15-446 Distributed Systems
Spring 2009

L-13 Security

Schedule up to Midterm

- 2/26 No class (work on project 1, hw3)
- Review 3/2 Monday 4:30 pm NSH 3002
- HW 3 due
- Midterm 3/3 Tuesday in class

Project

- Problem in coping files
  - Files are not deleted at every new run
  - Older files are copied into SD card
- Will fix (release a new ruby server)

Important Lessons - CDNs

- Akamai CDN → illustrate range of ideas
  - BASE (not ACID design)
  - Weak consistency
  - Naming of objects → location translation
  - Consistent hashing
- Why are these the right design choices for this application?
Today's Lecture

- Internet security weaknesses
- Establishing secure channels (Crypto 101)
- Key distribution

What is “Internet Security”?

- Denial-of-Service
- Worms & Viruses
- Traffic modification
- Trojan Horse
- DNS Poisoning
- Phishing
- Spyware
- IP Spoofing
- End-host impersonation
- Route Hijacks
- Traffic modification
- End-host impersonation
- Spam

Internet Design Decisions: (ie: how did we get here? )

- Origin as a small and cooperative network (largely trusted infrastructure)
- Global Addressing (every sociopath is your next-door neighbor*)
- Connection-less datagram service (can't verify source, hard to protect bandwidth)

Internet Design Decisions: (ie: how did we get here? )

- Anyone can connect (ANYONE can connect)
- Millions of hosts run nearly identical software (single exploit can create epidemic)
- Most Internet users know about as much as Senator Stevens aka “the tubes guy” (God help us all...)

* Dan Geer
Our “Narrow” Focus

Yes:
1) Creating a “secure channel” for communication (today)
2) Protecting resources and limiting connectivity (after exam)

No:
1) Preventing software vulnerabilities & malware, or “social engineering”.

Secure Communication with an Untrusted Infrastructure

ISP A
ISP B
ISP C
ISP D

Alice
Bob
Mallory

Hello, I’m “Bob”
What do we need for a secure communication channel?

- Authentication (Who am I talking to?)
- Confidentiality (Is my data hidden?)
- Integrity (Has my data been modified?)
- Availability (Can I reach the destination?)

Today's Lecture

- Internet security weaknesses
- Crypto 101
- Key distribution

What is cryptography?

"cryptography is about communication in the presence of adversaries."
- Ron Rivest

What is cryptography?

Tools to help us build secure communication channels that provide:

1) Authentication
2) Integrity
3) Confidentiality
**Cryptography As a Tool**

- Using cryptography securely is not simple
- Designing cryptographic schemes correctly is near impossible.

Today we want to give you an idea of what can be done with cryptography. Take a security course if you think you may use it in the future (e.g. 18-487)

---

**The Great Divide**

<table>
<thead>
<tr>
<th>Symmetric Crypto: (Private key)</th>
<th>Asymmetric Crypto: (Public key)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: AES</td>
<td>Example: RSA</td>
</tr>
<tr>
<td>Requires a pre-shared secret between communicating parties?</td>
<td>Yes</td>
</tr>
<tr>
<td>Overall speed of cryptographic operations</td>
<td>Fast</td>
</tr>
</tbody>
</table>

---

**Symmetric Key: Confidentiality**

**Motivating Example:**
You and a friend share a key $K$ of $L$ random bits, and a message $M$ also $L$ bits long.

**Scheme:**
You send her the $xor(M,K)$ and then they "decrypt" using $xor(M,K)$ again.

1) Do you get the right message to your friend?
2) Can an adversary recover the message $M$?

---

**Symmetric Key: Confidentiality**

- One-time Pad (OTP) is secure but usually impractical
  - Key is as long at the message
  - Keys cannot be reused (why?)

In practice, two types of ciphers are used that require only constant key length:

**Stream Ciphers:**
Ex: RC4, A5

**Block Ciphers:**
Ex: DES, AES, Blowfish
Symmetric Key: Confidentiality

- Stream Ciphers (ex: RC4)

Alice: $K_{A-B}$ → PRNG → Pseudo-Random stream of L bits → XOR → Message of Length L bits → Encrypted Ciphertext

Bob uses $K_{A-B}$ as PRNG seed, and XORs encrypted text to get the message back (just like OTP).

Symmetric Key: Integrity

- Background: Hash Function Properties
  - Consistent: hash(X) always yields same result
  - One-way: given X, can’t find Y s.t. hash(Y) = X
  - Collision resistant: given hash(W) = Z, can’t find X such that hash(X) = Z

Message of arbitrary length → Hash Fn → Fixed Size Hash

Symmetric Key: Confidentiality

- Block Ciphers (ex: AES)

Bob breaks the ciphertext into blocks, feeds it through decryption engine using $K_{A-B}$ to recover the message.

Symmetric Key: Integrity

- Hash Message Authentication Code (HMAC)

Step #1: Alice creates MAC

Step #2: Alice Transmits Message & MAC

Step #3: Bob computes MAC with message and $K_{A-B}$ to verify.

Why is this secure? How do properties of a hash function help us?
**Symmetric Key: Authentication**

- You already know how to do this! (hint: think about how we showed integrity)

  ![Diagram](image1.png)

  - I am Bob
  - Hash Fn
  - A43FF234
  - Alice receives the hash, computes a hash with $K_{A-B}$, and she knows a hash with $K_{A-B}$
  - **Wrong!**

---

**Symmetric Key: Authentication**

- A “Nonce”
  - A random bitstring used only once. Alice sends nonce to Bob as a “challenge”. Bob replies with “fresh” MAC result.

  ![Diagram](image2.png)

  - Alice
  - Bob
  - Nonce
  - Hash
  - B4FE6d
  - Performs same hash with $K_{A-B}$ and compares results

---

**Symmetric Key: Authentication**

- A “Nonce”
  - A random bitstring used only once. Alice sends nonce to Bob as a “challenge”. Bob replies with “fresh” MAC result.

  ![Diagram](image3.png)

  - Alice
  - Bob
  - Mallory
  - ?!?!?
  - If Alice sends Mallory a nonce, she cannot compute the corresponding MAC without $K_{A-B}$

---

**Symmetric Key: Authentication**

- What is Mallory overhears the hash sent by Bob, and then “replays” it later?

  ![Diagram](image4.png)

  - ISP A
  - ISP B
  - ISP C
  - ISP D
  - Hello, I’m Bob. Here’s the hash to “prove” it
  - A43FF234
  - Mallory
Symmetric Key Crypto Review

- Confidentiality: Stream & Block Ciphers
- Integrity: HMAC
- Authentication: HMAC and Nonce

Questions??
Are we done? Not Really:
1) Number of keys scales as $O(n^2)$
2) How to securely share keys in the first place?

Asymmetric Key Crypto:

- Instead of shared keys, each person has a “key pair”
  
  - $K_B$ Bob’s public key
  - $K_B^{-1}$ Bob’s private key

- The keys are inverses, $K_B^{-1}(K_B(m)) = m$

Asymmetric Key: Confidentiality

- It is believed to be computationally unfeasible to derive $K_B^{-1}$ from $K_B$ or to find any way to get $M$ from $K_B(M)$ other than using $K_B^{-1}$.

=> $K_B$ can safely be made public.

Note: We will not detail the computation that $K_B(m)$ entails, but rather treat these functions as black boxes with the desired properties.
Asymmetric Key: Sign & Verify

- If we are given a message M, and a value S such that \( K_B(S) = M \), what can we conclude?
  - The message must be from Bob, because it must be the case that \( S = K_B^{-1}(M) \), and only Bob has \( K_B^{-1} \).
- This gives us two primitives:
  - Sign (M) = \( K_B^{-1}(M) \) = Signature S
  - Verify (S, M) = test( \( K_B(S) == M \) )

Asymmetric Key: Integrity & Authentication

- We can use Sign() and Verify() in a similar manner as our HMAC in symmetric schemes.
  - Integrity: \( S = \text{Sign}(M) \)
    - Message M
    - Receiver must only check Verify(M, S)
  - Authentication: \( S = \text{Sign}(\text{Nonce}) \)
    - Nonce
    - Verify(Nonce, S)

Asymmetric Key Review:

- Confidentiality: Encrypt with Public Key of Receiver
- Integrity: Sign message with private key of the sender
- Authentication: Entity being authenticated signs a nonce with private key, signature is then verified with the public key

But, these operations are computationally expensive*

Today's Lecture

- Internet security weaknesses
- Crypto 101
- Key distribution
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, $K_{A-KDC}$, $K_{B-KDC}$, for communicating with KDC.

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

KDC generates $R_1$

$K_{A-KDC}(A,B)$

$K_{A-KDC}(R_1, K_{B-KDC}(A,R_1))$

Bob knows to use $R_1$ to communicate with Alice

Alice and Bob communicate: using $R_1$ 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.

The Dreaded PKI

- Definition:
  Public Key Infrastructure (PKI)
  1) A system in which “roots of trust” authoritatively bind public keys to real-world identities
  2) A significant stumbling block in deploying many “next generation” secure Internet protocol or applications.

Certification Authorities

- **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.

**Bob’s public key**  
**Bob’s identifying information**  
**CA private key**  
**K^{-1}_{CA}**  
**CA generates**  
**S = Sign(K_B)**  
**Certificate = Bob’s public key and signature by CA**  

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

**Verif(S, K_B)**  
**If signature is valid, use K_B**  

**CA public key**  
**K_{CA}**
**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. Clients and servers negotiate exact cryptographic protocols
2. Client's validate public key certificate with CA public key.
3. Client encrypt secret random value with server's key, and send 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.

**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.
Important Lessons

- 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

- 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/