15-441 15-641 Computer Networking

Lecture 21 – Security:
Key management
Peter Steenkiste

Fall 2016

www.cs.cmu.edu/~prs/15-441-F16

With slides from: Debabrata Dash, Nick Feamster, Vyas Sekar, and others

Outline – Creating a Secure Channel

- Security threats
- Cryptography overview
- Securing channels
- Key management
- TOR

One last "little detail"…

How do I get these keys in the first place??
Remember:

- Symmetric key primitives assumed Alice and Bob had already shared a key.
- Asymmetric key primitives assumed Alice knew Bob's public key.

This may work with friends, but when was the last time you saw Amazon.com walking down the street?

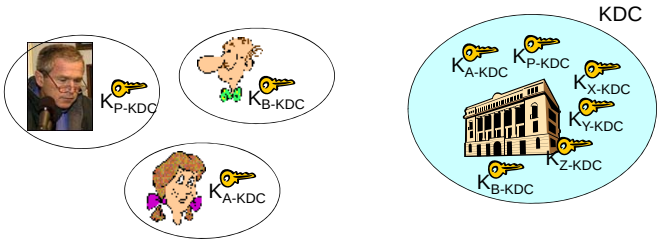
Symmetric Key Distribution

- How does Andrew do this?

Andrew Uses Kerberos, which relies on a Key Distribution Center (KDC) to establish shared symmetric keys.

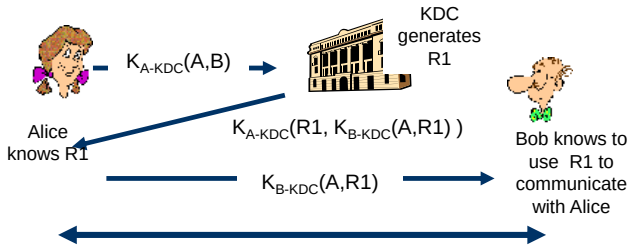
Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.



Key Distribution Center (KDC)

Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?



Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption

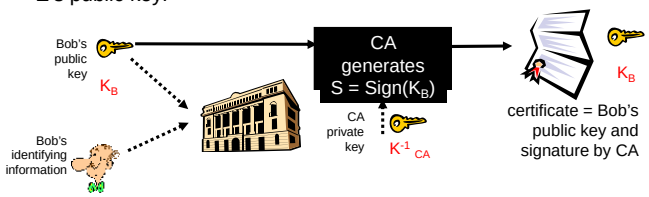
How Useful is a KDC?

- Must always be online to support secure communication
- KDC can expose our session keys to others!
- Centralized trust and point of failure.

In practice, the KDC model is mostly used within single organizations (e.g. Kerberos) but not more widely.

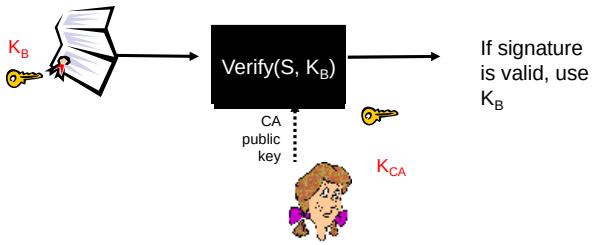
Certification Authorities: Distributing Public Keys

- Certification authority (CA): binds public key to particular entity, E.
- An entity E registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - Certificate contains E's public key AND the CA's signature of E's public key.



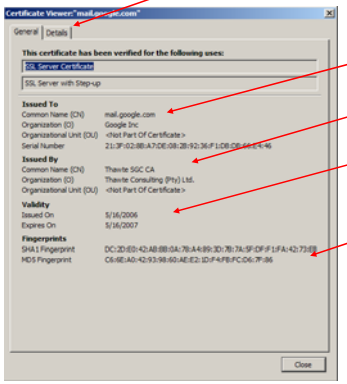
Certification Authorities

- When Alice wants Bob's public key:
  - Gets Bob's certificate (Bob or elsewhere).
  - Use CA's public key to verify the signature within Bob's certificate, then accepts public key



Certificate Contents

- info algorithm and key value itself (not shown)

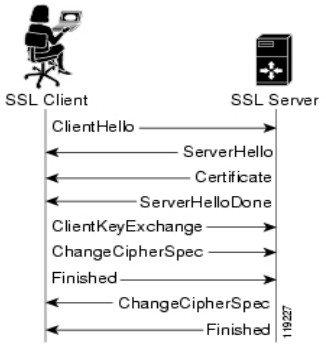


- Cert owner
- Cert issuer
- Valid dates
- Fingerprint of signature

Transport Layer Security (TLS) aka Secure Socket Layer (SSL)

- Used for protocols like HTTPS
- Special TLS socket layer between application and TCP (small changes to application).
- Handles confidentiality, integrity, and authentication.
- Uses "hybrid" cryptography.

Setup Channel with TLS "Handshake"



Handshake Steps:

1) Client and server negotiate exact cryptographic protocols
2) Client validates public key certificate with CA public key.
3) Client encrypts secret random value with server's key, and sends it as a challenge.
4) Server decrypts, proving it has the corresponding private key.
5) This value is used to derive symmetric session keys for encryption & MACs.

TLS Components

- Handshake authenticates endpoints and creates keys for confidentiality, integrity of data
  - Authentication based on certificates
    - Optional – typically client authenticates server only
  - Many cryptographic options for sessions key
  - Session resumption eliminates 1 RTT, reduces crypto
- Data exchange based on record protocol
  - Data stream broken up in chunks, which are encrypted
  - Encrypted chunk and MAC form a sequence of records

How TLS Handles Data

1) Data arrives as a stream from the application via the TLS Socket

2) The data is segmented by TLS into chunks

3) A session key is used to encrypt and MAC each chunk to form a TLS "record", which includes a short header and data that is encrypted, as well as a MAC.

4) Records form a byte stream that is fed to a TCP socket for transmission.

TLS Discussion

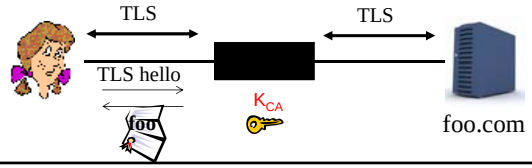
- The use of TLS is increasing – privacy concerns!
- TLS increases overhead on endpoints, network
  - Adds 1-2 RTTs to handshake; involves more packets
  - Overhead of key generation; mostly issue on server
  - Encryption overhead on server and client – minor
- TLS is very effective but has limitations: certificates and how they are handled
  - Compromised CAs
  - Users don't understand the technology
  - Must have "trusted" root certificates

Users don't Understand Certificates

- If the browser detects a problem with a certificate, it asks user what to do
  - Invalid, expired, self-signed, …
- Users often blindly click "yes"
  - They don't know about certificates or TLS; don't understand implications of a bad certificates
- Certificates are hard to read and can be misleading
  - Most information makes no sense to user
  - Names can be confusing, e.g., minor variants

Middleboxes: Good or Evil?

- Middleboxes are very widely used in the Internet
  - Companies have firewalls
  - Cellular operators use caches, compression, …
- But TLS makes middleboxes ineffective
- "Solution": install fake root certificate on device
  - Common for corporate networks
  - Sometimes also done by service providers



17

Tor Anonymity Network

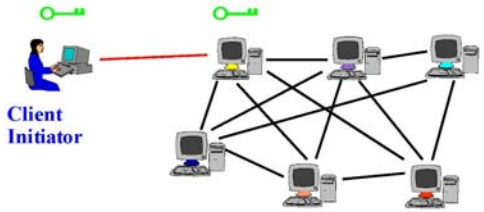
- Deployed onion routing network
  - http://torproject.org
  - Specifically designed for low-latency anonymous Internet communications
- Running since October 2003
  - Thousands of relay nodes, 100K-500K? of users
- Easy-to-use client proxy, integrated Web browser
  - Not like FreeNet – no data "in" TOR
- Really an overlay – not pure peer-to-peer

Based on slides by Vitaly Shmatikov

slide 18

Tor Circuit Setup (1)

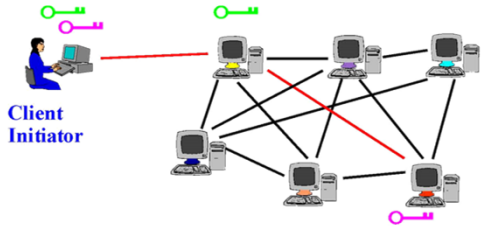
- Client proxy establish a symmetric session key and circuit with relay node #1
- All data sent over the circuit is encrypted $A = K(B)_k$



slide 19

Tor Circuit Setup (2)

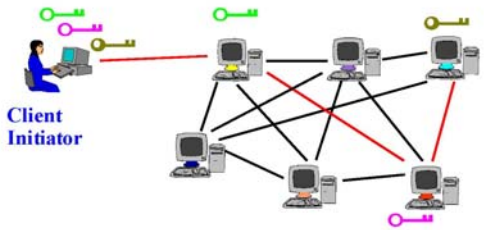
- Client proxy extends the circuit by establishing a symmetric session key with relay node #2
  - Tunnel through relay node #1
  - Relay #1 acts as source for packet to relay #2
    - Relay #2 must send packets to relay #1 on reverse path



slide 20

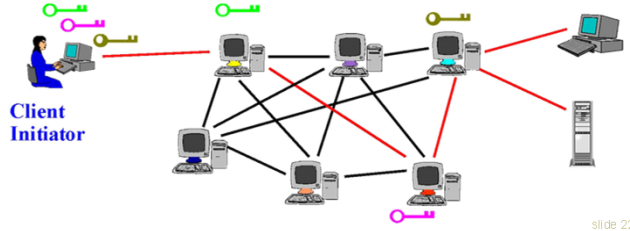
Tor Circuit Setup (3)

- Client proxy extends the circuit by establishing a symmetric session key with relay node #3
  - Tunnel through relay nodes #1 and #2
  - #2 acts as source for packets to #3



Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit
  - Datagrams decrypted at each link and forwarded to "previous" node in the Tor circuit
- Need end-to-end encryption – last hop in the clear



Using Tor

- Many applications can share one circuit
  - Multiple TCP streams over one anonymous connection
- Tor router doesn't need root privileges
  - Encourages people to set up their own routers
  - More participants = better anonymity for everyone
- Directory servers
  - Maintain lists of active relay nodes, their locations, current public keys, etc.
  - Control how new nodes join the network
    - "Sybil attack": attacker creates a large number of relays
  - Directory servers' keys ship with Tor code

Summary – Part I

- Internet design and growth => security challenges
- Symmetric (pre-shared key, fast) and asymmetric (key pairs, slow) primitives provide:
  - Confidentiality
  - Integrity
  - Authentication
- "Hybrid Encryption" leverages strengths of both.
- Great complexity exists in securely acquiring keys.
- Crypto is hard to get right, so use tools from others, don't design your own (e.g. TLS).

Resources

- Textbook: 8.1 – 8.3
- Wikipedia for overview of Symmetric/Asymmetric primitives and Hash functions.
- OpenSSL (www.openssl.org): top-rate open source code for SSL and primitive functions.
- “Handbook of Applied Cryptography” available free online: www.cacr.math.uwaterloo.ca/hac/

25