Reading Quiz
TO FIX THIS PROBLEM WE NEED . . .

BLACKBOX #5: PUBLIC KEY CRYPTOGRAPHY
Shortcomings of symmetric key

Establishing a pairwise key requires a **key exchange**, which requires both parties to be **online**

**Issue #1: Requires pairwise key exchanges**

**File downloads**

One-to-many: $O(N)$ key exchanges

**Email / chat**

All-to-all: $O(N^2)$ key exchanges
Shortcomings of symmetric key

Establishing a pairwise key requires a **key exchange**, which requires both parties to be **online**.

**Issue #3: How do you know to whom you’re talking?**

Diffie-Hellman is resilient to **eavesdropping**, but **not tampering**.
Trusted Third Party

A protocol that solves this with trust

Trent: A trusted third party

Alice → Trent → Bob

Bob → Trent → Alice
Trusted Third Party

A protocol that solves this with **trust**

**Trent**: A *trusted* third party

E($K_{AT}$, msg || to:Bob)

1. Everybody establishes a pairwise key with Trent
   - **Good**: $O(N)$ key exchanges

2. Trent validates each user’s identity; includes in message
   - **Good**: Authenticated communication
Trusted Third Party

A protocol that solves this with trust

**Trent:** A trusted third party

\[ E(K_{AT}, \text{msg } || \text{to:Bob}) \]
\[ E(K_{BT}, \text{msg } || \text{from:Alice}) \]

1. Everybody establishes a pairwise key with Trent
   **Good:** \( O(N) \) key exchanges

2. Trent validates each user’s identity; includes in message
   **Good:** Authenticated communication

**Bad:** All messages get sent through Trent
What are we trusting Trent not to do?

Just as “secure” meant nothing without an attack model, “trusted” means nothing without a trust model.

\[ E(K_{AT}, \text{msg} \parallel \text{to:Bob}) \]

**1. Do not read messages**
**2. Do not alter messages**
**3. Do not forge messages**
**4. Do not go offline**
Hybrid encryption

1. Generate public/private key pair (PK, SK); publicize PK
2. Obtain PK
3. Generate symmetric key K
4. Compute $c_{msg} = e(K, \text{msg})$
5. Compute $c_K = E(PK, K)$
6. Send $c_K || c_{msg}$
7. Decrypt $D(SK, c_K) = K$
8. Decrypt $d(K, c_{msg}) = \text{msg}$
Hybrid encryption

Obtain PK
Generate symmetric key $K$
Compute $c_{msg} = e(K, \text{msg})$
Compute $c_K = E(PK, K)$
Send $c_K || c_{msg}$

The easy key distribution of public key

The speed and arbitrary message length of symmetric key
Protocols with public key cryptography
Goal: determine from whom a message came

**Symmetric key**

*File downloads*

One-to-many:
O(N) key exchanges
Digital signatures

A digital signature scheme comprises two algorithms

**Signing function** \( \text{Sgn}(SK, m) \)
- Inputs
  - **Secret** key \( SK \)
  - Fixed-length message
- Outputs: a **signature** \( s \)

**Verification function** \( \text{Vfy}(PK, m, s) \)
- Inputs
  - **Public** key \( PK \)
  - Message and signature
- Outputs: Yes/No if valid \( (m, s) \)

This is a *randomized* algorithm (nondeterministic output)

**SK** a.k.a. “Signing key”

Only one person can sign with a given \( (PK, SK) \) pair

Deterministic algorithm

Anyone with the PK can verify

Bob generates a public key (also known as a **verification** key) and a private key (also known as a **signing** key).
Digital signatures

A digital signature scheme comprises two algorithms

**Signing** $\text{Sgn}(SK, m)$ → a signature $s$

**Verification** $\text{Vfy}(PK, m, s)$ → Yes/No if valid $(m,s)$

**Correctness**
$\text{Vfy}(PK, m, \text{Sgn}(SK, m)) = \text{Yes}$

**Security**
Same as with MACs: even after a chosen plaintext attack, the attacker cannot demonstrate an existential forgery
Protocols with digital signatures
Goal: determine from whom a message came

**Symmetric key**

*File downloads*

One-to-many:
*O(N) key exchanges*

- Generate public/private key pair (PK, SK)
- Announce PK publicly (on website, in newspaper, ...)
- Compute sig = Sgn(SK, msg)
- Publish msg || sig

*can now go offline!*
Protocols with digital signatures
Goal: determine from whom a message came

**Symmetric key**

*File downloads*

Generate public/private key pair (PK, SK)
Announce PK publicly (on website, in newspaper, ...)

Compute $\text{sig} = \text{Sgn}(SK, \text{msg})$

Publish $\text{msg} \parallel \text{sig}$

*can now go offline!*

One-to-many: $O(N)$ key exchanges
Digital signature properties

**Symmetric key**

File downloads

One-to-many: O(N) key exchanges

Generate public/private key pair (PK, SK)

Announce PK publicly (on website, in newspaper, …)

Compute sig = Sgn(SK, msg)

Publish msg || sig

*can now go offline!*

**Options**

A. Authenticity
B. Integrity
C. Confidentiality
D. No—repudiation
E. Availability
The two communicating parties thought, *but did not confirm*, that they were talking to one another.

Therefore, they were vulnerable to MITM attacks.

Certificates allow us to verify with whom we are communicating.

We will solve this by incorporating public key cryptography
Back to authentication

Generate public/private key pair (PK, SK); publicize PK

How can we know it was really \( \text{who posted PK?} \)

\[
E(K_{AT}, \text{msg } || \text{to: Bob}) \quad \text{E}(K_{BT}, \text{msg } || \text{from: Alice})
\]

Can we achieve authentication without Trent in the middle of every message?
Authentication with public keys

1. Trent's public key is widely disseminated (pre-installed in browsers/operating systems)

2. Alice generates a public/private key pair and asks Trent to bind her $\text{PK}_A$ to her identity

3. Trent signs a message (with $\text{SK}_T$):
   
   “The owner of the secret key corresponding to $\text{PK}_A$ is Alice”

   This message + sig = Certificate
Authentication with public keys

1. Trent's public key is widely disseminated (pre-installed in browsers/operating systems)

2. Alice generates a public/private key pair and asks Trent to bind her $PK_A$ to her identity

3. Trent signs a message (with $SK_T$):
   "The owner of the secret key corresponding to $PK_A$ is Alice"

This message + sig = Certificate
Authentication with public keys

4. Alice makes her certificate publicly available (or Bob simply asks for it)

5. Bob verifies the certificate using $PK_T$
   
   If Bob trusts Trent, then Bob trusts that he properly vetted Alice, and thus that her public key is $PK_A$

6. Bob (via hybrid encryption) sends a message to Alice using her public key $PK_A$
Authentication with public keys

Properties

Trent need be online only when giving out certificates, not any time users want to communicate with one another.

Alice and Bob can communicate in an authenticated manner without having to go through Trent.
Authentication with public keys

Trust assumptions from our symmetric key protocol:

1. Do not read messages
2. Do not alter messages
3. Do not forge messages
4. Do not go offline

Trust assumptions in this public key protocol:

1. Correctly vet users
   (Some more in practice...)

Trent (PKₜ, SKₜ) 
Trent vets Alice 

Alice (PKₐ, SKₐ) 
Alice = PKₐ

Bob PKₜ 
Alice = PKₐ
Authentication with public keys

A. Alice is doomed 😞
B. Trent is doomed :( 
C. Keep Calm there’s a fix (discuss possible solutions)
D. Keep Calm there are multiple solutions!

What happens if Alice’s key gets compromised?
(Stolen, accidentally revealed, …)
Certificate revocation

3. Trent *signs* a message (with $SK_T$):

“The owner of the secret key corresponding to $PK_A$ is Alice”

This message + sig = Certificate

Put another way:

“The only person who knows $SK_A$ is Alice”

What happens if Alice’s key gets compromised?
(Stolen, accidentally revealed, …)
Certificate revocation

Trent (PK_T, SK_T)

Please revoke my certificate (ID #3912…)

Trent signs a message (with SK_T):

“Certificate ID #3912… is no longer valid, as of April 5, …”

Alice
Certificate revocation

Please revoke my certificate (ID #3912...)

Trent signs a message (with \(SK_T\)):

“Certificate ID #3912... is no longer valid, as of April 5, ...”

This message + sig = revocation

Bob obtains revocation information
Obtaining revocation data

Certificate Revocation Lists (CRLs)

A (often large) signed list of revocations

“Certificate ID #3912… is no longer valid, as of April 5, …”

Browsers and OSes occasionally download CRLs

Disincentive: CRLs can be large, so it takes time & bandwidth

Result: delayed days/weeks/forever
Obtaining revocation data

Online Certificate Status Protocol (OCSP)

Browsers and OSes perform OCSP checks on-demand (when verifying the certificate)

Is certificate ID #3912… still valid?

“Certificate ID #3912… is still longer valid, as of April 5, …”

Disincentive: Still delays the initial validation of the certificate (can increase webpage load time)
Obtaining revocation data

OCSP Stapling

Websites issue OCSP requests, include responses in initial handshake

Is certificate ID #3912… still valid?

“Certificate ID #3912… is still longer valid, as of April 5, …”

Alice forwards this to Bob along with the certificate when they first start to communicate
Certificate revocation responsibilities

Alice’s responsibility:

Trent’s responsibility:

Bob’s responsibility:

Map the responsibility to the party (Alice, Bob, Trent)
A. Request revocations
B. Make revocations publicly available
C. Check for revocations
Certificates in the wild

The lock icon indicates that the browser was able to authenticate the other end, i.e., validate its certificate.
Certificate chain

**Subject** (who owns the public key)

**Common name:** the URL of the subject

**Issuer** (who verified the identity and signed this certificate)
Network Security!
What is the goal of a network?

• Allow devices communicate with one another and coordinate their actions to work together.

• Piece of cake, right?
A “Simple” Task

Send information from one computer to another
A “Simple” Task

Send information from one computer to another

- hosts: endpoints of a network
- The plumbing is called a link.
A “Simple” Task: Sending a message from host to destination

But first... let’s try the postal system, something we are all (still!) familiar with and address a couple of key challenges..
A “Simple” analogous task: Post-it Note

Alice and Mila are Swatties starting out their semester and are roommates. Alice wants to give Mila a reminder to get milk.

Alice

Message

Mila

Transport Link
A “Simple” analogous task: Post-it Note

Alice and Mila are roommates, Alice wants to give Mila a reminder to get milk. Figure out some key tasks:

1. **Structure of the message:**
   - Construct the message that Alice posts to Mila.

2. **Organizing a drop-off point.**
   - Who chooses the drop-off point?

3. **Write a protocol to write a note /post—it to your housemate**
A “Simple” analogous task: Post-it Note

Alice and Mila are roommates, Alice wants to give Mila a reminder to get milk.

1. **Structure of the message: (Alice to Mila)**

<table>
<thead>
<tr>
<th>To Mila, From Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Don’t forget the milk!</td>
</tr>
</tbody>
</table>

Irrespective of the source and destination, the format of the message stays the same.
A “Simple” analogous task: Post-it Note

Alice and Mila are roommates, Alice wants to give Mila a reminder to get milk.

1. Structure of the message: (Alice to Mila)

<table>
<thead>
<tr>
<th>To Mila, From Alice</th>
<th>Header: contains sender/receiver information + additional state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- timestamp</td>
</tr>
<tr>
<td></td>
<td>- urgent! (priority)</td>
</tr>
<tr>
<td></td>
<td>- ordering of messages (1 of 10..)</td>
</tr>
<tr>
<td></td>
<td>- error control.</td>
</tr>
</tbody>
</table>

Don’t forget the milk!

Irrespective of the source and destination, the format of the message stays the same.
• **Message**: Header + Data
• **Data**: what sender wants the receiver to know
• **Header**: information to support protocol
  • Source and destination addresses
  • State of protocol operation
  • Error control (to check integrity of received data)
What is a protocol?

Protocol: message format + transfer procedure

Human Protocol

- Alice: Hi!
- Mila: Hi!
- Alice: Got the time?
- Alice: 2:00

Network Protocols (defined in RFCs)

- GET http://www.cs.swarthmore.edu
- <file>
What is a protocol?

Goal: get message from sender to receiver
Protocol: message format + transfer procedure

• Expectations of operation
  • first you do $x$, then I do $y$, then you do $z$, ...

• Multiparty! so no central control
  • sender and receiver are separate processes
Message Encapsulation

- Higher layer within lower layer
- Each layer has different concerns, provides abstract services to those above
A “Simple” analogous task: Postal Mail

• Many more considerations..
  • Who decides the the sender and receiver addresses? Does someone maintain a mapping peoples’ names to addresses?
  • Can Mila always be guaranteed of this delivery date? What factors influence delivery?
  • What if the mail gets lost – who’s responsibility is it? Alice, Mila or someone else?
  • What about security? privacy?
A “Simple” Task

Send information from one computer to another

- hosts: endpoints of a network
- The plumbing is called a link.
Not Really So Simple…
Not Really So Simple…

Diagram showing connections between AT&T, Quest, Sprint, Cogent, Swat, and Google.
Not Really So Simple…
Not Really So Simple…
We only need...

- Manage complexity and scale up
- Naming and addressing
- Moving data to the destination
- Reliability and fault tolerance
- Resource allocation, Security, Privacy..
## Five-Layer Internet Model

<table>
<thead>
<tr>
<th>Layer</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>the application (e.g., the Web, Email)</td>
</tr>
<tr>
<td>Transport</td>
<td>end-to-end connections, reliability</td>
</tr>
<tr>
<td>Network</td>
<td>routing</td>
</tr>
<tr>
<td>Link (data-link)</td>
<td>framing, error detection</td>
</tr>
<tr>
<td>Physical</td>
<td>1’s and 0’s/bits across a medium (copper, the air, fiber)</td>
</tr>
</tbody>
</table>
Internet Protocol Suite

The Hourglass Model

FTP, HTTP, VoIP, SSH

TCP, UDP

IP

NET_1, NET_2, ..., NET_n

Applications

Transport

Data Link

Physical

“Thin Waist”
Message Encapsulation

- Higher layer within lower layer
- Each layer has different concerns, provides abstract services to those above
Layering and encapsulation

- **Layering**:
  - Application
  - Transport: reliability
  - Network: routing
  - Link: framing, error detection

- **Protocols**:
  - **TCP/UDP**
  - **IP**
  - **Ethernet**

- **Addresses**:
  - Port no.
  - IP address
  - MAC address
Layering: Separation of Functions

• explicit structure allows identification, relationship of complex system’s pieces
  • layered reference model for discussion
  • reusable component design

• modularization eases maintenance
  • change of implementation of layer’s service transparent to rest of system,
  • e.g., change in postal route doesn’t effect delivery of letter
Abstraction!

- Hides the complex details of a process

- Use abstract representation of relevant properties make reasoning simpler

- Ex: Alice and Mila’s knowledge of postal system:
  - Letters with addresses go in, come out other side
TCP/IP Protocol Stack

Slide 65
TCP/IP Protocol Stack
The “End-to-End” Argument

Don’t provide a function at lower layer if you have to do it at higher layer anyway ...

... unless there is a very good performance reason to do so.

Examples: error control, quality of service

Midterm Discussion