Security

O. Aktouf

Based on material from M. Cherfaoui (CISCO) and D. Harkey (SJSU)

Objective

- Review the basic security solutions
- Analyze security solutions features
- Cloud computing uses similar aspects for security

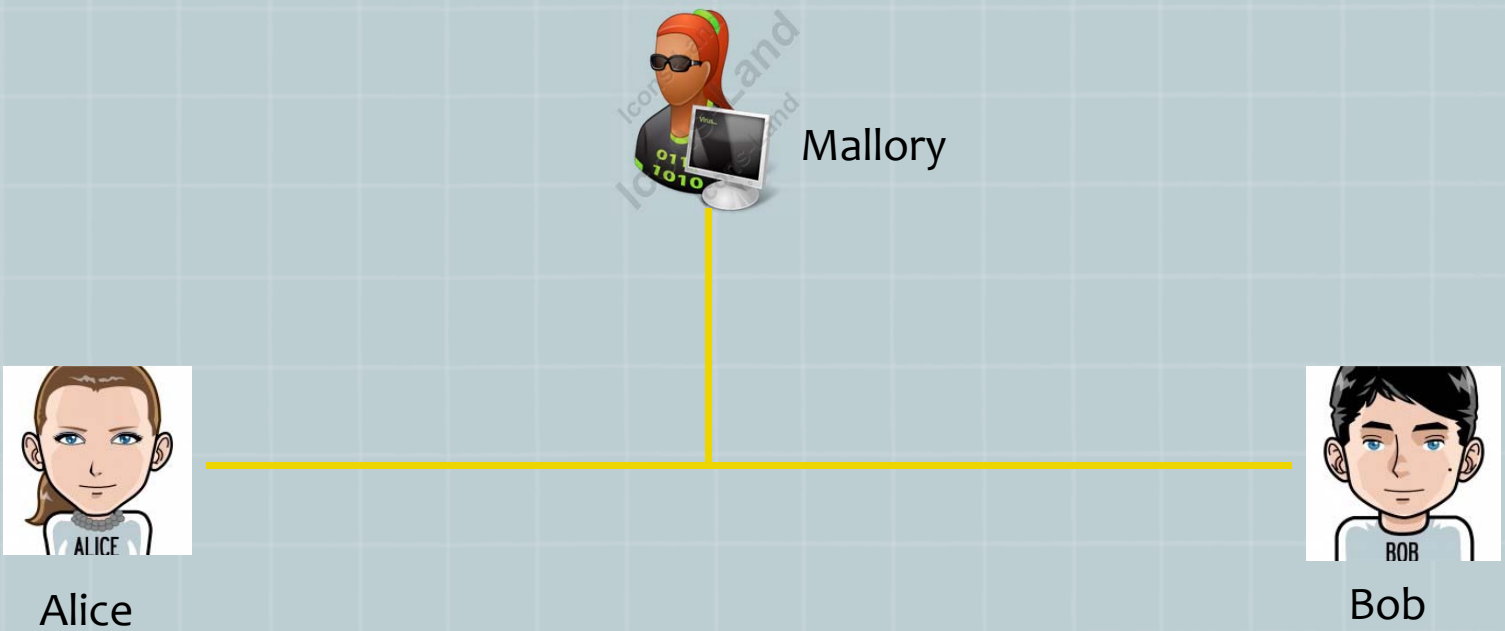
Introduction

- What is it all about?
  - ✓ Protecting resources, information, and people
  - ✓ Money, IP, reputation, national security
- Client/server environment introduces new security threats
  - ✓ Network is highly accessible
  - ✓ The system is mostly attacked through a weak link
- Types of Threats
  - ✓ Data Loss
  - ✓ Viruses
  - ✓ Identity Theft
  - ✓ Denial of Service
  - ✓ System Intrusions/Unauthorized use
  - ✓ Public embarrassment
  - ✓ Political/Hacktivism

Security Basics

- **Authentication**
  - ✓ Challenging the connecting person or application to identify themselves
- **Authorization**
  - ✓ Granting the requestor the right to an action or a resource
  - ✓ Are you allowed to use this resource
- **Data Protection and Integrity**
  - ✓ Integrity: Is my in-transit data safe?
  - ✓ Limiting access to authorized persons
  - ✓ Ensuring the content of the data has not been corrupted

The Actors



The Problem

- Alice and Bob are talking or exchanging information on the public internet. Some of their communication is secret and they don't want it to leak out.
- The obvious solution is to perform some transformation on the exchanged data so that Mallory can't understand it.
- Naïve solution: replace each letter in the message with the 3rd following letter in the alphabet (wrap around if needed).

Kerckhoffs's principle (reformulated)

- Always assume that the enemy knows or will know the transformation.
- Make the transformation (or algorithm) public and define a parameter that should remain secret.
- Parameter = key. Transformation = Encryption.
- In summary: use a well known encryption algorithm with a secret key.
- Rule: the more people are aware of the algorithm, the more flaws are discovered and corrected. We say that the algorithm has been vetted.

Secret Talk

- Alice and Bob decide to encrypt their communications using a well known, vetted, encryption algorithm.
- They need to exchange an encryption key that is only known to the 2 of them.
- The key will actually be used for both encryption and decryption. It's a symmetric key.

Equations

- Assume C is the clear text and E the encrypted text (or cipher text).
- k is the symmetric encryption/decryption key
- T and T^{-1} are the encryption and corresponding decryption transformation.
- We can write the following equations:

$$E = T(C, k) \quad (1)$$

$$C = T^{-1}(E, k) \quad (2)$$

Issues with a Simple Encryption

- Exchanging the key
- Authentication of the key exchange
- Scalability
- Non-repudiation
- Message integrity
- Key randomness
- Forward Secrecy
- Replay attack

Exchanging the Key

- Obviously, the key can't be exchanged via the communication channel.
- It has to be exchanged "out-of-band"
- Example of out-of-band channels: email, regular mail, phone, in person.

Authentication of the Key Exchange

- Even if Alice and Bob agree on an out-of-band key exchange method, Mallory can still do harm if she manages to send her own key to Alice and Bob and make each believe it comes from the other one. She will then be able to decrypt any communication between the 2.
- She could craft 2 emails with Alice's and Bob's email addresses.
- If she is a good voice impersonator, she can trick Alice and Bob on the phone.
- What Alice and Bob need is a way to know for *sure* that the key is coming from the other trusted party. This is called authentication.

Scalability

- **Assuming Alice and Bob find a way to do an authenticated exchange of the key, there will be *scalability* problems.**
- **If either Alice or Bob wants to communicate secretly with another party, they will need another key for that communication and they need to solve the authenticated exchange problem again with that party.**
- **What if they need to communicate with 20 or 30 other people?**

Non-Repudiation

- Assume Alice and Bob share an encryption key.
- Bob encrypts a file containing questionable contents with the key and law enforcement finds the file on the Internet and manages to decrypt it with Bob's key. Bob can deny the file is his. In other words he can repudiate the fact. Law enforcement can't challenge him since Alice has also the key.
- Non-repudiation is the desired property and symmetric encryption does not provide it.

Message Integrity

- Alice and Bob managed to exchange the encryption key without Mallory being able to know it. Mallory won't be able to "see" the contents of the traffic between Alice and Bob but can still do harm in another way.
- Mallory can change the contents of Alice' message to Bob in transit, in other words, corrupt the message.
- Depending on the type of data in the message, Bob won't be able to know that the message was "corrupted". For example, if the message contained a spreadsheet of financial data, detection of the corruption is almost impossible.

Key Randomness

- **Alice and Bob need to generate encryption keys that are almost impossible to guess by Mallory.**
- **The randomness of the key is very important**

Forward Secrecy

- Bob and Alice managed to exchange a very randomly generated encryption key and happily communicating on the Internet.
- Mallory is recording all their communications even if she can't decrypt them.
- 3 years later, she manages to guess the key.
- Now she can decrypt not only the current communication but past communications as well!
- What Bob and Alice need to do is to change the key on a regular basis.

Replay Attack Protection

- Mallory watches a message sent by Alice to Bob but can't decrypt it.
- However, she observes that upon receiving the message, Bob opens the front door of the bank.
- Mallory concludes that the message instructs Bob to open the front door and only has to "replay" the same message to obtain the same result and allow her partners to enter the bank and do a hold up.
- To solve this problem, most encryption algorithms prepend a "salt" to the clear text before encryption. Salt is a random value.

Additional Cryptographic Tools

- **The previous limitations show the need for additional tools beside simple encryption/decryption.**
- **Some of these tools are:**
  - ✓ **Public/Private key Hashing and MACs**
  - ✓ **Digital Signatures**
  - ✓ **Key exchange**
  - ✓ **Secure random number generation.**

Public key cryptography

- The idea is to use a pair of keys for encryption and decryption. The public key (k_{pub}) is known to everyone and the private key (k_{priv}) is only known to the owner. Public key cryptography is also called asymmetric encryption.
- We have the following equations:

$$E = T(k_{pub}, C) \quad (1)$$

$$C = T(k_{priv}, E) \quad (2)$$

Note 1: the same transform T is used in both equations.

Note 2: to each public key k_{pub} corresponds a unique private key k_{priv}.

How Can it Help?

- With Public key cryptography, we solve the scalability problem!
- Alice just needs to give her public key to all the other parties she needs to communicate with.
- They can then encrypt their communications with Alice' public key. Alice can decrypt the communications with her private key.

Asymmetric Encryption

- **Asymmetric encryption solves nicely the scalability issue.**
- **However, asymmetric encryption is very slow compared to symmetric encryption. One solution is for the communicating parties to encrypt a symmetric key with the public key of the owner and encrypt the communication with the symmetric key.**

Combining Symmetric and Asymmetric Encryption

Bob wants to send a message C to Alice. He knows her public key. Bob will do the following:

1. **Bob generates a symmetric key k.**
2. **Bob encrypts it with Alice's public key and sends it to her.**
3. **Bob and Alice can now talk securely using key k.**

More on Public Key Encryption

- Public key encryption solves the scalability problem but not necessarily the other problems
- For example, we still need a way for Bob to make sure that the public key sent to him by Alice is hers without a doubt. In other words, authentication is needed.
- We will see later how authentication can be achieved.

Hashing (1)

- Hashing is a technique that allows to produce a "fingerprint" of some data.
- A simple hashing technique would be to produce 0 or 1 depending on whether the binary sequence representing the data is odd or even (last bit is 1 or 0).
- However, this simple hashing and similar techniques cause many collisions, i.e. many data hash to the same value.
- Hashing is used for many applications: error detection, integrity checks, lookup table. We will focus on the integrity check application.

Hashing (2)

- A desirable property for hashing in cryptography is that given a hash value, it's impossible to find data that hashes to that value.
- In the integrity check application, Alice can do the following in order to send a text C to Bob:

1. E = T (C, k)
2. m = hash (C)
3. Send E and m to Bob
4. Bob recovers C using $C = T^{-1}(E, k)$, computes m' using m' = hash (C)
5. If m == m', Bob concludes that C has not been corrupted in transit.

Message Authentication Code (MAC)

- Sending a hash along with the message has been proven to be not secure.
- Instead a MAC is used. A MAC involves some form of hashing of the combination of the data to be integrity-checked and a secret key.
- The previous steps become:

1. E = T (C, k)
2. m = MAC (C, k')
3. Send E and m to Bob
4. Bob recovers C using $C = T^{-1}(E, k)$, computes m' using m' = MAC(C, k')
5. If m == m', Bob concludes that C has not been corrupted in transit.

Note: k and k' can be the same.

Passwords and Hashing

- Hashing finds another application in passwords protection.
- Instead of storing passwords in clear text or encrypted format, a hash of the password is stored.
- The Operating System or application hashes the password that a user provides and compares it to the stored password.
- To avoid “rainbow table” attacks, password are salted with a random value before hashing and storage.

Digital Signatures

On top of providing integrity protection, MAC provides some authentication: only the owner of the secret key could have produced the MAC.

However symmetric encryption suffers from 2 drawbacks: non-scalability and repudiation.

Integrity protection and authentication are better supported by public key cryptography.

Integrity protection + authentication with public cryptography = Digital Signature.

Digital Signatures in Equations

Alice wants to assert that text C is from her and has not been corrupted. She can use her private key k_{priv} to produce a signature S. The sequence of operations is:

1. m = hash (C)
2. S = T (k_{priv}, m)

Now she can send text C to anyone along with signature S. Everyone will be able to verify the signature by using her public key k_{pub}:

1. m = hash (C)
2. m' = T (k_{pub}, S)
3. m == m' ? => C has not been corrupted and it came from Alice!

Digital Signatures Properties

- Digital signatures are scalable since the signer can use the same private key for everyone.
- Digital signature provides non-repudiation: the signer cannot deny he signed a message since he is the only one who has the private key.

Authenticating Public Keys

- We saw that public key cryptography offers scalability. However recipients of the public key need a way to authenticate the origin of the public key.
- This can be done by an authority that everyone trusts and who would digitally sign the information mapping an owner to her/his public key.
- Everyone trusts the authority by *knowing* his public key.

Third Party Digitally Signed Public key

- A trusted third party can certify the truthfulness of the public key of Alice k_Alice_{pub} using his private key $k\_Authority_{priv}$ in the following way:

    1. claim = …||k_Alice_{pub}||Alice||…–|| is a concatenation
    2. m = hash (claim)
    3. cert = T ($k\_Authority_{priv}$, m)

- Now, anyone who trusts Authority will know for sure that Alice' public key is k_Alice_{pub} upon seeing claim and cert.

Digital Certificates

- **The Pair (claim, cert) is called a digital certificate**
- **Main fields in a certificate:**
  - ✓ **Issuer: the authority that validated the claim**
  - ✓ **Subject: the entity on behalf of whom the claim is made (e.g. Alice)**
  - ✓ **Public key: the public key of the subject**
  - ✓ **Date of validity: validity period of the claim**
  - ✓ **Digital signature: = T (k_Issuer_{priv}, claim fields other than digital signature)**

More on Certificates

- X509 is the standard that defines how certificates are formatted.
- Latest version is 3.

Certificate Example

- **Digital signature which includes a public key and general information about the individual**
- **Based on X.500**
- **Signed Jars, Web browsers (SSL), secure email, B2B, …**
- **Certificate Authority (CA) vouches for the integrity of the certificate holders data, not the holder's integrity.**
  VeriSign is an example of a CA
- **Contains** (CN = Common Name, OU = Organizational Unit, O = Organization, C = Country)

```
Version Number
Certificate Unique ID
Signature Algorithm
Issuer (X.500 based)
Subject's Name (X.500 based)
Subject's Public Key

CN=John Smith, OU=College of Engineering, O=SJSU, C=US
```

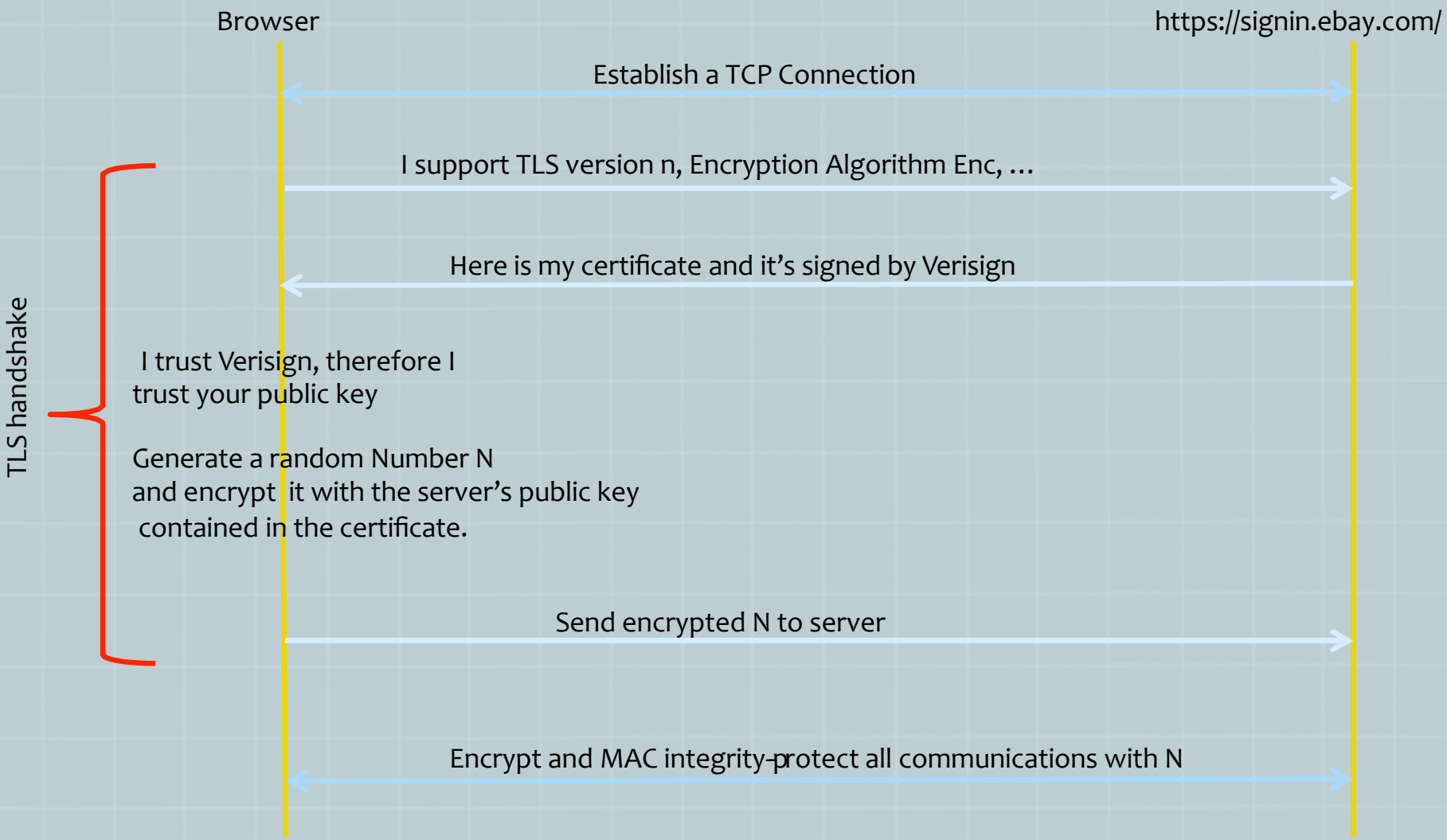
Certificate Example

Version: 3 (0x2)
Serial Number:
15:1e:01:a5:a8:57:2f:b9:e9:53:e5:cb:52:f4:64:62
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=US, O=VeriSign, Inc., OU=VeriSign Trust Network, OU=Terms of use at https://www.verisign.com/rpa (c)06, CN=VeriSign Class 3 Extended Validation SSL CA
Validity
Not Before: Nov 30 00:00:00 2012 GMT
Not After : Jan 24 23:59:59 2015 GMT
Subject: 1.3.6.1.4.1.311.60.2.1.3=US/1.3.6.1.4.1.311.60.2.1.2=Delaware/businessCategory=Private Organization/serialNumber=2871352, C=US/postalCode=95125, ST=California, L=San Jose/street=2145 Hamilton Ave, O=eBay, Inc., OU=Site Operations, CN=signin.ebay.com
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (2048 bit)
Modulus (2048 bit):
00:aa:1e:28:0f:2b:fc:3b:ca:d4:48:07:7c:ed:da:
de:c6:13:b9:af:02:91:ac:4b:49:14:b4:4b:0b:1d:
<omitted>
bc:5c:12:83:e0:47:9c:5a:84:8c:ac:7f:24:32:ea:
db:13
Exponent: 65537 (0x10001)
X509v3 extensions:
<omitted>
Signature Algorithm: sha1WithRSAEncryption
26:d2:5b:45:f4:01:f6:50:af:2d:ed:70:0d:85:9e:c0:d2:8c:
<omitted>
d0:97:2d:be:cf:01:8e:96:ba:c6:d3:35: d : 5 0 : 5 3:26:e7:8
c: bb:95:40:60

First Application of Cryptography

- We have seen many of the building blocks that are used to secure the Internet and Cloud applications.
- One of the most important applications is the TLS:Transport Layer Security (SSL: Secure Socket Layer, in older versions) protocol.
- https is an SSL/TLS implementation for the web.
- Let's look at how a browser uses https.

https Flow (Simplified)



SSL/TLS Client Authentication

- Base SSL/TLS only authenticates the server.
- The Client can authenticate once a “tunnel” is established between the client and the server using a username/password for example.
- Optionally, a server can request a client to authenticate with a certificate.

More on SSL/TLS

- SSLv1 and SSLv2 were the first versions. Their use is discouraged because of known vulnerabilities.
- SSLv3 is still used today.
- TLSv1.0 is very close to SSLv3
- TLSv1.1 and TLSv1.2 are the latest versions and their use is strongly recommended.
- SSL/TLS is limited to protecting TCP traffic. DTLS: Datagram Transport Layer Security is a close protocol that can secure UDP traffic.

Key Exchange

- One issue with symmetric encryption is that the encryption key needs to be exchanged out of band.
- What if the key can be negotiated between Alice and Bob on the communication channel without Mallory being able to see it?
- This is what the Diffie-Hellman (DH) does. Alice and Bob can derive the same key locally after exchanging data. The key never leaves Alice' or Bob's machine.
- Mallory can't infer the key from the exchanged data.

Diffie-Hellman (DH)

1. Alice and Bob agree on a prime number p and base g. p and g are known to Mallory.
2. Alice and Bob each choose a secret number a and b respectively.
3. Alice sends Bob A = g^a mod p, Bob sends Alice B = g^b mod p
4. Alice computes s = B^a mod p, Bob computes s = A^b mod p
5. Since both Alice and Bob compute the same value s, they will use it to encrypt their communications. Proof:
    6. s = B^a mod p = $(g^b)^a$ mod p = g^{ba} mod p = g^{ab} mod p = $(g^a)^b$ mod p = A^b mod p

DH

- DH does not provide authentication. Alice and Bob cannot be sure that the data used in the DH exchange really comes from the other party and not from Mallory.
- This can be solved by Alice and Bob digitally signing the DH data exchange with their private keys and each party validating the signature with the public key of the other party.
- DH supports forward secrecy!

ssh

- ssh is another protocol used in Cloud Computing to get a secure shell on a remote server.
- ssh is based on DH for the key exchange and public key cryptography for the server authentication.
- The ssh client authentication supports 2 modes: password based and public key cryptography.
- ssh does not assume that the client and the server trust a common known authority.

ssh Server Authentication

- Initially the client does not know the public key of the server:

```
MCHERFAO-M-T2JD:~ mcherfao$ ssh vos-cm100
The authenticity of host 'vos-cm100 (10.195.96.100)' can't be
established. RSA key fingerprint is
91:6b:68:65:7c:a9:4c:a9:de:9e:15:6b:22:f1:20:ed.
Are you sure you want to continue connecting (yes/no)?
```

- Once the server is trusted, the public key gets added to ~/.ssh/known_hosts.
- From then on, the connection proceeds without asking.

ssh Server Authentication (2)

- **If the public key of the server changes, the user is prompted to validate:**

```
MCHERFAO-M-T2JD:~ mcherfao$ ssh vos-cm100
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle
attack)! It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
91:6b:68:65:7c:a9:4c:a9:de:9e:15:6b:22:f1:20:ed.
Please contact your system administrator.
Add correct host key in /Users/mcherfao/.ssh/known_hosts to get rid of
this message.
Offending RSA key in /Users/mcherfao/.ssh/known_hosts:21
RSA host key for vos-cm100 has changed and you have requested strict
checking. Host key verification failed.
```

ssh Client Authentication

- ssh client can authenticate themselves to the ssh server using username password.
- They can also generate a public/private key pair and use it to authenticate to the server.
- Run the "ssh-keygen" to generate a public/private key pair on the client. The private key will be in ~/.ssh/id_rsa and the public key in ~/.ssh/id_rsa.pub
- Copy ~/.ssh/id_rsa.pub to the sever's ~/.ssh/ authorized_keys file.
- From now on, the client does not need to enter the username/ password. This is useful when the client is not a human user. This mechanism is widely used in AWS and OpenStack.

VPN

- VPN stands for Virtual Private Network.
- VPN establishes a "tunnel" between 2 hosts, 2 networks or a host and a network. All traffic between the 2 peers goes through the tunnel.
- A VPN is not necessarily encrypted but in these slides, we will focus on VPNs that provide security.
- There are 2 main secure VPN technologies: IPSec based VPN and SSL based VPN.
- Note that applications running on a host don't have to authenticate/encrypt/integrity protect their traffic. In fact, they don't even know that their traffic is eventually encrypted.

IPSec VPN

- IPSec is the most common technology for securing VPNs.
- IPSec leverages directly the IP layer, i.e. it runs in parallel with the TCP and UDP protocol. IPSec uses IP protocol numbers 50 and 51 (TCP uses 6 and UDP 17)
- All the packets entering or leaving the host get “intercepted” first by IPSec and “processed” (i.e. encrypted and/or authenticated).
- Security material between the peers is negotiated via UDP port 500 using a framework called ISAKMP (Internet Security Association and Key Management Protocol)
- IPSec is a very powerful protocol but suffers from complexity and poses configuration and troubleshooting challenges.

SSL VPN

- SSL VPN has been gaining traction in the last years.
- SSL VPN establishes an SSL/TLS connection between 2 peers and all the traffic is tunneled through it.
- OpenVPN is a popular implementation

Randomness

- True randomness is needed to generate keys (and salts). Without true randomness, an attacker can guess the key.
- Don't use Java's Random Number Generator classes (e.g. java.util.Random). These classes don't generate true random numbers. They use an algorithm to generate a sequence of "pseudo" random numbers. Knowing a number in the sequence, an attacker can guess the next one.
- Note: Java does provide a class that generates true random numbers: java.security.SecureRandom

Collecting Randomness

- So how does a programming language runtime collect randomness?
- Most modern OSs provide a randomness source (or entropy) to applications. They do so by timing random system events (key press, disk access, packet received, etc.) and feeding a randomness collector, generally a file.
- On most Unix systems, randomness is collected in /dev/ random. Windows uses a similar approach.

Standards

- Encryption: 3DES, RC4, AES128, AES256
- Encryption uses “modes” of operation, e.g. AES128-ECB, AES128-CBC
- Hashing: MD5, SHA1, SHA2 (SHA256, SHA512)
- Public/Private key: RSA, DSA, Elliptic Curve
- Key negotiation: Diffie-Hellman
- Secure Hash (MAC): HMAC