11/12/21 Notes Security & Cryptography

Cryptography:
- Goal: Render a message incomprehensible to all except recipient
- Requirement: Use a well-known algorithm to encrypt.
  - But why?
  - Relying upon the secrecy of the algorithm is a very bad idea.
    - German Enigma
    - Japanese purple
  - Algorithm has 2 inputs: data & Key
    - Key is known only to authorized users.

Cryptography Basics
- Algorithm (E, D) are publicly known.
- Keys (K_E, K_D) represent a shared secret.
- The E_i

Caesar Cipher
- Reputed to have been used by the Gallic Wars
- Replace each letter w/ the letter 3 positions down the, iterates modular arithmetic

Unbreakable Code
- There is such a thing as an unbreakable code: it's called the one time pad.
  - Use truly random key as long as the message
    - XOR the message w/ key of a bit at a time
  - Code is unbreakable b/c
    - Key could be any string of bits
    - Plaintext message be any message given appropriate key
  - Difficulty: distributing the key is as hard as distributing the message.
    - May be easier b/c of timing.
  - Difficulty: generating truly random bits

Modern Encryption Algorithms

- Data Encryption Standard (DES)
  - Use 56 bit keys
  - Same key is used to encrypt & decrypt
  - Needs to try 2^{56} different keys, on average
  - But... Modern computers can try billions of keys per second
- Current algorithm (AES) use 128-bit key
  - Adding one bit to the key makes it twice as hard to guess
  - Must try 2^{127} keys, on average, to find the right one

Attacking AES

- Suppose that we have an infinitely fast computer
  - But such a computer must still consume energy
- There is a fundamental limit on the energy required to flip a bit.
- The Landauer principle says that the energy is required to flip a bit:
  - $E \geq k T \ln 2$ Joules
  - k

Simon

- family of lightweight block ciphers publicly released by NSA
- Simon is a balanced Feistel cipher w/ a n-bit word, and therefore the block length is $2n$.

Public-key Cryptography
- Instead of using a single shared ~~secret~~ secret, keys come in pairs.
  - public key K_U
  - private key K_R
  - $K_U \neq K_R$
- These keys are typically - but not always - inverse of one another
- Encryption & decryption are the same algorithm, i.e.
  - $E(K_U, E(K_R, M)) = E(K_R, E(K_U, M)) = M$
- Public key cryptography is usually slow
- Typically used for
  - Encrypting small amounts of data.
  - Establishing a shared Key for symmetric encryption algorithm.
- Most popular method: RSA.

Diffie-Hellman Key exchange
- Named after Whitfield Diffie and Martin Hellman

RSA

- Choose a large prime pair p, q.
  - These are secret, and temporary - you can throw them away
- Let $n = p \times q$
- $\varphi(n) = (p-1)(q-1)$
- choose e a number e such that

Attacking RSA

- Requires factoring n in order to find $\varphi(n) = (p-1)(q-1)$
  - Factoring of large composite is believed to be hard
  - best algorithm is the General Number Field Sieve

RSA - Key Generation

- Public key → integers: n (public modulus), e (public exponent)
  - Choose a large prime p & q.
  - Let $n = p \times q$
  - Compute $\varphi(n) = (p-1)(q-1)$
  - Choose random e such that $\gcd(e, \varphi(n)) = 1$
- Private key is composed of integers n and d.
  - Calculate d such that $ed \equiv 1 \pmod{\varphi(n)}$.

Primality testing.

- we can use the Miller-Rabin numbers.

$c = m^e \pmod{n}$

rsa_make_priv(d, e, p, q)
n = (p-1)(q-1)
mpz_invert(e, n) = d

~~write_log2(mpz_t result,~~
rsa_encrypt(mpz_t c, mpz_t m, mpz_t e, mpz_t n)

mpz_powm(c, m, e, n);

void rsa_encrypt_file(FILE *infile, FILE *outfile, mpz_t n, mpz_t e)

1) ~~static~~
mpz_t k;
mpz_log2n(R, n);
mpz_sub_ui(k, k, 1);
mpz_fdiv_q_ui(k, k, 8);
$k = \lfloor (\log_2 n - 1)/8 \rfloor$

2) ~~uint~~ uint8_t *block = (uint8_t *) calloc(mpz_get_ui(k), sizeof(uint8_t));

3) *block[0] = 0xff

4) mpz_t message;
uint32_t j = 0; buff_index = 1
while(j < sizeof(infile)):
uint32_t true_byte = (uint32_t) fread(block + buff_index, sizeof(uint8_t), k-1,
j += true_byte;
mpz_import(message, (size_t) true_byte, 1, (size_t) ~~true_byte~~ uint8_t, 1, 0, block)
gmp_fprintf(outfile, "%Zx\n", message);

11/15/21 Notes Processes

Processes and threads
- Processes
- Threads
- Scheduling
- Inter-process communication
- Classical IPC problems

What is a process?
- Code, data, and stack
  - Usually has its own address space
- Program state
  - CPU registers
  - Program counter (current location in the code)
  - Stack pointer
- Only one process can be running in a single CPU core @ any given time!
  - Multi-core CPUs can be supporting multiple processes

What is an address space?
- Program executes code
  - Each instruction has an address
- Programs access data
  - Each byte of data also has an address
- We would like to think our program is the only program executing on the computer
  - But we would be wrong
- And address space is the region of computer memory where a program executes.
  - Ideally, it is protected from other programs accessing it.

How is that accomplished?
- loader could relocate the instructions by address a base address to each one.
- There could be registers that point to the first byte of the programs memory (base register) that is added to every mem. addr.

In an ideal world.
- ideal world has memory that is:
  - Very Large
  - Very fast
  - non-volatile
- real world:
  - very large
  - very fast
  - Affordable!
  - Pick any two...
- Memory management goal: make the real world look as much like the ideal world as possible

Memory hierarchy
- mem hierarchy
  - different levels of memory
  - Some are small & fast
  - Others are large and slow
- levels:
  - Cache: small group of fast, expensive memory
  - Main memory: medium-speed, med. price memory (DRAM)
  - Disk: many gigs of slow, cheap, non-volatile storage.
- Memory manager handles memory hierarchy.

Basic memory management.
- Components include
  - Operating System. [w/ device drivers]
  - Single process.

Fixed partitions: multi. programming
- Fixed memory partitions
  - Divide memory into fixed spaces
  - Assign a process to a space when it's free.
- Mechanisms
  - Seperate input queues for each partition
  - Single input queue: better ability to optimize CPU usage.

Multiprogramming system performance
- Arrival and work reqs of 4 jobs.
- CPU utilization for 1-4 jobs w/ 80% I/O wait
- Sequence of events as jobs arrive and finish
  - Assume each job gets 1/4 of CPU time
  - each interval.
  - More processes ⇒ better utilization, less time per process

Memory and multiprogramming.
- Memory needs two things for multiprogramming
  - relocation
  - protection
- The OS cannot be certain where a program will be loaded in memory
- Variables and procedures can't use absolute locations in memory
- Several ways to guarantee this

- The OS must keep processes' memory separate
  - Protect a process from other processes reading or modifying its own memory
  - Protect yourself from modifying its own memory in undesirable ways
- Base and limit registers
  - special CPU registers: base & limit
  - Access to the registers limited to system
  - R registers contain
    - Base: start of the process's memory partition
    - Limit

Allocating memory
- Search through region list to find a large enough space.
- Suppose more than several choices: which one to use
  - First fit: first suitable hole on the list
  - Next fit:

Freeing memory
- Allocation structure must be updated when memory is freed
- Easy w/ bitmaps: just set the appropriate bit in the bitmap.
- Linked lists: modify adjacent elements as needed
  - Merge adjacent free regions into a single region
  - May involve merging 2 regions w/ the freed area

Buddy allocator
- Allocate memory in power of two
  - Good for objects of varied sizes
- Split larger chunk to create 2 smaller chunks
- When chunk is freed, see if it can be combined w/ its buddy to rebuild a larger chunk.

Virtual memory
- Basic idea: allow the OS to hand out more memory than exists on the system
- Keep recently used stuff in physical memory
- Move less recently used stuff to disk
- Keep all of this hidden from process
  - Process still see an address space from 0 - max-addr
  - Movement of information to and from disk handled by OS w/o process help
- VM especially helpful in multiprogrammed system
  - CPU schedules process B while process A waits for its memory to be retrieved from disk

Virtual and physical addresses
- Program use virtual addresses
  - Addresses local to the process
- Hardware translates virtual addresses to physical addresses
- Translation done by the memory Management unit
  - Usually on the same chip as the CPU
  - Only physical addresses leave the CPU/MMU chip
- Physical memory indexed by physical addresses

What's in a page table entry?
- each entry in the page table contains
  - Valid bit - set if the logical page number has a corresponding physical frame in memory
    - If not valid, remainder of PTE is irrelevant
  - Page frame number: page in physical memory
  - referenced bit: set if data on the page has been accessed.
  - Dirty (modified) bit: set if do
- Mapping logical addresses to physical addresses
- split address from CPU into 2 pieces.
  - page number (p)
  - page offset (d)

The process Model
- Conceptual model of multiprogramming for systems
- 4 independent processes
- Processes run sequentially
- Only one process active at any instant
- An instant can be very short
- Only applies if there's a single CPU w/ a single core in the system

When is a process created?
- Processes can be created in 2 ways
  - System initialization: one or more processes created when the OS starts up
  - Execution

11/17/21 Notes De bugging

Definition
- a bug is an error or flaw of a program that produces an unexpected or incorrect output.
- Debugging is the process of identifying and fixing problems.

Kinds of errors (Bugs)
- Syntax Errors.
  - etc.
- Logical errors.
  - off-by-one
  - operator precedence giving unexpected output.
  - etc.
- Semantic errors
  - dividing a float by a string.
  - returning a non-void value in a void function.

Easy Bugs.
- Syntax errors.
- Fixing these bugs:

Hard Bugs.
- Location of the bug is hard to pinpoint.
- Finding these bugs:
  - assert()
  - print statements
    - logging events leading up to seg fault
    - should use fflush() to flush any buffered data.
  - Playing around w/ input / parameters to code until bug is reproducible.
  - Writing test harness to check functionality
    - Can isolate function that work improperly
  - Using specialized debugging tools

assert()
- verifies preconditions and post conditions
  - precondition: condition must be true before executing code.
  - Post condition: condition that must be true after execution of some code
- Assertion checks can be turned off during compile time
  - if define NDEBUG
  - or -DNDEBUG compiler flag
- Sole argument to assert() is a bool expression
  - if expression is true, then nothing happens.
  - if " is false, an error

scan-build
- a static analyzer
  - infer is another static analyzer.
- Static analyzers find bugs in programs w/o running them.
  - works @ compile time

Running infer
- cannot to direct Makefiles.

valgrind
- Dynamic analysis tool used to identify memory management problems
- Invalid read: process tried to read outside its available mem. addrs.
- Invalid write: process tries to write outside its memory address

Static vs. Dynamic Analyzers
- Static analyzers work @ compile time.
  - try to catch, everything, & run things that aren't bugs.
- Dynamic analyzers work @ runtime.

11/19/21 Notes Hashes

Hash?

Problem
- we want to use this to find record by x!

Data structure that can be used to solve the problem?
Expected time in arrays and linked lists:
linear search: looks - y in unsorted array, take O(n)
Binary search:

Hashing:
- idea behind hashing as a way to storage data to accomplish up

Hash table:
- an unordered collection of key-value

Hash function:
- a function that hashes keys to generate indices for each value.

Hash functions: Division based method
- most simple method for hashing is the remainder modulo method
- Formula $h(K) = K \bmod N$

Hash Collision
- occurs when 2 piece of data have the same hash value
- Because $h(x)$ is a hash function
- we have 2 keys K_i and K_j such that

Tackling Collision,
- open Addressing allows elements to move from their preferred.

Linear probing
- move sequentially through the hash table slot until I encounter next empty slot
- for eg a table of size 512 B, if we want to store in

Chaining w/ Linked Lists
- each slot contains a link to a singly linked-list
  - a key-value pairs w/ the same hash

Bloom Filters
- data structure to give you a probabilistic answer.
- Internally structured like a bit array
- 2 operations: add and search/query
- Assume bloom filter on the right has k=3 hash functions
  - pass elements {x, y, z} through 3 hash functions and set the corresponding bits in the bloom filter

Why Bloom Filters?
- Developed by Burton Howard Bloom (1970)
- fast and memory efficient

Scanned with CamScanner

11/22/21 Notes Regular Expressions

What is RegEx

- A series of characters that defines a search pattern
- Used by debuggers, websites, vim, and command line programs

Basic Regex Syntax.

a → matches character itself
. → matches any character
+ → matches one or more occurances of a.
* → matches zero or more occurances of a
? → zero or one occurance of an e-character expression

Basic Regex Syntax continued

[] → any character enclosed
[a-z] → any lower case letter b/t a-z
() → Group expression
{} → Number of times an expression is matched

Theory of finite automata

- Deterministic Finite Automata (DFA)
- Non Deterministic Finite Automata (NFA)
- Push down Automata
- Linear bounded Turing Machine
- Turing Machine

How are automata theory and regular exp.

Deterministic Finite Automata (DFA)
- mathematical model of a machine that accepts a particular set of words over some alphabet
- The set of words is also called the language
- a given string will always produce the same result

Letters and strings
- $a \in \Sigma$, a = letter
- $\chi \in \Sigma^*$, χ = string
- $\lambda \in \Sigma^0$
- λ is a string
- if χ is a string, then $a \cdot \chi$ is a string
- $\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$

Nondeterministic Finite Automata (NFA)
- NFA is a quintuple:
  $\langle \Sigma, S, S_0, \delta, F \rangle$
- Σ => input alphabet
- S => finite non-empty set of states
- S_0 => start state

- State transitions are not uniquely determined by the current state and input

CMP 25

9/24/21

Element Hierarchy

sub object
- Vertex
- Edge
- Face
- mesh → a collection of points, edges, and triangles that make a shape

object

group or complex object

All faces are 2 sided

10/4/21 Notes

Color models
2 models of color:
- additive → all colors add up to white
- subtractive → all colors added ideally produces black

Subtractive => CMYK
Additive => RGB

What influences color in an image?
- color of material
- color and lighting of surroundings
- lighting level
- atmosphere
- add/subtract media
- accuracy of inks
- observers vision

Color gamut
- range of colors that can be shown by a medium
- animal that widest range of colors is what can be perceived by human vision

Graphics
Color depth
- refers to the # of distinct colors an image can contain
  - 2 bit → black/white
  - 8 bit → 256 colors
  - 24 bit - 16.7 mill colors

Naming and organization -

- proper naming is key
- Others may not be working in the same system
- never include spaces
- Python scripts interpret hyphens as a minus operator
- windows reserve these characters

/
\
:
*
?
"
>
<
|

- reliable ways to backup a sequence of words:
  - underscore
  - CamelCase

Asgn 7

```
Node *user_badspeak = bst_create();   Node *user_translation = bst_create();
while (next_word()) {
    if (bf_probe(oldspk)) {
        Node *ht_lookup = ht_lookup(ht, oldspk)
                                    ==
        if (ht_lookup -> oldspeak != NULL && newspeak != NULL) {
            bst_insert(ht_lookup -> oldspk, NULL);
        } user_badspeak
        if (ht_lookup -> oldspeak != NULL && newspeak != NULL) {
            bst_insert(user_translation, ht_lookup -> oldspk, ht_lookup -> newspeak);
        }

if (user_badspeak != NULL && user_translation != NULL) {
    - thoughtcrime_msg  mixspeak_msg
    - bst_print(user_badspeak);
    - bst_print(user_translation);
}
if(user_badspeak != NULL && user_translation == NULL) {
    - thoughtcrime_msg
    - bst_print(user_badspeak);
}
if(user_badspeak == NULL && user_translation != NULL) {
    - use wrong_think_msg
    - bst_print(user_translation);
```

11/29/21 Notes Multi threading

Process & Thread

- What's a process?
  - Code, data and stack
  - Usually has its own address space
- Program state
  - CPU registers
  - Program counter (current location in the code)
  - Stack pointer

Only one process can run on a single CPU core at any given time.

- Multicore CPUs can support multiple processes and threads

A process contains one or more threads.

- Threads within a process can run concurrently.
- Threads share memory w/ each other.
  - They live in the same address space!

Address Space

Program execute code

- Each instruction has an address

Programs access data

- Each byte of data has an address

An address space is the region of the computer's memory where a program executes.

Why use Threads?
- faster to create or destroy than process.
  - No separate address space.
- Allow a single application to do many things at once
  - can keep workers during IO wait.
  - each thread gets to issue its own IO.
  - More IOs can be outstanding / pending.

Context switching:
- expensive b/t process
- less expensive b/t threads.

sharing memory:
- Process don't inherently share memory.
  - Need to use inter-process communication (IPC) to manage shared data.
- Threads share memory, making it easier to share data.

13/1/21 Notes Security

Security environment: threats.

- OS have goals.
  - Confidentiality
  - Integrity
  - Availability.
- Someone attempts to subvert the goals
  - Fun
  - commercial gain

What kinds of intruders are there?

- Casual prying by non technical users
  - Curiosity
- Snooping by insiders.
  - Often motivated by curiosity or money
- Determined attempt to make money
  - May not even be an insider
- Determined attempt to make mischief

- Commercial or military espionage
  - This is very big business

Protection

- Security is about mechanism
  - How to enforce policies
  - Policies largely independent of mechanisms.
- Protection is about specifying policies
- Specifications must be
  - Correct
  - Efficient
  - Easy to use

Protection Domain
- 3 protection Domains
  - each lists objs. w/ permitted operations.

Eugene H
- need help seperating strats

- Use append file
- format print statements.