Secure Client Applications

Networking

Sirindhorn International Institute of Technology
Thammasat University

Prepared by Steven Gordon on 26 June 2014
Common/Reports/secure-client-apps.tex, r900

Acronyms and Abbreviations

CA	Certificate Authority (same as TA)
HTTP	HyperText Transfer Protocol
HTTPS	HTTP over SSL
PGP	Pretty Good Privacy
PR	Private key
PU	Public key
SSL	Secure Sockets Layer (same as TLS)
TA	Trusted Authority (same as CA)
TCP	Transmission Control Protocol
TLS	Transport Layer Security (same as SSL)

Contents

Workshop Aims

- Understand security limitations of common Internet applications
- Increase awareness of "extensions" of Internet applications that increase security
- Learn about techniques for enhancing your communication secrecy and privacy

Applications and Extensions

Web Browsing

- Secrecy: HTTPS and certificates, HTTPS Everywhere
- Privacy: AdBlock Plus, Ghostery, FoxyProxy, Hola . . .
- Safety: NoScript, . . .

Email

- Signatures and Secrecy: OpenPGP, Enigmail, Thunderbird

File Encryption

- File:
- Disk: TrueCrypt, BitLocker

Contents

Cryptography

Symmetric Key Cryptography

- Source: Encrypt message with secret key K
- Destination: must also know K; decrypts data with K
- Pro: Fast for large amounts of data
- Con: Requires K to be securely exchanged in advance

Public Key Cryptography

- Each node has a (public, private) key pair, (PU_a, PR_a)
- Encrypt a message with one key in pair, can only be decrypted with other key in key pair
- Pro: Does not require exchange of secrets
- Con: Slow for large amounts of data

Public Key Cryptography

Public Key Cryptography for Confidentiality

- Source: Encrypt message with public key, PU_{dst} of destination
- Destination: decrypts data with PR_{dst}
- Only destination can decrypt it

Public Key Cryptography for Signatures

- Source: Encrypt message with own private key, PR_{src}
- Destination: decrypts data with PU_{src}
- Only source could have sent it

Contents

HTTP and HTTPS

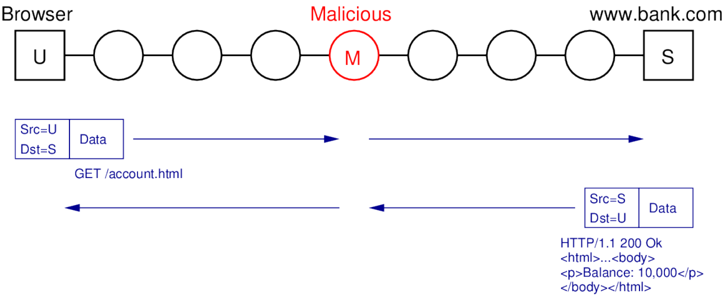
HTTP

- Send request to web server; returns the web page
- Malicious use can intercept/modify data

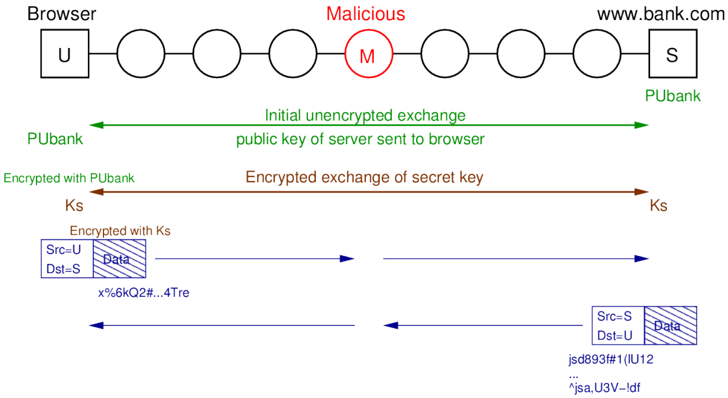
HTTPS

- Establish secure SSL/TLS connection between browser and server; then use HTTP
- Data is encrypted; interception/modification not possible
- But . . .

HTTP: Interception is Easy

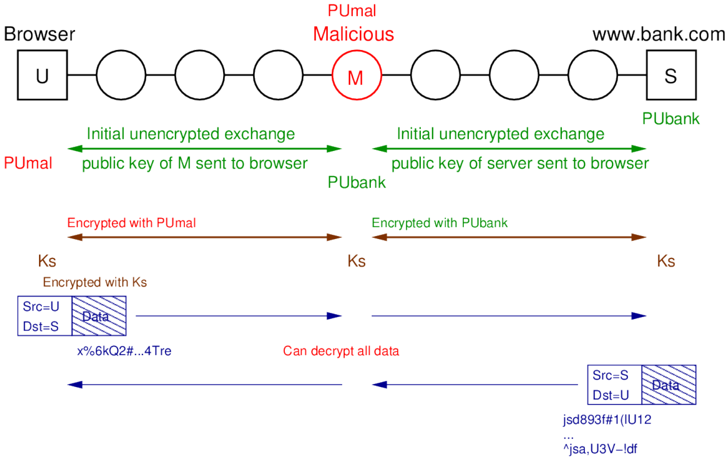


HTTPS: Data is Encrypted



- Public key cryptography used to exchange a secret key
- Data encrypted with secret key

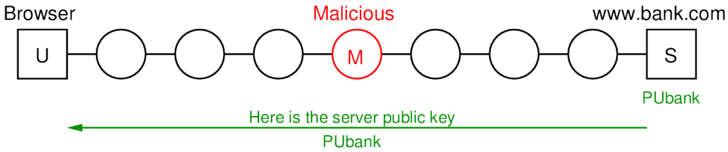
HTTPS: Man-in-the-Middle Attack



HTTPS Encryption

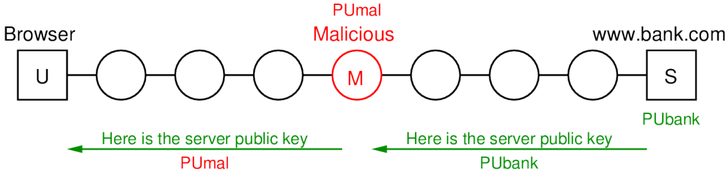
- To encrypt data, browser and server must exchange a secret key
- But cannot send secret key, unencrypted, across Internet
  - Use public-key cryptography for secret key exchange
- Server has (public, private) keypair
  - Encrypt with one, can only decrypt with the other in pair
- Server sends its public key to browser, then used to encrypt secret key

HTTPS: Challenge is Public Key Distribution



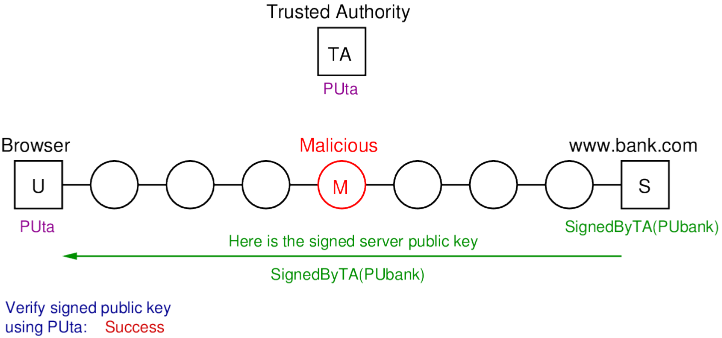
- How does browser know received public key is that of the server?

HTTPS: Challenge is Public Key Distribution



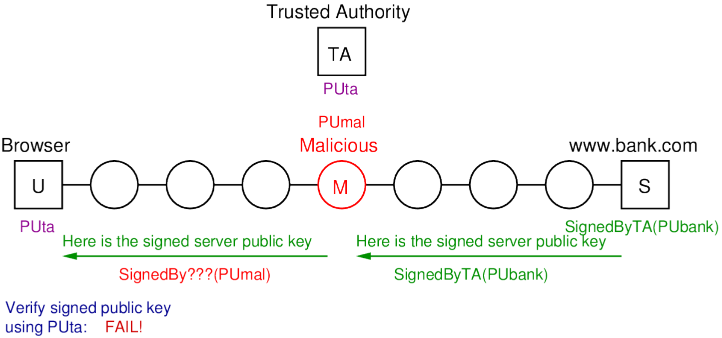
- How does browser know received public key is that of the server?

HTTPS: Trusted Authority Signs Key



- Trusted Authority: Another entity trusted by the browser and server
- Trusted Authority "signs" public key of server
- Browser "verifies" received public key using TA's public key

HTTPS: Trusted Authority Signs Key



- If malicious node modifies signed public key of server, the verification at browser will detect it
- A public key signed by someone else is called a digital certificate

Digital Certificates in Practice

How does a server obtain a certificate?

- Prove identity to CA by:
  - Domain validation
  - Extended validation
- Free and commercial services

How does browser obtain CA certificate?

- Pre-loaded into browsers
- Hierarchy of certificates is supported

What if CA certificate is not in browser?

- Browsers commonly present warning to user

Security Issues with Digital Certificates

- Identity verification of server (owners)
- Security of CA private key
- Pre-loaded certificates by browser publisher
- Response when invalid certificate received
- Algorithms used in certificates should be strong

Contents

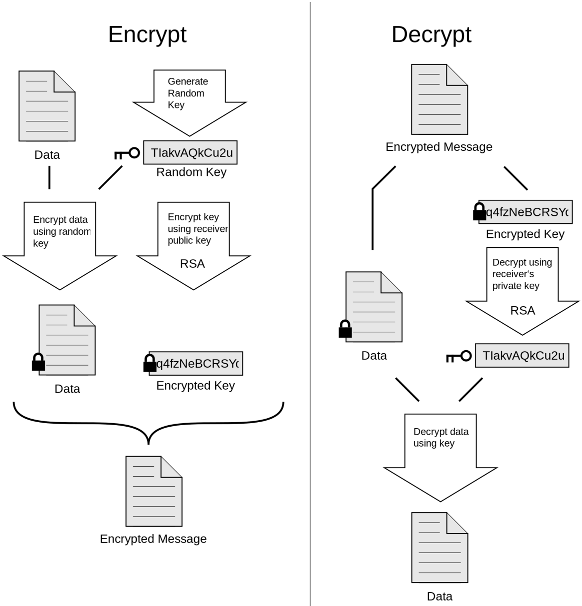
Secure Email

- Email messages originally only text with pre-defined headers (To, From Subject, CC, ...)
- Multipurpose Internet Mail Extensions (MIME) allows for different message and header formats: different character sets, attachments, new headers
- Secure email requirements:
  1. Authentication: receiver can confirm the actual sender, and that content is not modified
  2. Confidentiality: only sender/receiver can read the contents
- Two common ways to implement secure email:
  1. S/MIME
  2. OpenPGP
- Both use similar approach: sender signs message with private key, encrypts message with symmetric key encryption using a secret key, and encrypts the secret key using recipients public key

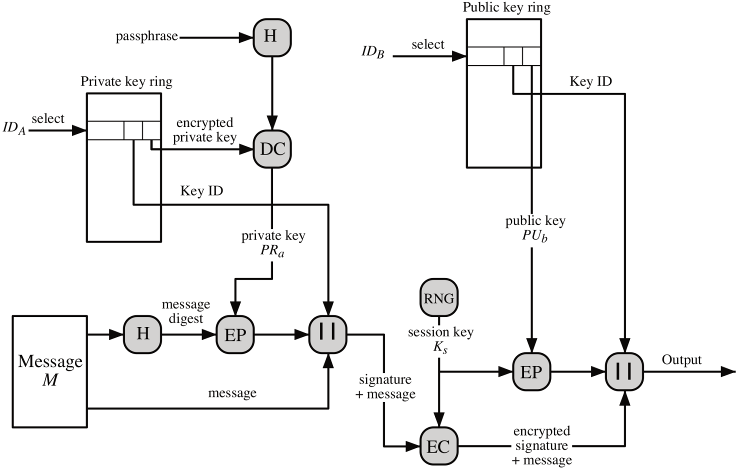
OpenPGP

- Pretty Good Privacy (PGP) developed by Phil Zimmerman in 1991
- IETF standardised as OpenPGP
- One of first and most widely used applications of public-key cryptography
- Implementations:
  - Original by Zimmerman: Symantec
  - GNU Privacy Guard (GPG)
  - Many email clients (either direct or through plugins, e.g. Enigmail, GPG4Win)
- OpenPGP vs S/MIME:
  - OpenPGP: public keys distributed informally: phone, websites, email
  - S/MIME: public keys distrubuted as X.509 digital certificates

PGP Operation: Concept

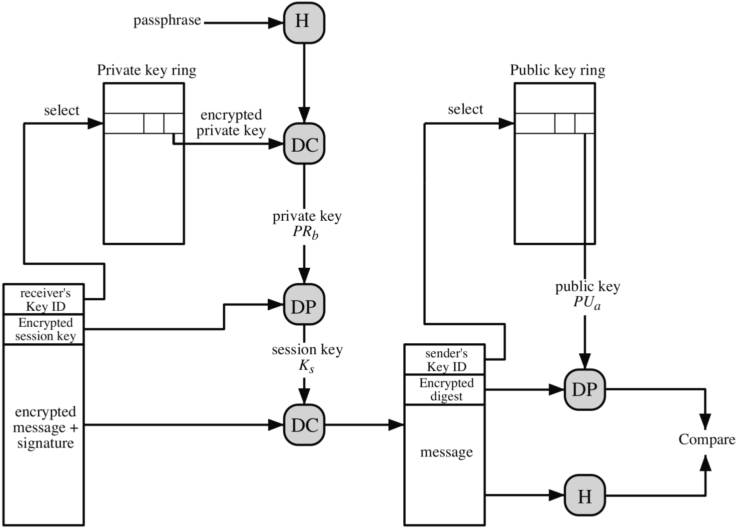


PGP Operation: Message Generation at A



Credit: Figure 18.5 in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

PGP Operation: Message Reception at B



Credit: Figure 18.6 in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011