) MOD 13
(59049) MOD 13
Shared Secret = 3

Agree upon two numbers:

P Prime Number 13
G Generator of P 6

Randomly generate a Private Key

Calculate Public Key:

$(G^{Private})$ MOD P

Calculate the Shared Secret

$(Shared\ Public^{Private})$ MOD P

Everybody else knows 13(P),6(G),2(Alice public key) and 9(Bob public Key)

There is no way to get to 3 from these numbers

Bob

Private = 4

(6^{4}) MOD 13
(1296) MOD 13
Public = 9

(2^{4}) MOD 13
(16) MOD 13
Shared Secret = 3

Deffie-Hellman(DF)

- Establish Shared Secret over an unsecured medium
  - Shared Secret is then used to generated Symmetric Keys
- Security of DH is dependent on **Discrete Logarithm** problem
  - Exponentiation:
    - Given **G** and **X**, it is easy to find **N**
  - Logarithm:
    - Given **G** and **N**, it is difficult to find **X**

$$G^X = N$$

- Discrete Exponentiation:
  - Given **G**, **X**, **P**, it is easy to find **N**
- Discrete Logarithm
  - Given **G**, **P**, **N**, it is infeasible to find **X**
  - Only method is trying every possible combination (i.e., *brute force*)

$$G^X \bmod P = N$$

It takes thousands and thousands of years to brute force X value

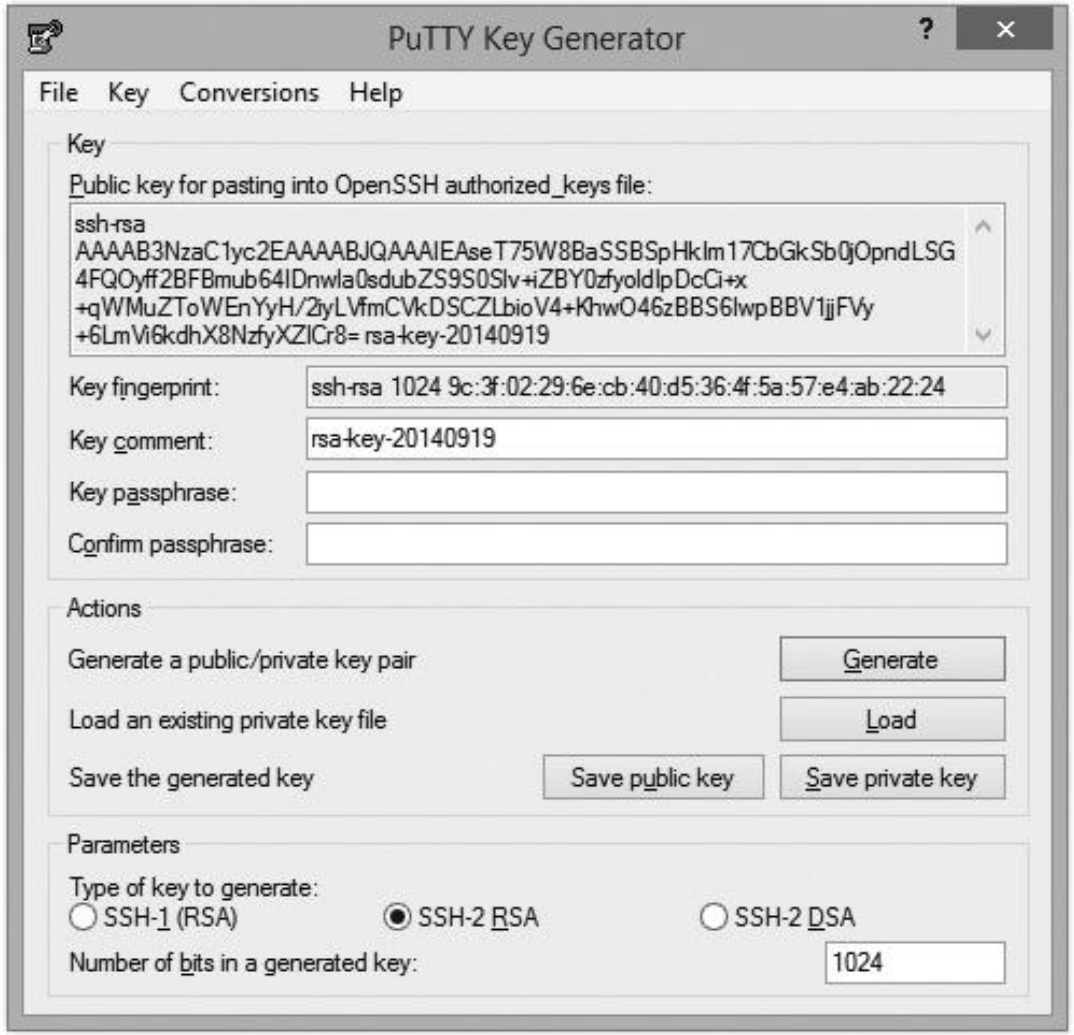
Lab(Encryption)

SSH

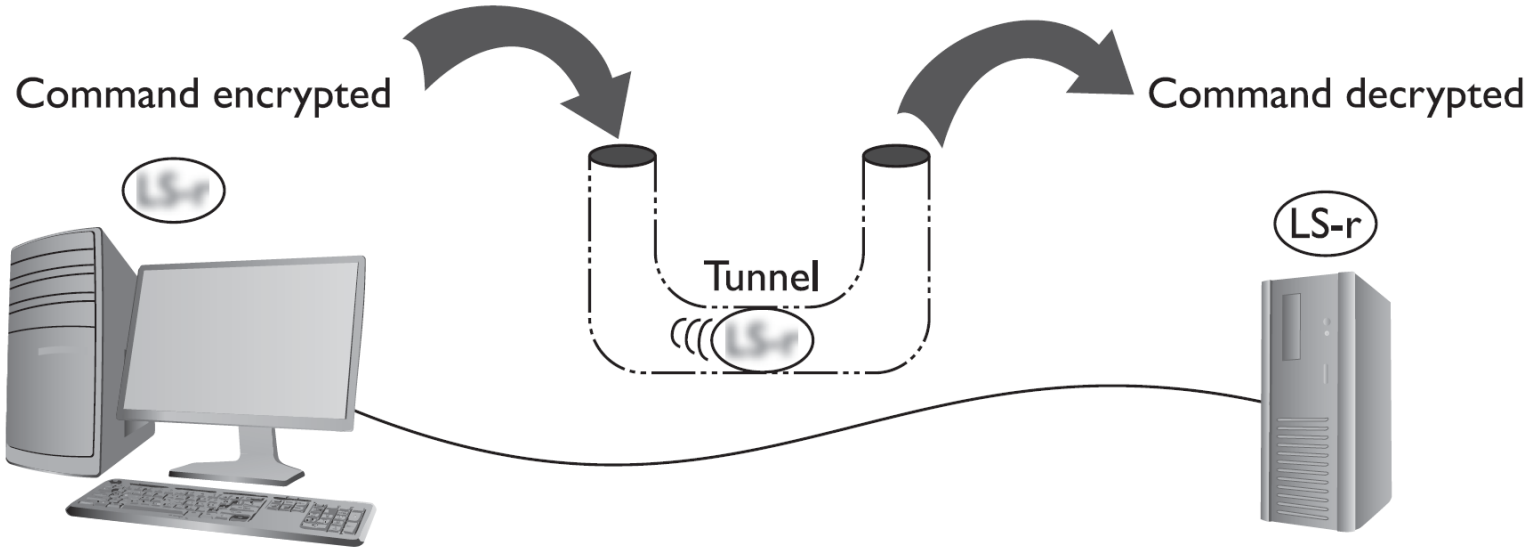
- Replaces TELNET (similar behaviour)
- More secure than TELNET
- SSH = Secure Shell
- Uses a PKI, which is an RSA key
- After the client receives this key, it creates a session ID, encrypts it using the public key, and sends it back to the server.
- The server decrypts this session ID and uses it in all data transfers going forward.
- SSH uses username and password to log in.
- Can use public keys also.

To use public/private keys

- Using PuTTYgen, you can generate public and private keys and install them to server and client.


PuTTY Key Generator
File Key Conversions Help
Key
Public key for pasting into OpenSSH authorized_keys file:
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIEAseT75W8BaSSBSpHkIm17CbGkSb0jOpndLSG
4FQOyff2BFBmub64IDnwla0sdubZS9S0Slv+iZBY0zfyoldlpDcCi+x
+qWMuZToWEnYyH/2iyLVfmCVkDSCZLbioV4+KhwO46zBBS6IwpBBV1jjFVy
+6LmVi6kdhX8NzfyXZlCr8= rsa-key-20140919
Key fingerprint: ssh-rsa 1024 9c:3f:02:29:6e:cb:40:d5:36:4f:5a:57:e4:ab:22:24
Key comment: rsa-key-20140919
Key passphrase:
Confirm passphrase:
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key
Parameters
Type of key to generate:
SSH-1 (RSA)
SSH-2 RSA
SSH-2 DSA
Number of bits in a generated key:
1024

Tunneling



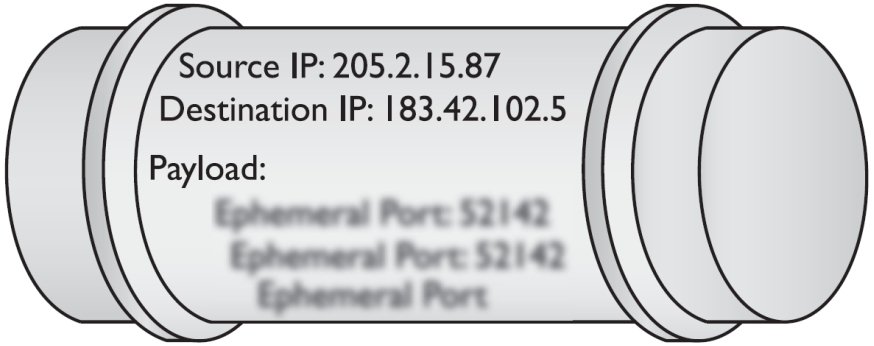
Tunnel

- An encrypted link between two programs on separate computers.
- When we go through a tunnel to access information, we call it "redirecting"
- Any packet that enters the encrypted tunnel, including a packet with unencrypted data, is automatically encrypted, goes through the tunnel, and is decrypted on the other endpoint.

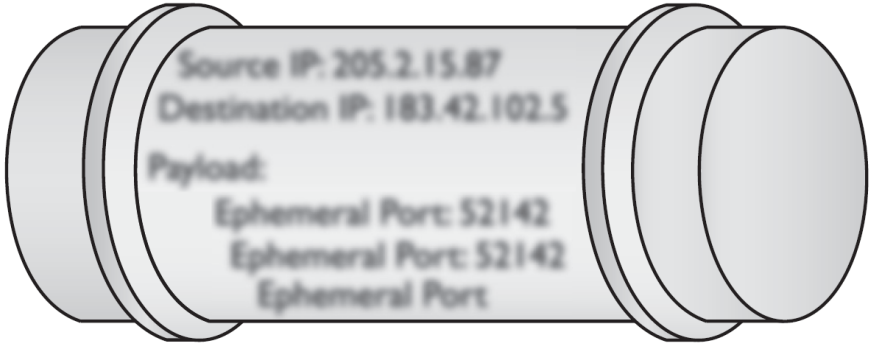
IPSEC

- Works on the Internet/Network layer of the OSI model.
- Works in 2 modes (Tunnel and Transport)
- Transport mode: Only payload (data) of packet are encrypted.
- Tunnel mode: All is encrypted within a tunnel.

Transport mode



Tunnel mode



Some common Auth/Encryp uses

- SSL/TLS
- Secure Sockets Layer requires a certificate. We use TLS to secure SSL.
- SSL is used to secure HTTPS connections.
- Your HTTPS connection is actually a tunnel.

Other uses Cont.

- FTPS: Secured FTP protocol

Champlain

COLLEGE SAINT-LAMBERT