Computer Network Programming

Elements of Secure Communication

Dr. Petru Florin Mihancea

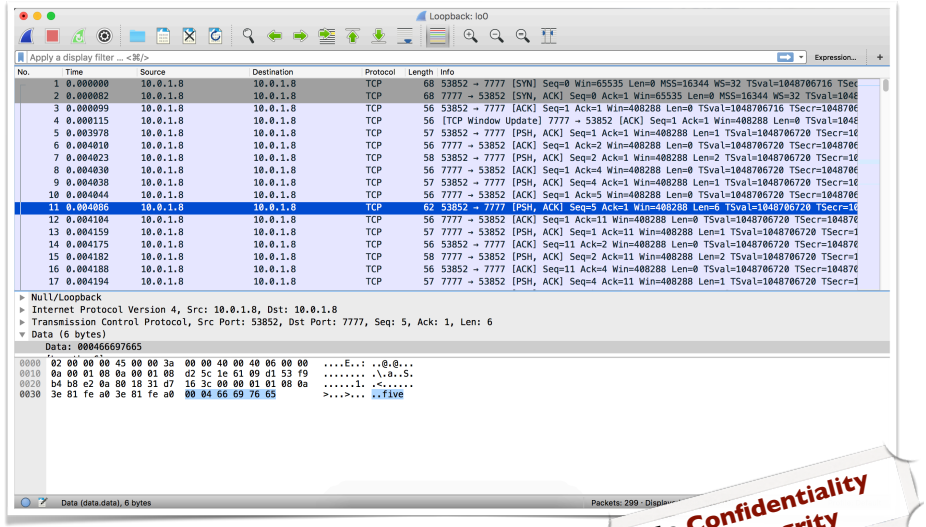
1

The Problem

Remember our Demo with TCP Sockets?

Can one see what is actually passed through the network?

Remember our Demo with TCP Sockets?



No Confidentiality
No Integrity

2

Some Security Concepts

addressing confidentiality and integrity

Just remember that many other security issues exist in distributed systems e.g. availability

Symmetric (Share Secret) Key Cryptography

Communicating nodes (Alice and Bob) have a secret (only them know it) key K_{AB}

- A sends the encrypted message $\{M\}K_{AB}$
- Only B can decrypt and extract M knowing K_{AB}

Algorithms

Encrypt so that is computationally unfeasible to decrypt without knowing the key or by trying all the keys
e.g., ~~DES~~, IDEA, ~~RC4~~, AES

Pros/Cons

Fast & can ensure secrecy, integrity and authentication (if the key is not compromised)
Hard to establish a common secret key in a large distributed environment

Asymmetric (public) Key Cryptography

Receiver node (Bob) has a pair K_{B_Public}-$K_{B\_Private}$

- Alice sends the encrypted message $\{M\}K_{B\_Public}$
- Only B can decrypt and extract M using its non-disclosable $K_{B\_Private}$

Algorithms

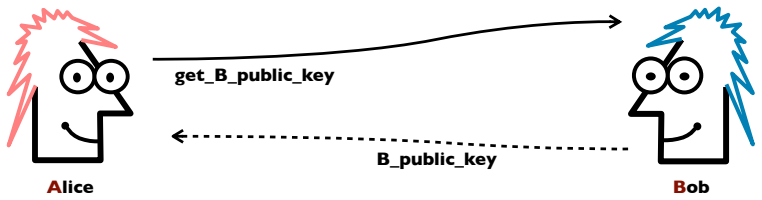
Encrypt so that is computationally unfeasible to decrypt without knowing the private key, by trying all the keys or trying to infer the private key
RSA

Cons/Pros

Significantly slower that symmetric key cryptography, must be sure of the ownership of the public key
Enables digital signatures and … establishing a common secret key to apply symmetric cryptography next :)

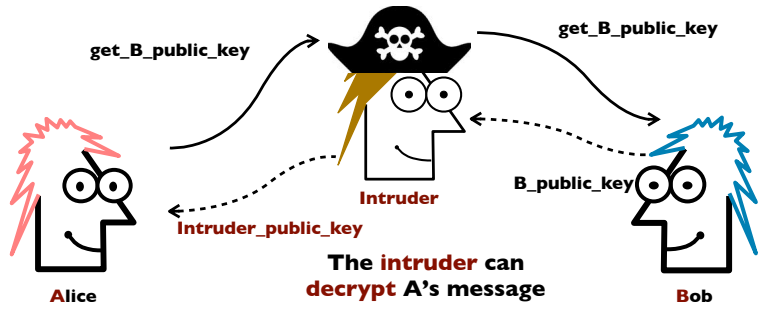
Public-Key Distribution Problem

How does Alice know K_{B_Public} is really Bob's?



Public-Key Distribution Problem

How does Alice know K_{B_Public} is really Bob's?



Man-In-The-Middle Attack

Public-Key Certificate

A document guarantying something

e.g., the ownership of a public-key
Digitally signed by an issuer

But how does Alice know that the Issuer is right (the validity of K_{Issuer_Public})?

Digital signature

The issuer has his own K_{Issuer_Public} - $K_{Issuer\_Private}$ pair
His signature is $\{Digest(Document)\}K_{Issuer\_Private}$
Digest functions are secure hash functions e.g., ~~MD5~~, ~~SHA~~, SHA2
Anybody can verify the signature on a document
Decrypt the signature using the K_{Issuer_Public} and compare the digests

Checking the certificate of the issuer that is issued by a greater authority, and then the certificate of that authority and so on

Until a trusted authority is reached in this chain

3

keytool

The Java Key & Certificate Management Tool

Frequent commands (1)

```
keytool
    -genkeypair -alias key_pair_name
    -keystore filename.jks -storetype jks
```

Many other options are possible to control the size of the keys, the algorithm, etc.

Generates a new public-private key pair
- with a given name (alias)
- saved in the specified keystore having the given storetype format
- the public key is wrapped into a digitally self-signed certificate
- you'll be asked (among others) about the Common Name (CN) referring to the name you are associating to the public-key

```
keytool
    -certreq -alias key_pair_name -file filename.csr
    -keystore filename.jks -storetype jks
```

Generates a certificate signing request file
- for the given key pair (alias) from the given keystore
- next, you should follow the signing authority procedure

Frequent commands (2)

keytool
-importcert -alias name -file certificate.cer
-keystore filename.jks -storetype jks

Records the certificate from the given file in the given keystore
- **replaces the certificate if a key pair with the given alias exists**
  used to record the signed certificate got from an authority
- **otherwise, adds a trusted certificate entry in the keystore**
  used to mark an authority certificate as trusted; cacerts keystore in the Java home folder already contains the certificates of many world-wide trusted authorities

keytool
-exportcert -alias key_pair_name -file certificate.cer
-keystore filename.jks -storetype jks

Saves the certificate in the given file
- **associated to a given entry (alias) from the given keyst**

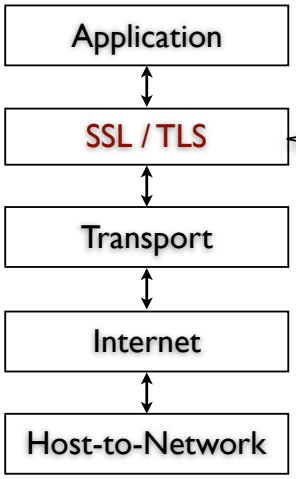
Be careful at the certificate file format e.g. DER or PEM

4

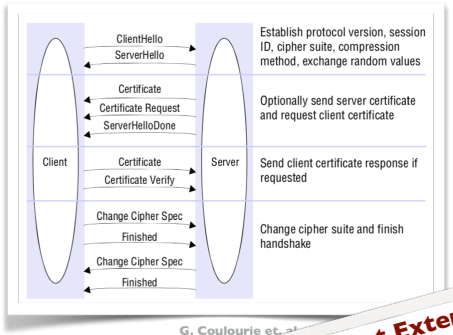
SSL / TLS

Secure Sockets Layer / Transport Layer Security and corresponding API in Java

Transport Extension



Ensures transparent data transport between application processes, guaranteeing secrecy, integrity and (optionally) partner authentication

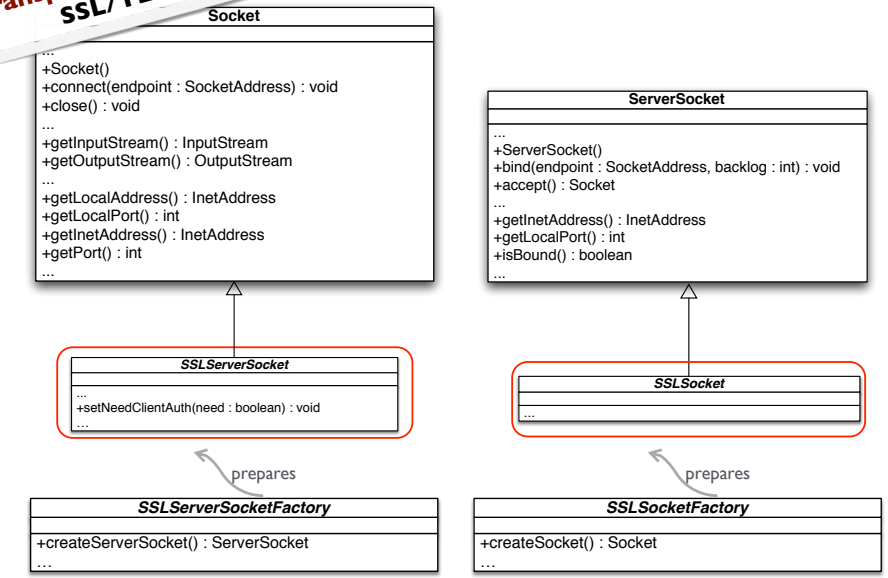


G. Coulouris et. al

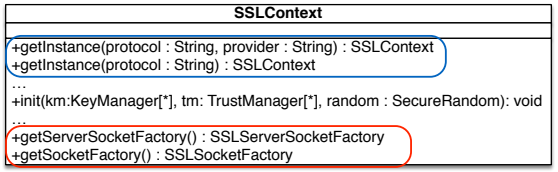
Java Secure Socket Extension (JSSE) provides a framework and an implementation in Java

Secure TCP Sockets

Client/Server code works transparently with simple or SSL/TLS sockets



Obtaining a SSL/TLS Implementation



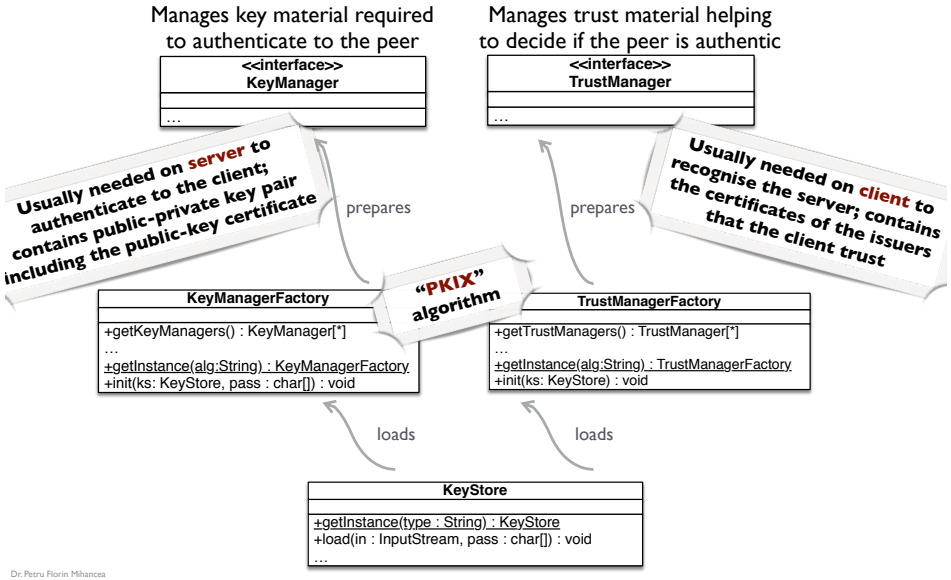
The provider

a particular implementation of the security subsystem e.g. "SunJSSE"
all registered providers can be found via Security.getProviders()

The protocol

specifies what protocol to be used e.g., "TLSv1.2"
see the *Standard Algorithm Name Documentation*

Key/Trust Managers



5

Demo Application

Secure our application that was based on TCP sockets. In general, other infrastructures can be configured to work over SSL/TLS.