CS 458 / 658: Computer Security and Privacy

* - Review of Course Content

- - -

Meng Xu *(University of Waterloo)*

Winter 2022

Outline

What is security?

In the context of computers, security generally means three things:

- **C**onfidentiality
  - Access to systems or data is limited to authorized parties
- **I**ntegrity
  - When you receive data, you get the "right" data
- **A**vailability
  - The system or data is there when you want it

What is privacy?

There are many definitions of privacy. A useful definition can be: "informational self-determination"

- This means that you get to control information about you
- "Control" means many things:
  - Who gets to see it
  - Who gets to use it
  - What they can use it for
  - Who they can give it to
  - etc.

Some terminology

- Vulnerabilities
- Threats
  1. Interception
  2. Interruption
  3. Modification
  4. Fabrication
- Attacks
- Controls

One sentence to chain them together: You control a vulnerability to prevent an attack and defend against a threat.

Outline

Types of unintentional flaws

- Buffer overflows
- Integer overflows
- Format string vulnerabilities
- Incomplete mediation
- TOCTTOU erros

What does the memory layout of a process look like?

- Program code (Text)
- Global data (BSS and data segments)
- Dynamically alllocated data (Heap)
- Function call data (Stack)

Q: What happens in stack during a function call?

Function calls

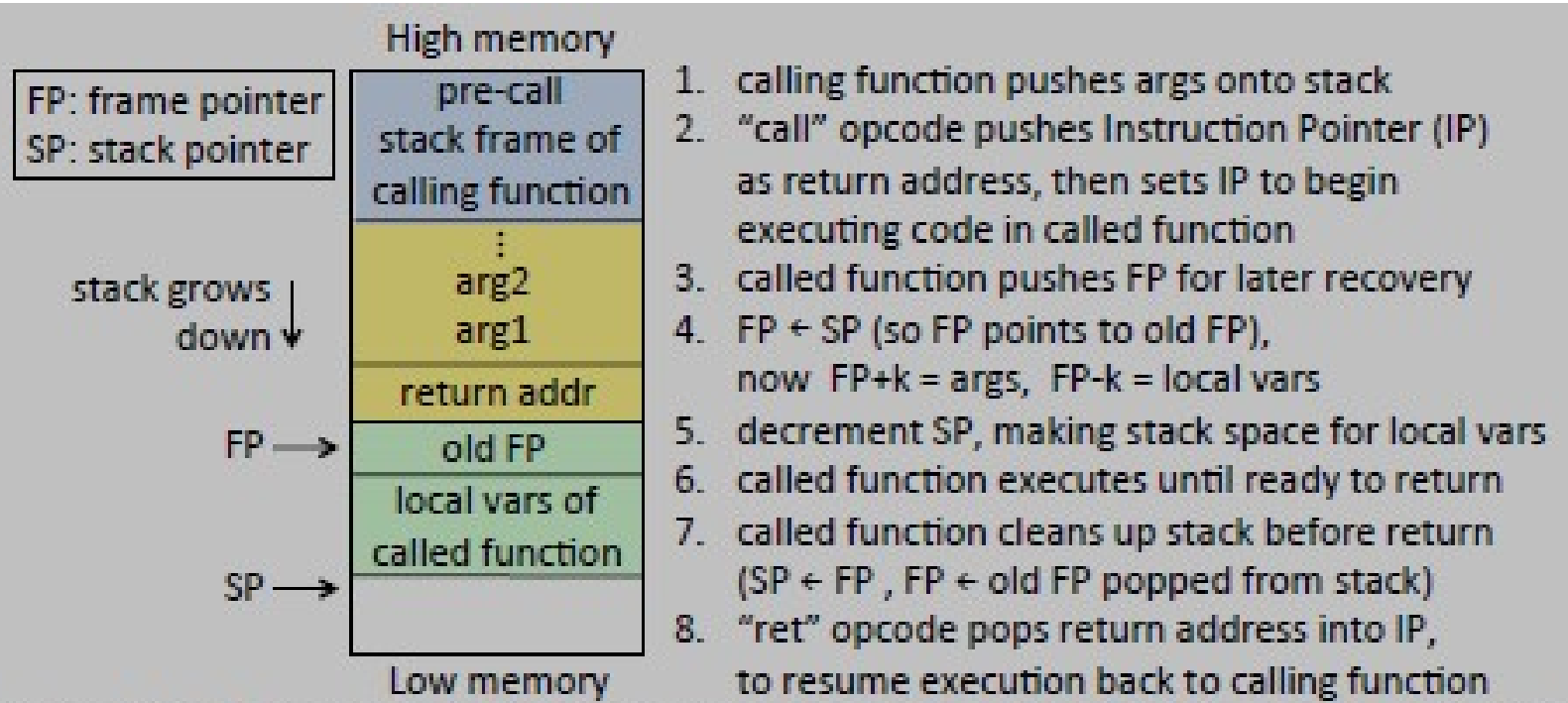


Figure 6.4: User-space stack and function call sequence (x86 conventions). FP is also called BP (Base Pointer). Register state may also be saved onto the stack (not shown).

(Source: van Oorschot textbook, Chapter 6, https://people.scs.carleton.ca/~paulv/toolsjewels.html)

Buffer overflows

The single most commonly exploited type of security flaw

Example:

```
#define LINELEN 1024
char buffer[LINELEN];

gets(buffer);
```

or

```
strcpy(buffer, argv[1]);
```

Important reading: Smashing The Stack For Fun And Profit

Defences against buffer overflows

- Programmer: Use a language with bounds checking
- Compiler: Place padding between data and return address ("Canaries")
  - Detect if the stack has been overwritten before the return from each function
- Memory: Non-executable stack
  - "W⊕X", DEP (memory page is either writable or executable, but never both)
- OS: Stack (and sometimes code,heap,libraries) at random virtual addresses for each process
  - Address Space Layout Randomization (ASLR) - All mainstream OSes do this
- Hardware-assistance: pointer authentication, shadow stack, memory tagging

Integer overflows

Root cause: Machine integers can represent only a limited set of numbers, might not correspond to programmer's mental model

Example: If the programmer assumes that integer is always positive, overflow will make (signed) integer wrap and become negative, which will violate the assumption

Format string vulnerabilities

Unfiltered user input is used as format string in `printf()`, `fprintf()`, `sprintf()`, ...

For example:

- `printf(buffer)` instead of `printf("%s", buffer)`
  - The first one will parse `buffer` for `%s` and use whatever is currently on the stack to process found format parameters
- `printf("%s%s%s%s")` likely crashes your program
- `printf("%x%x%x%x")` dumps parts of the stack
- `%n` will write to an address found on the stack

Incomplete mediation

- Inputs to programs are often specified by untrusted users
- Users sometimes mistype data in web forms
- The web application needs to ensure that what the user has entered constitutes a meaningful request $\rightarrow$ mediation

- Incomplete mediation occurs when the application accepts incorrect data from the user
- We focus on catching entries that are clearly wrong
  - Not well formed (e.g., DOB: `1980-04-31`)
  - Unreasonable values (e.g., DOB: `1876-10-12`)
  - Inconsistent with other entries
- We need this to prevent
  - SQL injections
  - Cross-Site Scripting (XSS) attacks

TOCTTOU errors

Time-Of-Check To Time-Of-Use errors, also known as "race condition" errors

These errors may occur when the following happens:

1. User requests the system to perform an action
2. The system verifies the user is allowed to perform the action
3. The system performs the action

Q: What happens if the state of the system changes between steps 2 and 3?

Race condition example

- A particular Unix terminal program is `setuid` (runs with superuser privileges)
- It supports a command to dump the terminal contents to a log file
- It first checks if the user has permissions to write to the requested file; if so, it opens the file for writing
- The attacker makes a symbolic link:
  `logfile -> file_she_owns`
- Between the "check" and the "open", she changes it:
  `logfile -> /etc/passwd`

Types of malware

- Virus
  - Malicious code that adds itself to benign programs/files
  - Code for spreading + code for actual attack
  - Usually activated by users
- Worms
  - Self-contained piece of code
  - Malicious code spreading with no or little user involvement
- Trojans
  - Malicious code hidden in seemingly innocent program downloaded
- Logic Bombs
  - Malicious code hidden in programs already on your machine

Other malicious code

- Web bugs (beacon)
- Back doors
- Salami attacks
- Privilege escalation
- Rootkits
- Keystroke logging
- Interface illusions
- Phishing
- Man-in-the-middle attacks

Covert channels and side channels

Covert channel: An attacker creates a capability to transfer sensitive/unauthorized information through a channel that is not supposed to transmit that information.

Side channel: Sensitive/unauthorized information is leaked through a channel that is not supposed to transmit that information.

Examples of such channels:

- Bandwidth consumption
- Timing computations
- Electromagnetic emission
- Sound emissions

Design-time security controls

Q: How can we design programs so that they're less likely to have security flaws?

- Modularity
- Encapsulation
- Information hiding
- Mutual suspicion
- Confinement

Implementation-time security controls

Q: When you're actually coding, what can you do to control security flaws?

- Don't use C (but this might not be an option)
- Static code analysis
- Hardware-assistance
- Formal methods
- Genetic diversity

Software-lifecycle security controls

- Change management
- Code reviews
- Testing
- Documentation
- Maintenance

Outline

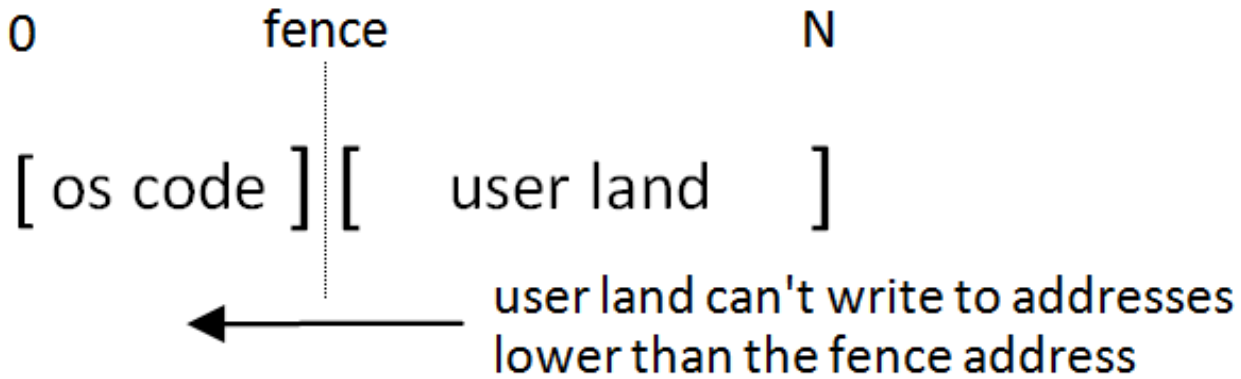
Protected objects by the OS

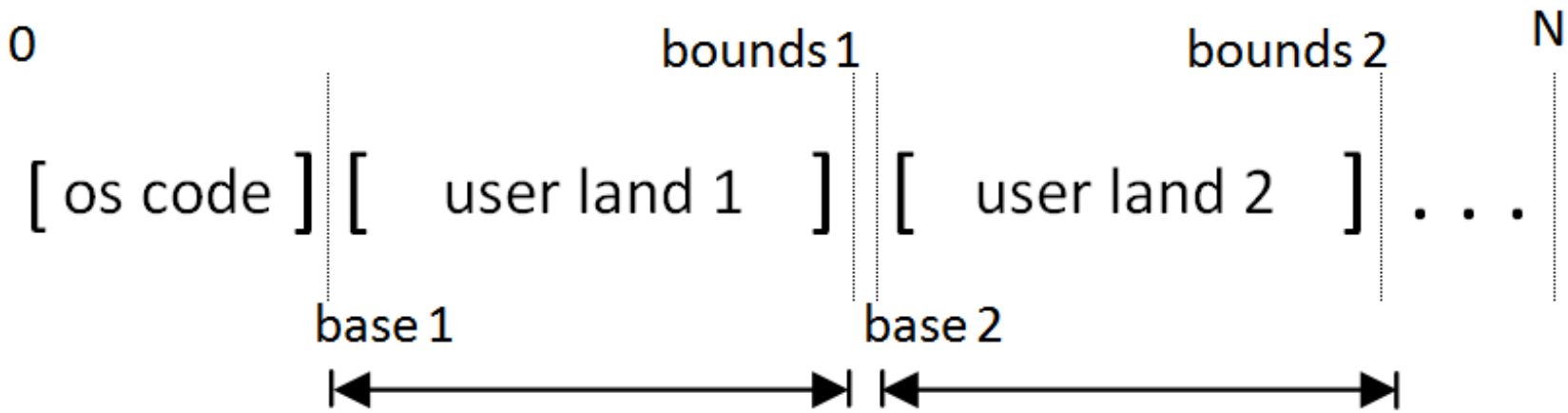
An operating system needs to handle the separation and sharing of the following resources:

- CPU
- Memory
- Cache
- Disk
- I/O devices (disks, printers, keyboards, sensors, ...)
- Networks

Amongst different programs as well as between the kernel and the programs.

Memory and address protection



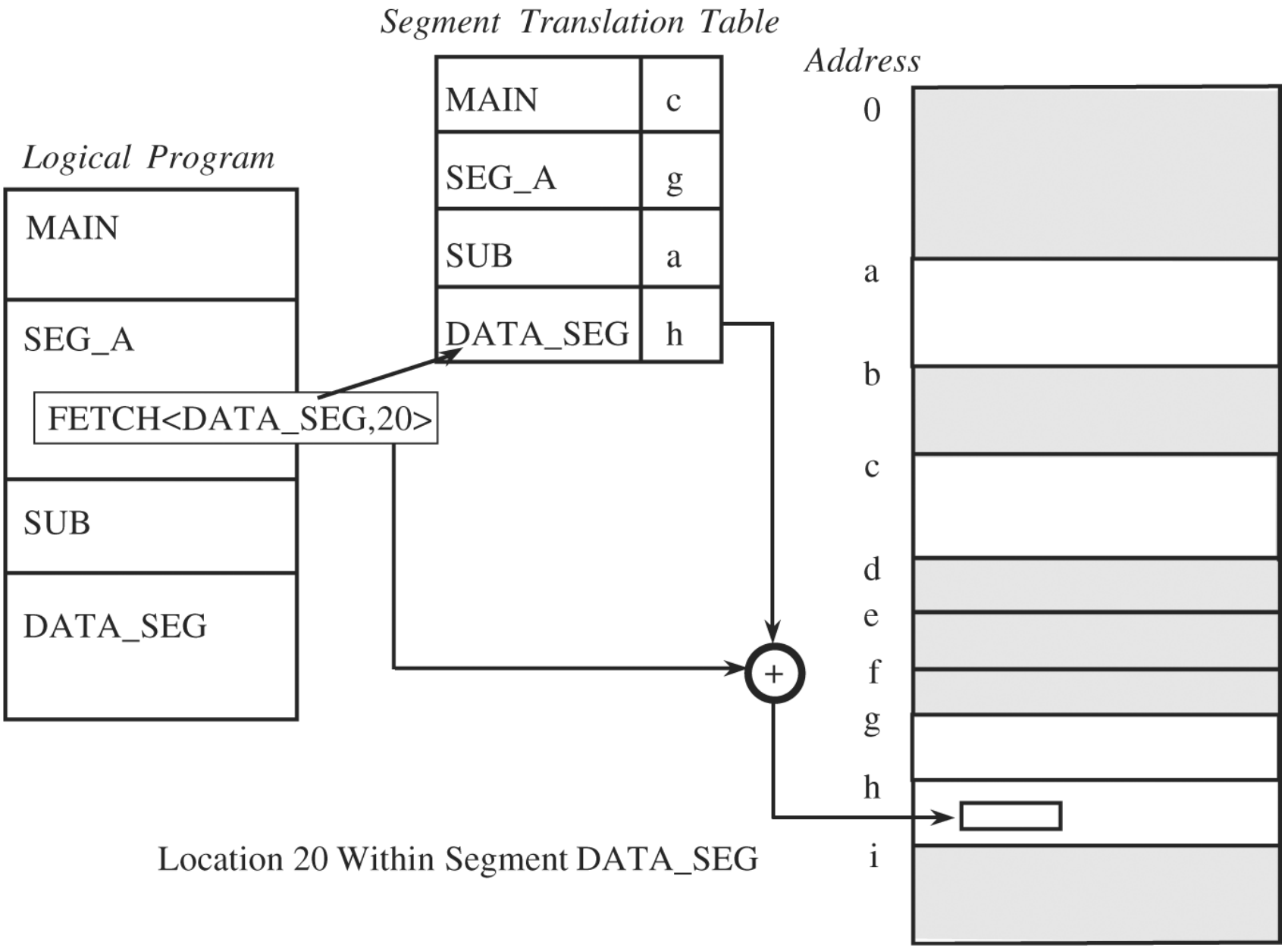


"user land n" can't write to addresses lower than the "base n" address or higher than the "bounds n" address

Segmentation

- Each program has multiple address spaces (segments)
- Different segments for code, data, and stack
  - Or maybe even more fine-grained, e.g., different segments for data with different access restrictions
- Virtual addresses consist of two parts:
  - <segment name, offset within segment>
- OS keeps mapping from segment name to its base physical address in Segment Table
  - A segment table for each process
- OS can (transparently) relocate or resize segments and share them between processes
- Segment table also keeps protection attributes

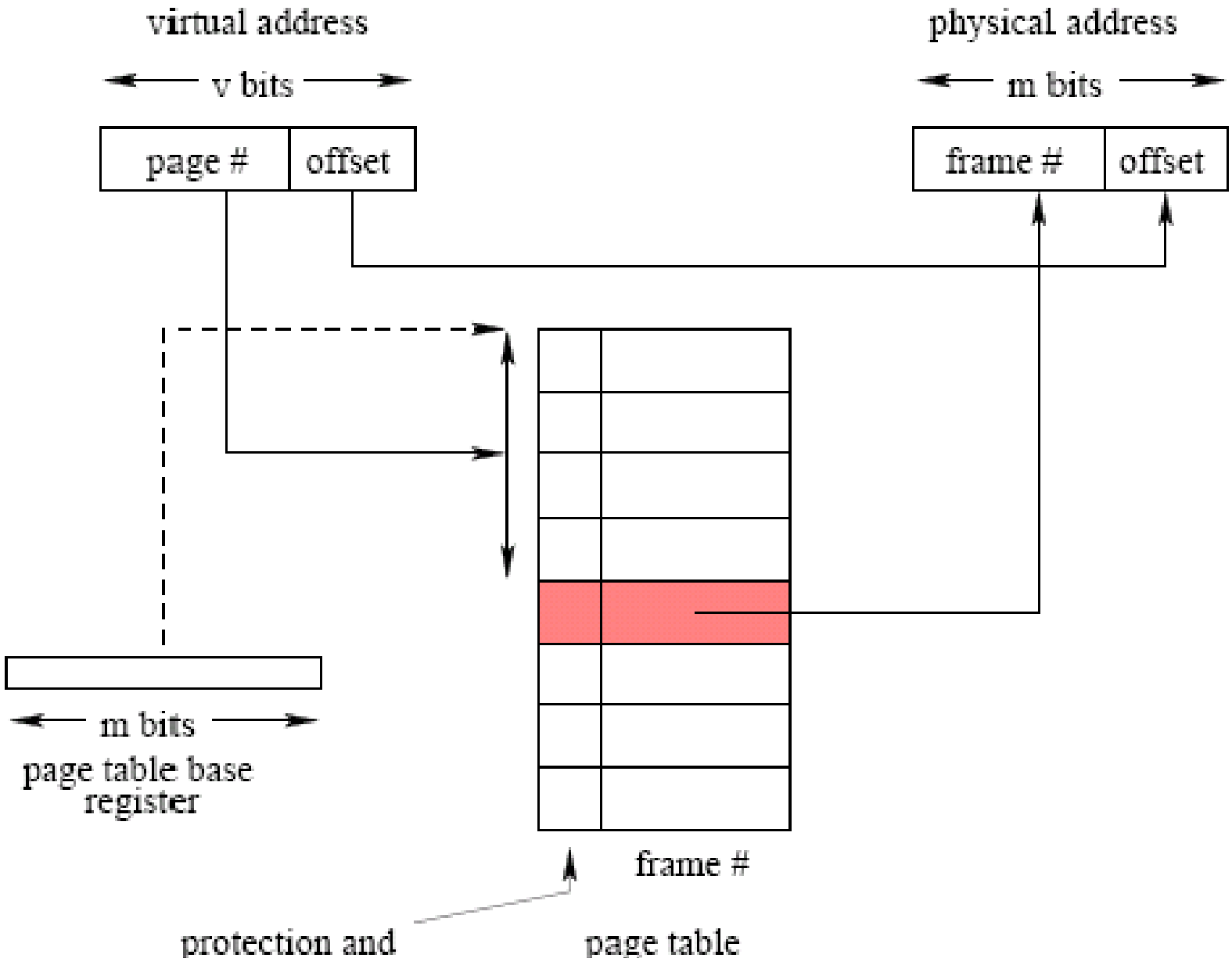
Segment table



Paging

- Program (i.e., virtual address space) is divided into equal-sized chunks (pages)
- Physical memory is divided into equal-sized chunks (frames)
- Frame size equals page size
- Virtual addresses consist of two parts:
  - <page #, offset within page>
  - # bits for offset = $\log_2$(page size)
- OS keeps mapping from page # to its base physical address in Page Table
- Page table also keeps memory protection attributes

Page table


virtual address
v bits
page #
offset
physical address
m bits
frame #
offset
m bits
page table base register
frame #
protection and
page table

Access control matrix

- Set of protected objects: O
  - e.g., files or database records
- Set of subjects: S
  - e.g., humans (users), processes acting on behalf of humans or group of humans/processes
- Set of rights: R
  - e.g., $\{$read, write, execute, own$\}$
- Access control matrix consists of entries $a[s, o]$, where $s \in S$, $o \in O$ and $a[s, o] \subseteq R$

Access control matrix implementation

Access control lists (ACLs) Each object has a list of subjects and their access rights

Capabilities A capability is an unforgeable token that gives its owner some access rights to an object

- Unforgeability enforced by having OS store and maintain tokens or by cryptographic mechanisms
- Tokens might be transferable (e.g., if anonymous)

Authentication factors

- Four classes of authentication factors
- Something the user knows
  - Password, PIN, answer to "secret question"
- Something the user has
  - ATM card, badge, browser cookie, physical key, uniform, smartphone
- Something the user is
  - Biometrics (fingerprint, voice pattern, face,...)
  - Have been used by humans forever, but only recently by computers
- Something about the user's context
  - Location, time, devices in proximity

Password security

- Store only a digital fingerprint of the password (using a cryptographic hash) in the password file
- When logging in, system computes fingerprint of entered password and compares it with user's stored fingerprint
- Still allows offline guessing attacks when password file leaks

- UNIX makes guessing attacks harder by including user-specific salt in the password fingerprint
- Two users who happen to have the same password will likely have different fingerprints
- Makes guessing attacks harder, can't just build a single table of fingerprints and passwords and use it for any password file

Interception attacks and challenge-response protocols

An attacker may intercept password (or its fingerprint) while it is in transmission from client to server

Solution: One-time passwords make intercepted password useless for later logins.

1. Server sends a random challenge to a client
2. Client uses the received challenge and the (long-term) password to compute a one-time password
3. Client sends one-time password to server
4. Server checks whether client's response is valid given that the server also knows the (long term) password

NOTE: Given intercepted challenge and response, attacker might be able to brute-force password if it is too short

Level of protections

Level of privileges:

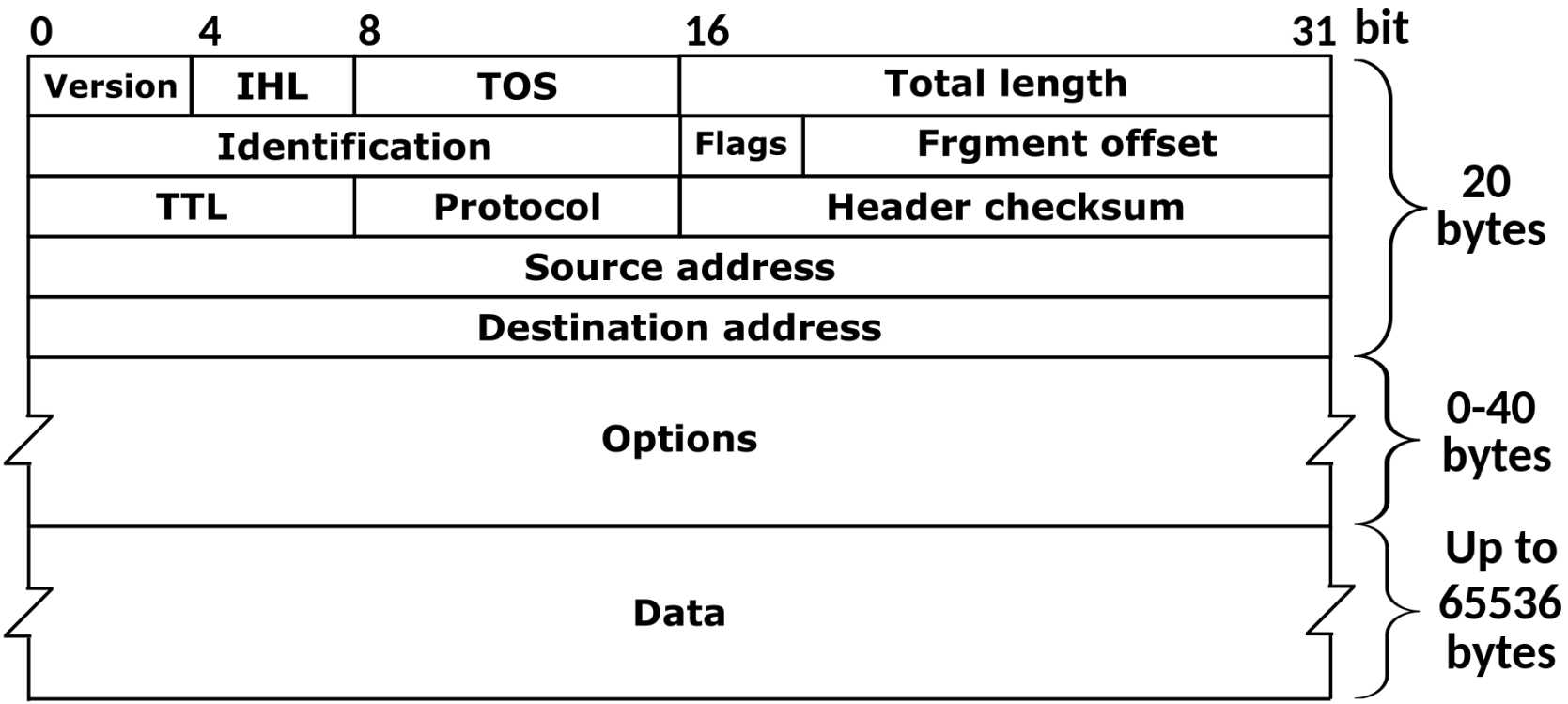
- Virtualization
- Kernel
- Reference monitor
- Application insulation

Mechanism of controls:

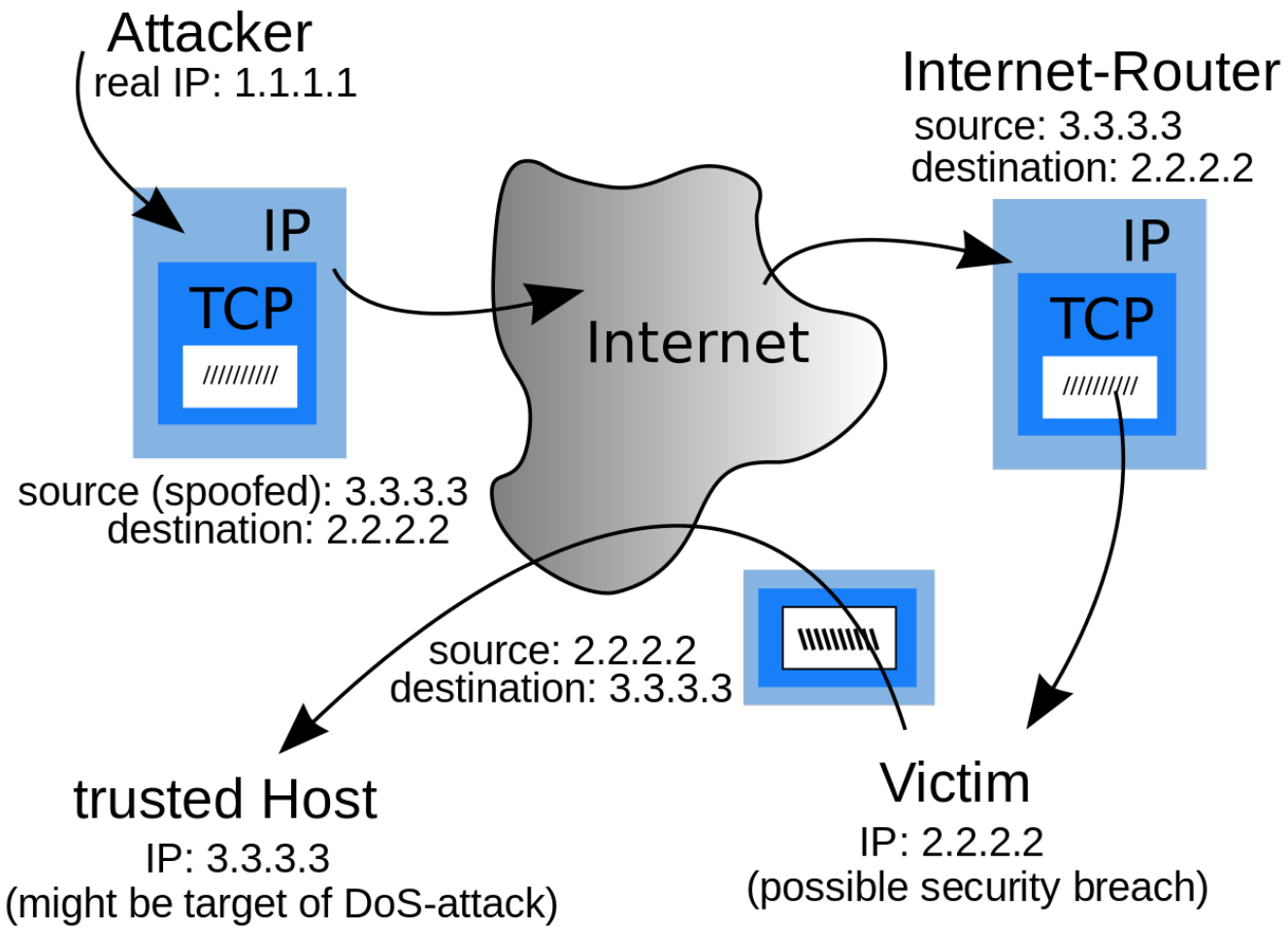
- Chroot
- Containers
- Compartmentalization
- `setuid`/`suid` bit

Outline

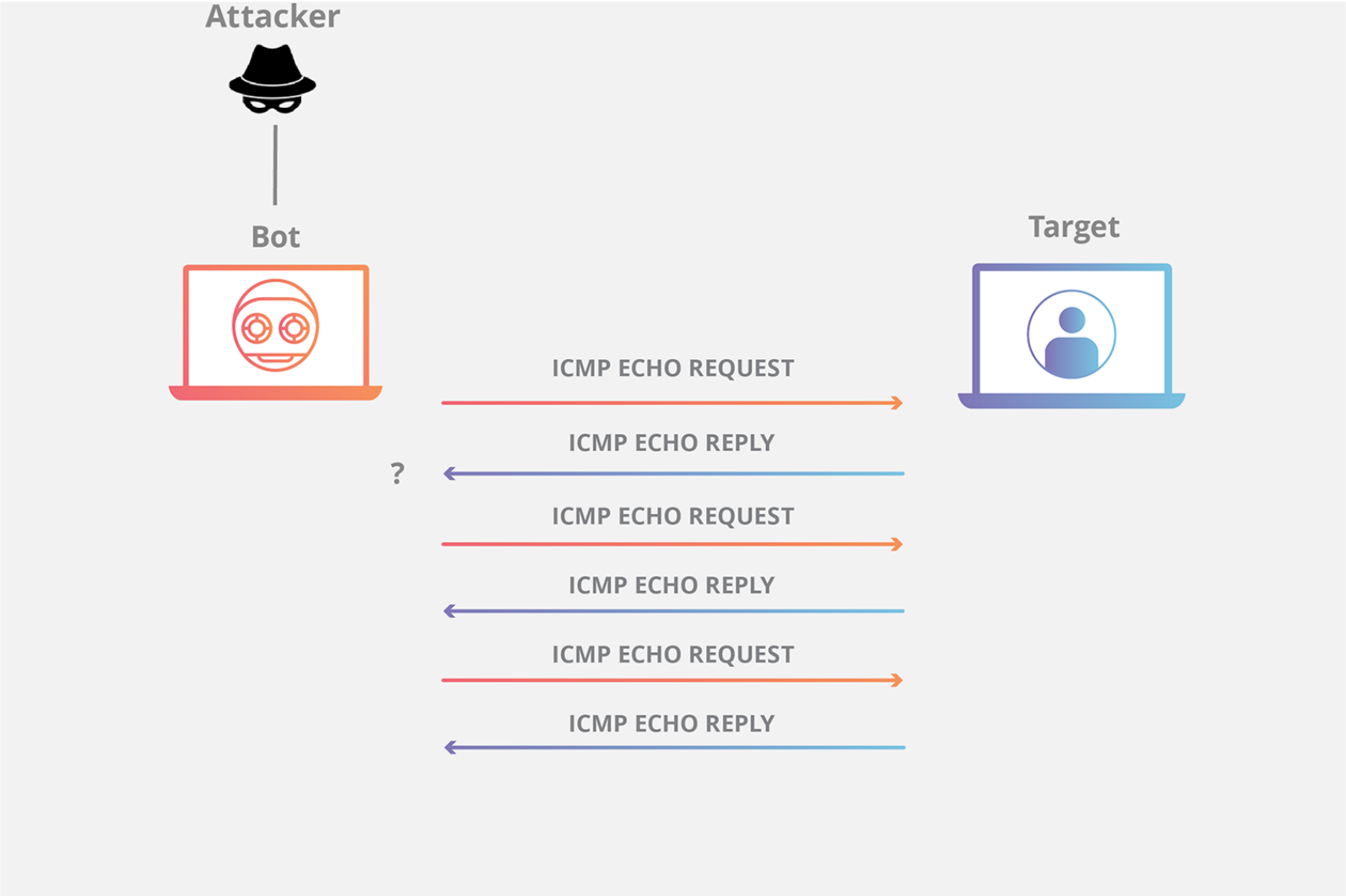
IPv4 packet


0
4
8
16
31
bit
Version
IHL
TOS
Total length
Identification
Flags
Frgment offset
TTL
Protocol
Header checksum
Source address
Destination address
Options
Data
20 bytes
0-40 bytes
Up to 65536 bytes

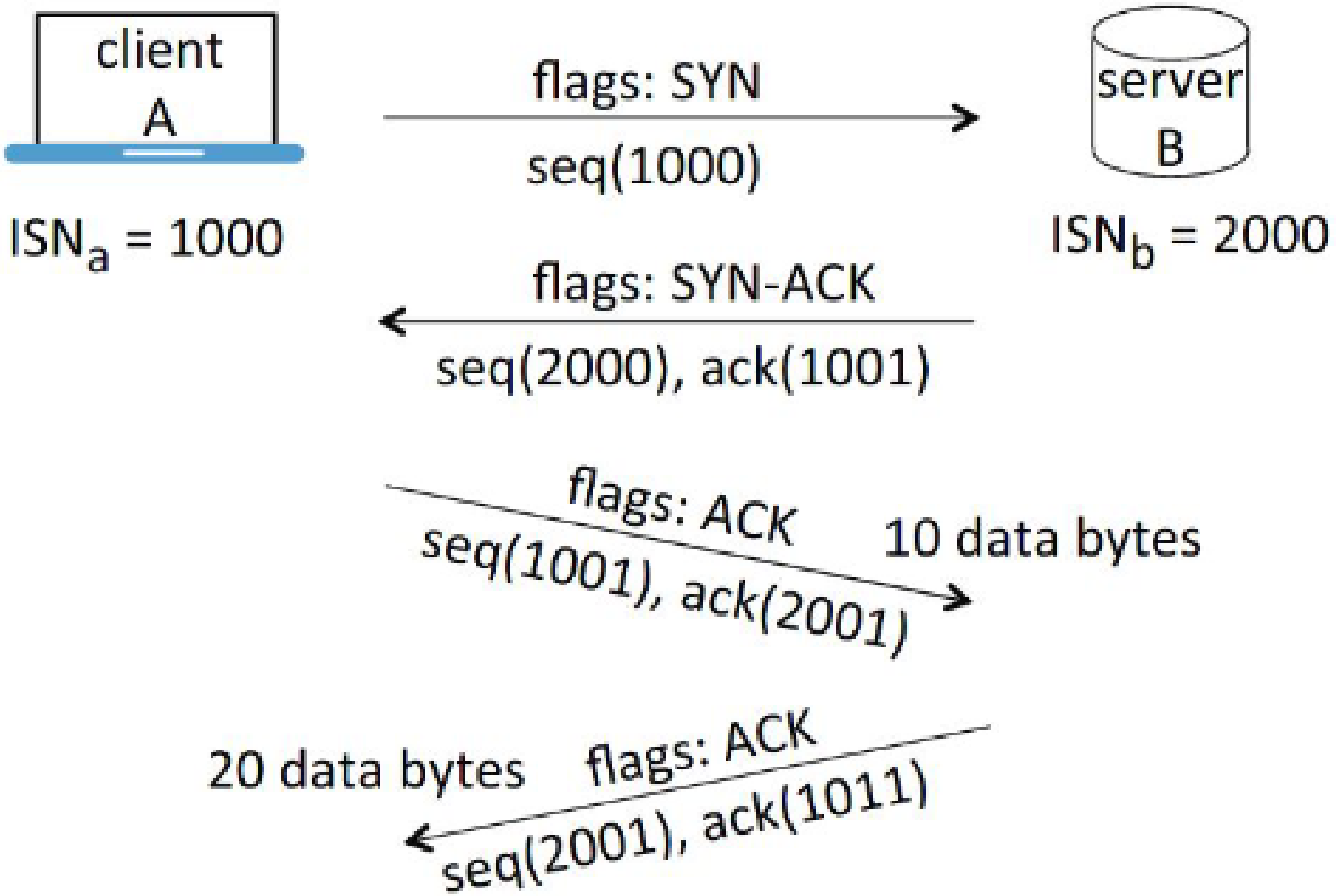
IP spoofing example



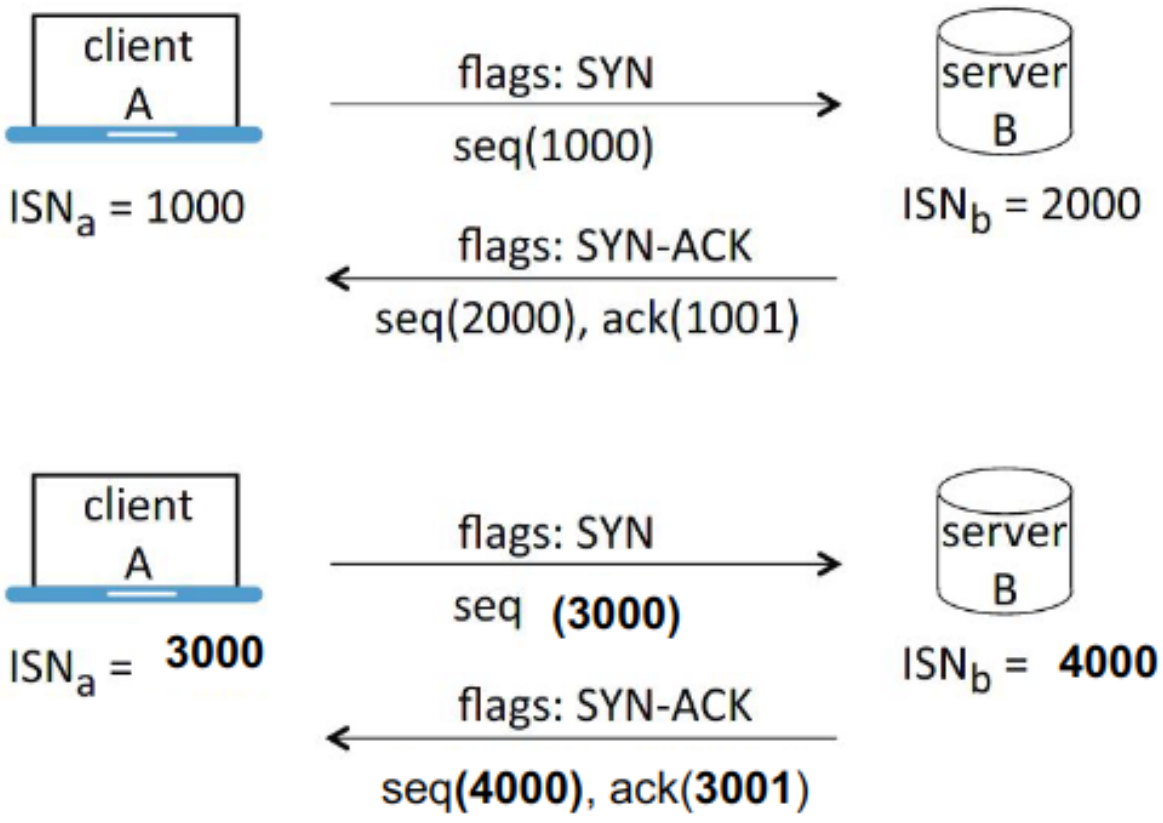
Ping (ICMP) flood



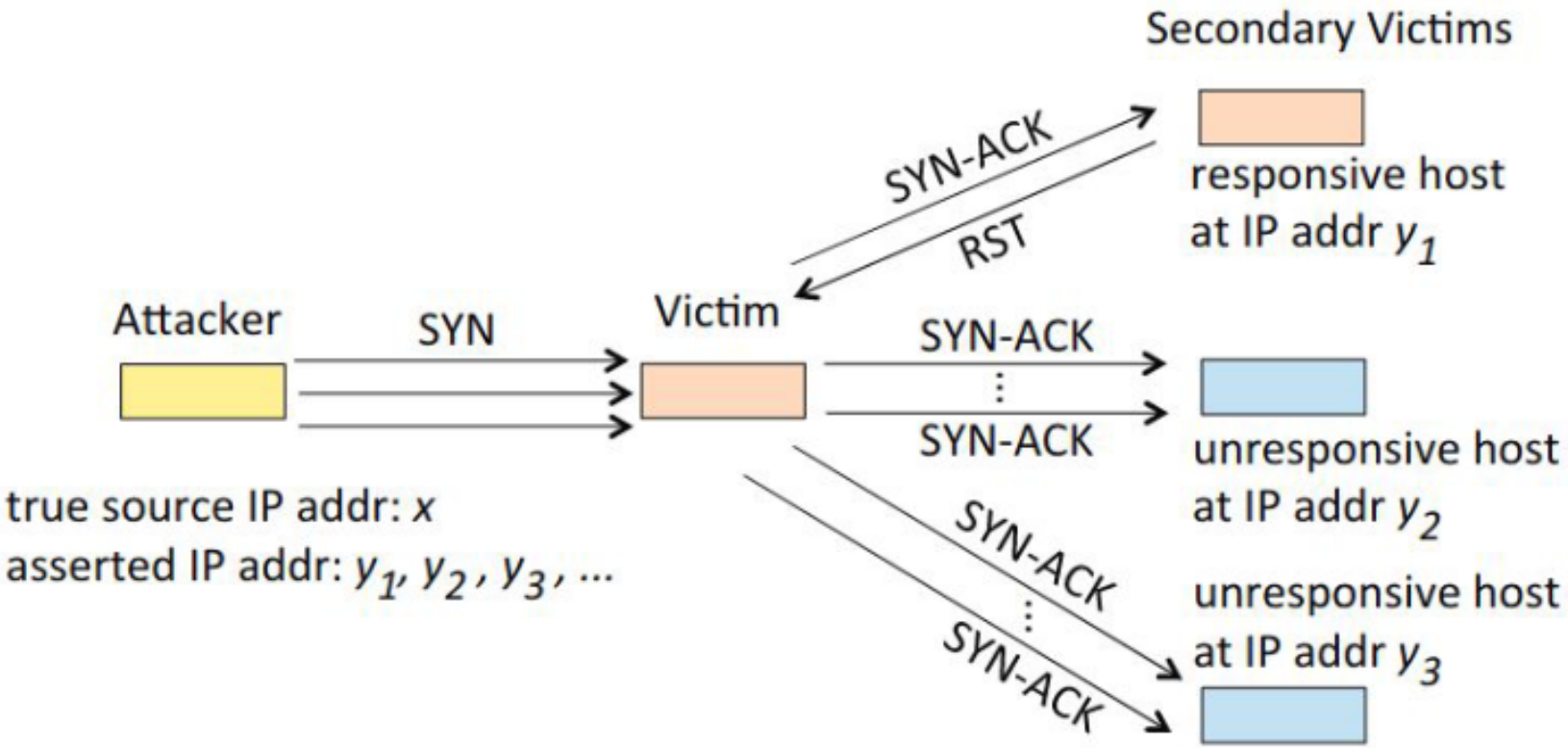
TCP three-way handshake



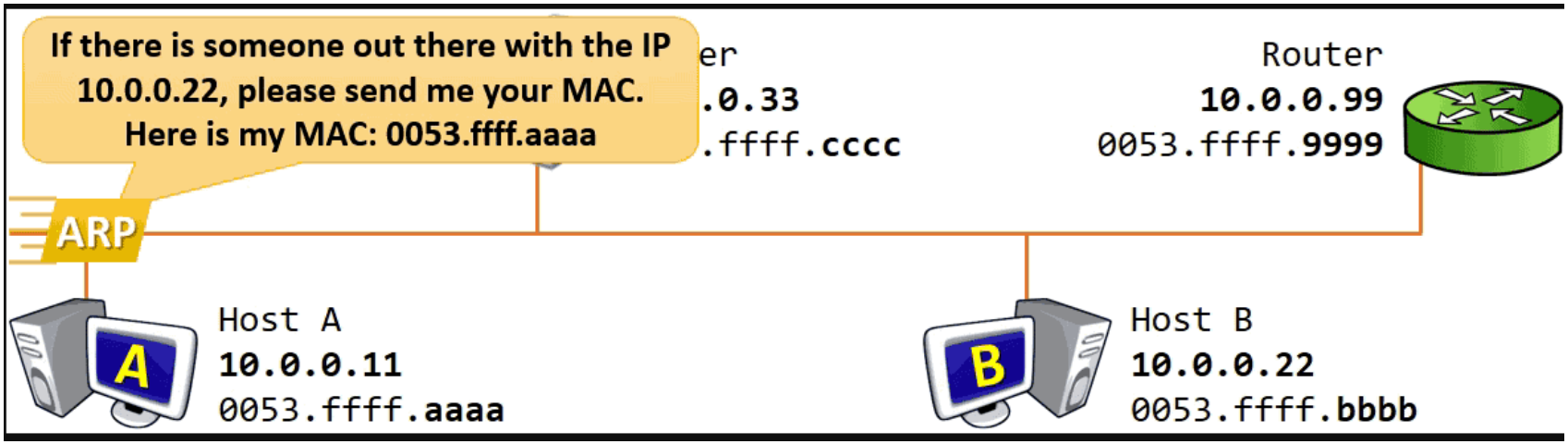
Basic SYN flood

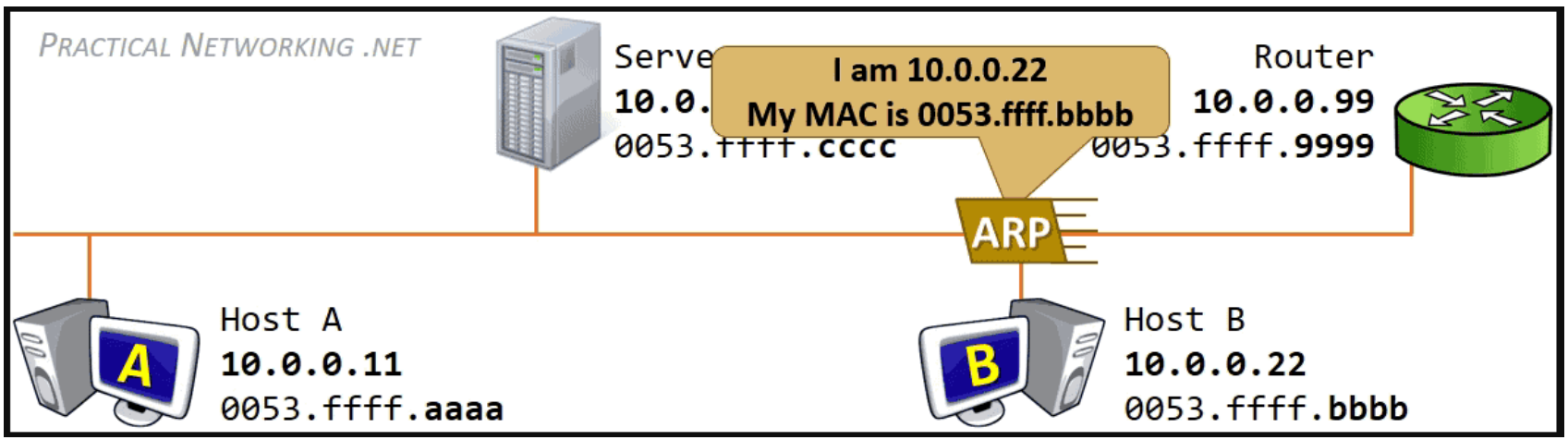


Basic SYN flood with IP spoofing



Address Resolution Protocol (ARP)





Outline

Kerckhoffs' principle

Kerckhoffs's principle: a cryptosystem should be secure, even if everything about the system, except the key, is public knowledge

Which can also be re-stated as:

Shannon's maxim: one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them.

Secret-key encryption

The key Alice uses to encrypt the message is the same as the key Bob uses to decrypt it:

$$D_k(E_k(m)) = m$$

One-time pad: perfect secret-key encryption

- k is a truly random bitstring of the same length as m
- The "Encrypt" and "Decrypt" functions are both XOR

Ciphers and modes of operations

- Stream ciphers
- Block ciphers: ECB (not secure), CBC, CTR, etc.

Public-key encryption

1. Bob creates a key pair (e_k, d_k)
2. Bob gives everyone a copy of his public encryption key e_k
3. Alice uses it to encrypt a message, and sends the encrypted message to Bob
4. Bob uses his private decryption key d_k to decrypt the message
   - Eve can't decrypt it; she only has the encryption key e_k
   - Neither can Alice!
   - It must be hard to derive d_k from e_k

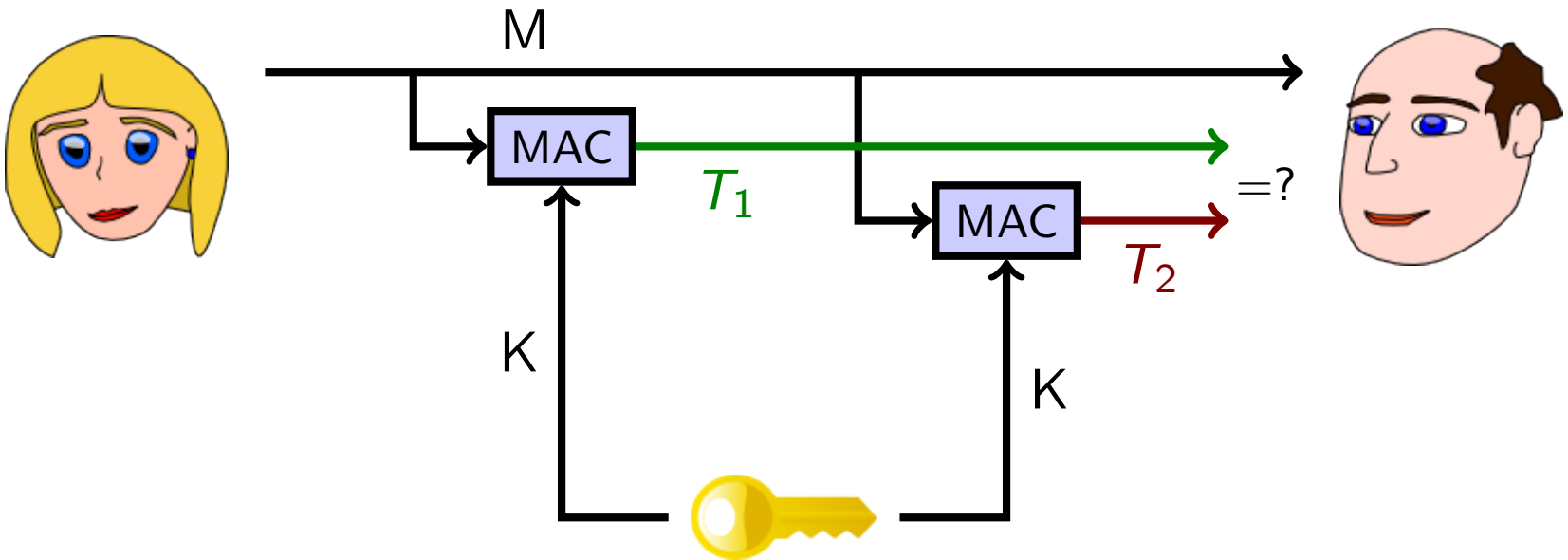
Cryptographic hash functions

A hash function h takes an arbitrary length string x and computes a fixed length string $y = h(x)$ called a message digest

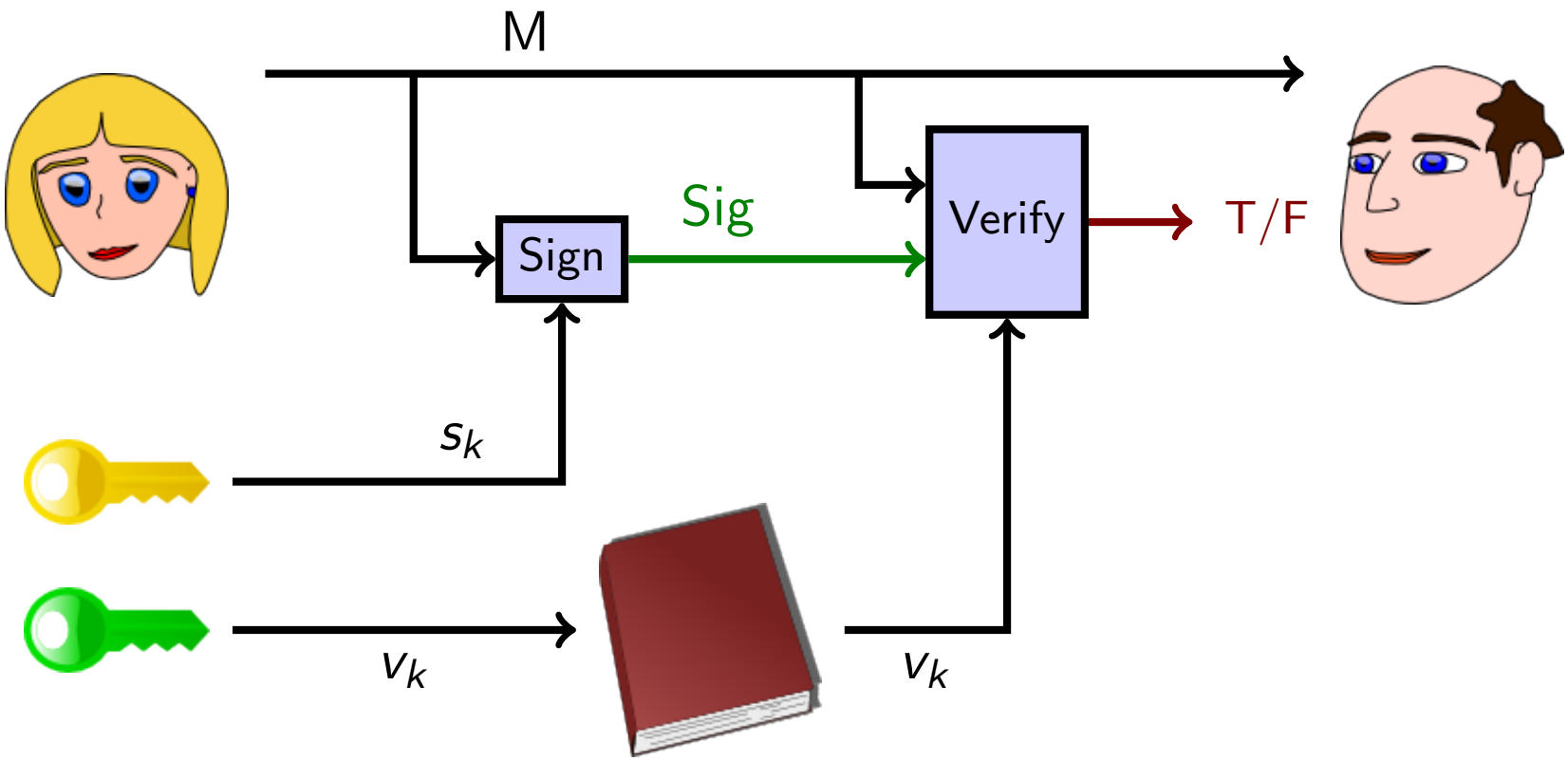
Hash functions should have three properties:

- Preimage-resistance:
  - Given y, it's hard to find x such that $h(x) = y$
- Second preimage-resistance:
  - Given x, it's hard to find $x' \neq x$ such that $h(x) = h(x')$
- Collision-resistance:
  - It's hard to find any two distinct values x, x' such that $h(x) = h(x')$

Message authentication codes (MAC)



Digital signatures



Diffie-Hellman key exchange

- Alice chooses prime p at random and finds a generator g
- Alice chooses $X \leftarrow_R \{0, 1, \ldots, p-1\}$ and sends $A = g^X \pmod{p}$ to Bob, together with p and g
- Bob chooses $Y \leftarrow_R \{0, 1, \ldots, p-1\}$ and sends $B = g^Y \pmod{p}$ to Alice
- Alice and Bob both compute $s = g^{XY} \pmod{p}$
  - Alice does that by computing $B^X \pmod{p}$
  - Bob does that by computing $A^Y \pmod{p}$

Now they share a common secret s which can be used to derive a symmetric key

IPSec: modes of operations

A regular IP packet in the form of $\langle$ H $\|$ P $\rangle$ can be transformed into an IPSec packet depending on the mode of operation:

	AH	**ESP**
Transport	H $\|$ AH $\|$ P $\hookrightarrow$ Int. of H and P	H $\|$ ESP-H $\|$ $\langle$ P $\rangle_k$ $\|$ ESP-T $\hookrightarrow$ Int. and Conf. of P only
Tunnel	H' $\|$ AH $\|$ $\langle$ H $\|$ P $\rangle$ $\hookrightarrow$ Int. of H and P	H' $\|$ ESP-H $\|$ $\langle$ H $\|$ P $\rangle_k$ $\|$ ESP-T $\hookrightarrow$ Int. and Conf. of H and P

PGP: web-of-trust via signing keys

- Once Alice has verified Bob's key, she uses her signature key to sign Bob's key
  - i.e., Alice signs a message that says "*I have verified that the key with fingerprint* `B117 ... 8BF5` *really belongs to Bob*"
- Bob can attach Alice's signature to the key on his webpage

- If Carol doesn't know Bob, but does know Alice (and has already verified Alice's key, and trusts her to introduce other people):
  - she downloads Bob's key from his website
  - she sees Alice's signature on it
  - she is able to use Bob's key without checking with Bob personally

OTR: deniable authentication

Alice and Bob chat online in a way that:

- They can decrypt and authenticate each other's messages but,
- No one else knows what they say
- No one can prove what was said

- Do not want digital signatures
  - Non-repudiation is great for signing contracts, but undesirable for private conversations
- Use Message Authentication Codes (MAC)

Private information retrieval (PIR)

Goal: allow a user to query a database while hiding the identity of the data-items the user is after

Formal model:

- Server: holds an n-bit string $\{X_1, X_2, ..., X_n\}$
- User: wishes to retrieve X_i AND keep i private

Comparison of CPIR and IT-PIR

CPIR

- Possible with a single server
- Server needs to perform intensive computations
- To break it, the server needs to solve a hard problem

IT-PIR

- Only possible with > 1 server.
- Server may need lightweight computations only
- To break it, the server needs to collude with other servers

Outline

DAC for databases

DAC is built-in in the SQL language.

- Use the `GRANT` keyword to assign a privilege to a user
- Use the `REVOKE` keyword to withdraw a privilege.

Fine-grained DAC using SQL views

Accounts A1, A2, A3

Relations: `Employee(Name, SIN, DOB, Address, Salary, Dpt)`

Create a view

```
> A2:  CREATE VIEW CSEmployeePublicInfo
           SELECT Name, DOB, Address FROM Employee
           WHERE Dpt = "CS";
```

The table owner (A2) creates a view that only expose the (`Name`, `DOB`, `Address`) information for `Employees` in the CS department.

Relation-level privilege via views

```
> A2:  GRANT SELECT ON CSEmployeePublicInfo TO A3;
```

The table owner (A2) grants user A3 the privilege to run `SELECT` queries on the restrict view instead of the whole `Employee` table.

RBAC for databases

Creating and using roles

```
> Admin:  CREATE ROLE "DptAdmin", "CompanyHR";

> Admin:  GRANT "DptAdmin" TO A1;
> Admin:  GRANT "CompanyHR" TO A3;

> A2:  GRANT SELECT ON CSEmployeePublicInfo TO "DptAdmin";
> A2:  GRANT UPDATE ON Employee(Address) TO "CompanyHR";
```

Element integrity

Example on element integrity violations

```
CREATE TABLE Employee (Name VARCHAR(255), Age INTEGER);
INSERT INTO Employee VALUES ("SMITH", 400);
```

The type system is not expressive enough. There is no way to restrict that `Age` must be in a proper range (e.g., 0-150).

And there are even more tricky situations, for example:

- At all times, there is at most one employee can have the `Position` attribute set to "CEO".
- A salary increase cannot exceed 100% of the current salary.

Referential integrity

Referential integrity ensures that each value of a foreign key *refers* to a valid primary key value, i.e. there are no dangling foreign keys.

One use case: to prevent accidental or intentional deletion of records that are still being used.

Inconsistent state

Recall that integrity is about ensuring the data records are in a sensible/correct state at all times.

But what if a transaction requires two or more write operations?
For example: transfer money from Alice to Bob requires two `UPDATE`:

- `UPDATE Ledger SET Balance = Balance - 100 WHERE Name = "Alice";`
- `UPDATE Ledger SET Balance = Balance + 100 WHERE Name = "Bob";`

Q: What happens if the database fails after the first `UPDATE`?

Data race

If two clients send the request concurrently, what will be the result?

Client 1

```
SELECT @balance = Balance
  FROM Ledger WHERE Name = "Alice";

UPDATE Ledger SET Balance =
  @balance - 100 WHERE Name = "Alice";
```

Client 2

```
SELECT @balance = Balance
  FROM Ledger WHERE Name = "Alice";

UPDATE Ledger SET Balance =
  @balance - 100 WHERE Name = "Alice";
```

One possible interleaving:

Transaction interleavings

```
SELECT @balance = Balance FROM Ledger WHERE Name = "Alice";
SELECT @balance = Balance FROM Ledger WHERE Name = "Alice";
UPDATE Ledger SET Balance = @balance - 100 WHERE Name = "Alice";
UPDATE Ledger SET Balance = @balance - 100 WHERE Name = "Alice";
```

Q: How much is deducted from Alice's balance?

Privacy notions

k-**anonymity**: For each published record, there exists at least $k-1$ other records with the same quasi-identifier (where $k \geq 2$)

ℓ-**diversity**: For any quasi-identifier value, there should be at least ℓ distinct values of the sensitive fields (again $\ell \geq 2$)

t-**closeness**: Distribution of sensitive attribute values in each equi-class should be close to that of the overall dataset

- Privacy is measured by the information gain of an observer.
- The gain is the difference between
  - *prior belief*, what the observer knows *before* seeing the data, and
  - *posterior belief*: what the observer knowns *after* seeing the data.

Neighboring databases

Two databases D_1 and D_2 are neighbouring if they agree except for a single entry.

- *Unbounded DP*: D_1 and D_2 are neighboring if D_2 can be obtained from D_1 by adding or removing one element
- *Bounded DP*: D_1 and D_2 are neighboring if D_2 can be obtained from D_1 by replacing one element

ϵ-differential privacy

Idea: If the mechanism M behaves nearly identically for D_1 and D_2, then an attacker can't tell whether D_1 or D_2 was used (and hence can't learn much about the individual).

Definition:
A mechanism $M : X \to Y$ is ϵ-differentially private (ϵ-DP) if for any two neighboring databases $D_1 : X$ and $D_2 : X$:

$$\forall T \subseteq Y, \quad \Pr[M(D_1) \in T] \leq e^{\epsilon} \Pr[M(D_2) \in T]$$

Sensitivity

Definition: given a query processing function $f : X \to \mathbb{R}^k$, the ℓ_1-sensitivity of f is defined as:

$$\Delta_1^f = \max_{D_1 \sim D_2} \|f(D_1) - f(D_2)\|_1 \quad \text{where } D_1, D_2 \in X$$

NOTE 1: The range of f is k-dimensional

NOTE 2: ℓ_1-sensitivity is $\|\vec{x_1} - \vec{x_2}\|_1 = \sum_i |\vec{x_1}[i] - \vec{x_2}[i]|$

Sensitivity w/ one pair of neighboring databases

D1 with Alice enrolled:

- Alice: 90
- Everyone else (29 of them): 50

D2 with Alice not enrolled:

- Everyone (30 of them): 50

Algorithm: You are allowed to make a query that returns the average score of this course.

Q: What is the ℓ_1-sensitivity here?
A: $|\text{Avg}(D_1) - \text{Avg}(D_2)| = 1.33$

Sensitivity w/ more database candidates

Q: What if we don't know the scores?

Suppose we only know that each student's score $\in [0 - 100]$, and

- (in bounded DP): there are 30 students enrolled
- (in unbounded DP): there are 29 or 30 students enrolled

Algorithm: You are allowed to make a query that returns the average score of this course.

Q: What is the ℓ_1-sensitivity here?

Sensitivity w/ more database candidates - bounded

Suppose we only know that each student's score $\in [0 - 100]$, and there are 30 students enrolled in the course.

Algorithm: You are allowed to make a query that returns the average score of this course.

$$
\begin{aligned}
\ell_1 &= \max(|\frac{\sum_{29 \text{ students}} + k_1}{30} - \frac{\sum_{29 \text{ students}} + k_2}{30}|) \\
&= \frac{1}{30} \max(|k_1 - k_2|) \\
&= \frac{1}{30} \times 100 \qquad \hookleftarrow (k_1 = 0 \wedge k_2 = 100) \vee (k_1 = 100 \wedge k_2 = 0) \\
&= \frac{10}{3}
\end{aligned}
$$

Sensitivity w/ more database candidates - unbounded

Suppose we only know that each student's score $\in [0 - 100]$, and there are either 29 or 30 students enrolled in the course.

Algorithm: You are allowed to make a query that returns the average score of this course.

$$
\begin{aligned}
\ell_1 &= \max(|\frac{\sum_{29 \text{ students}}}{29} - \frac{\sum_{29 \text{ students}} + k}{30}|) \\
&= \max(|\frac{\sum_{29 \text{ students}}}{29 \times 30} - \frac{k}{30}|) \\
&\xrightarrow{\text{case1}} \max(\frac{\sum_{29 \text{ students}}}{29 \times 30}) - \min(\frac{k}{30}) \\
&\xrightarrow{\text{case2}} \max(\frac{k}{30}) - \min(\frac{\sum_{29 \text{ students}}}{29 \times 30}) \\
&= \frac{10}{3} \text{ for both cases}
\end{aligned}
$$

Laplace mechanism

Definition: Let $f : X \to \mathbb{R}^k$ is the function that calculates the "true" value of a query. The Laplace mechanism is defined as:

$$M(D) = f(D) + (Y_1, Y_2, \cdots, Y_k)$$

where Y_i are independent and identically distributed (i.i.d) random variables sampled from $\text{Lap}(\frac{\Delta_1^f}{\epsilon})$

Outline

Law vs ethics

- Laws are a set of formal rules that governs how we behave as members of a society.
- The goal is to create a set of basic and objectively enforceable standard of behaviors.
- Specifies, in greater details, what we MUST do and more frequently, what we MUST NOT do.
- Laws are upheld and applied by a state-backed justice system.

Q: Why laws are not enough?

- The lengthy legislative process does not match with the fast-pacing tech industry
- Laws usually have a very narrow focus.

Responsible disclosure

Q: You have found a security vulnerability, what should you do?

Coordinated vulnerability disclosure

- A private full disclosure to all responsible parties (e.g., software vendors for most software bugs)
- Wait for either a patch from the responsible parties of a specific amount of time (e.g., 90-days or 120-days)
- A public partial disclosure if you want to further pressure the responsible parties; or a public full disclosure in the interests of potential victims.

Talk to independent experts

Institutional review board (IRB), a.k.a., independent ethics committee (IEC), ethical review board (ERB), or research ethics board (REB), etc...

is a committee that applies research ethics by reviewing the methods proposed for research to ensure that they are ethical.

Codes of professional ethics

You will probably be a member of one or more professional societies

- Association for Computing Machinery (ACM)
- Institute of Electrical and Electronics Engineers (IEEE)
- Canadian Information Processing Society (CIPS)

These organizations have codes of professional ethics

Types of intellectual property

Four kinds of IP here:

- Trade secrets,
- Trademarks,
- Patents, and
- Copyrights

These four kinds of IP:

- Cover different intangibles
- Convey different rights
- Have different durations
- Use different registration process

Risk assessment

Definition: A risk is a potential problem that a system or its users may experience

Risks have two important characteristics:

- Probability: what is the probability (between 0 and 1) that the risk will occur? (That is, the risk will turn into a problem)
- Impact: if the risk occurs, what harm will happen? This is usually measured in terms of money (cost to clean up, direct losses, PR damage to the company, etc.)

The risk exposure = probability $\times$ impact

Project savings due to control

- The expected cost of not controlling the risk is just the risk exposure, as computed earlier
- For each control, the cost of the control is its direct cost (e.g., buying the network monitoring equipment, training, etc.), plus the exposure of the controlled risk
  - Most controls aren't perfect: even with the control, there will still be a (smaller, hopefully) probability of a problem
- Savings = Risk exposure − Cost of control − New risk exposure