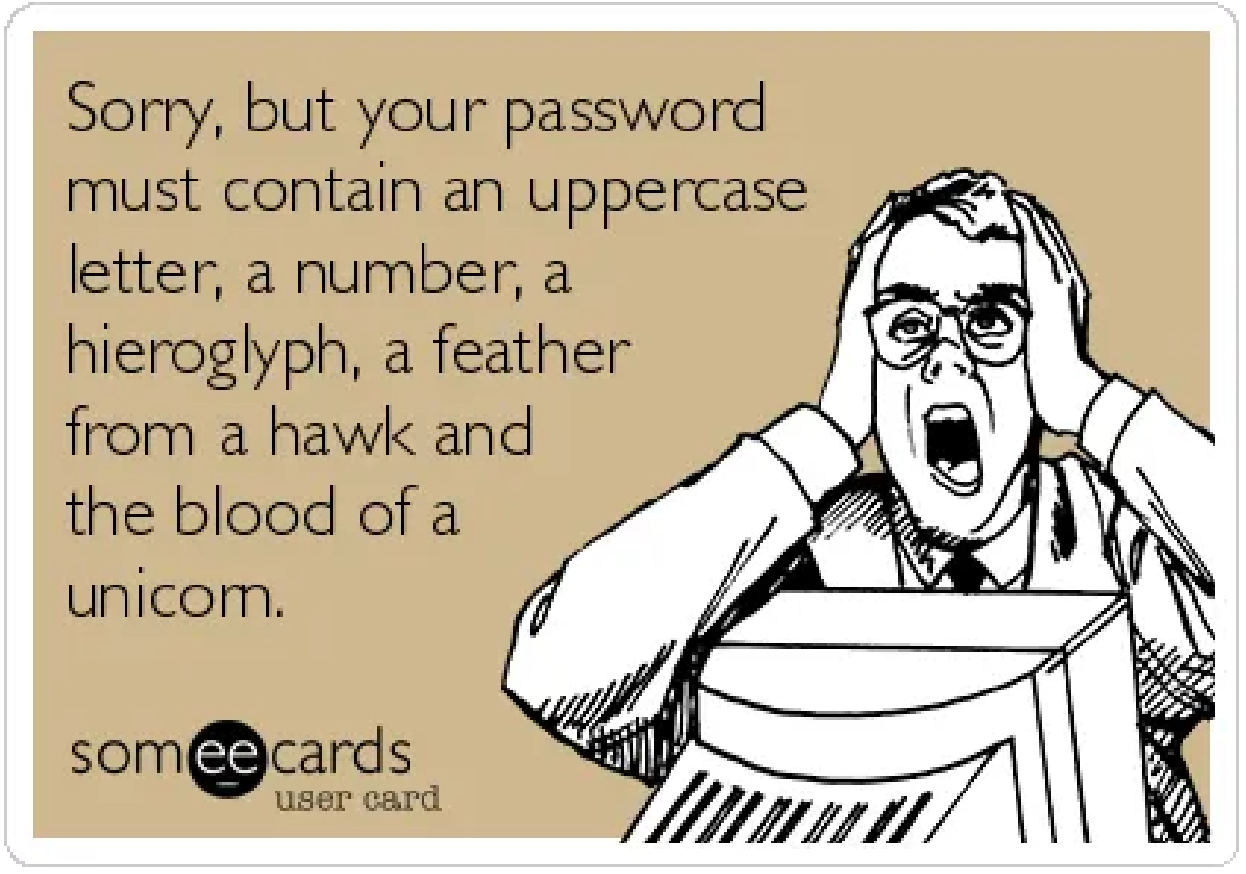
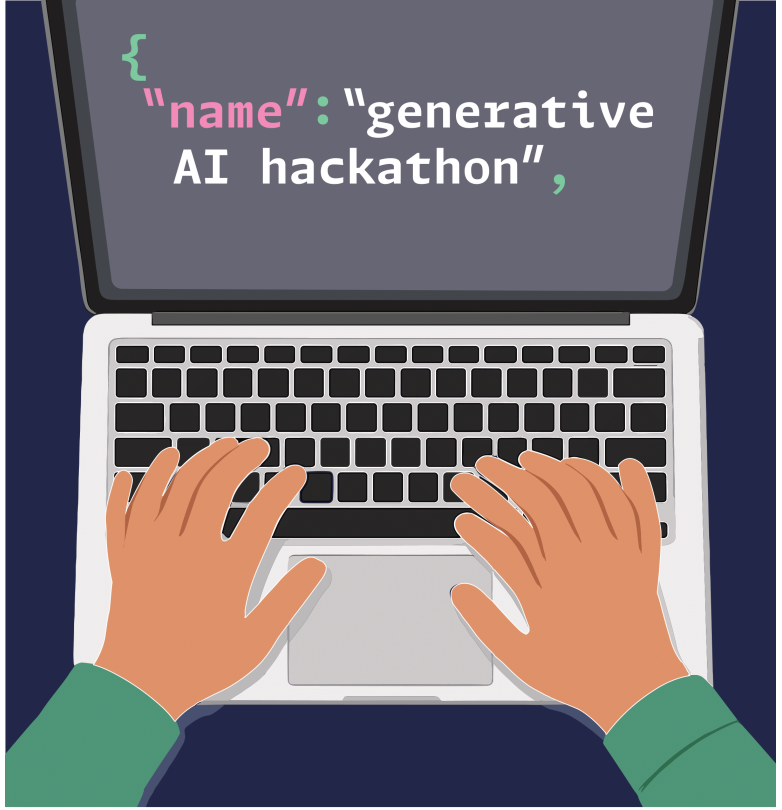
CSCI 3330 Cybersecurity





AI Hackathon @ Bowdoin

October 14, 2025
9:00 a.m.–5:00 p.m.
Barry Mills Hall
Bowdoin College

- Open to all Bates, Bowdoin, and Colby students.
- Individuals or teams of up to 4.
- Breakfast, lunch, snacks, and giveaways provided.

// Sponsored by the Computer Science Department, CXD, and the Hastings Initiative for AI and Humanity

Bowdoin

https://bowdo.in/hackathon

LibreChat

- https://librechat.bowdoin.edu/

Word of the day

DOGFOOD n. Software code not fit for public consumption but good enough for internal purposes, very unrefined and buggy, but containing the basic nutrients. Alternately, code you're developing and using in daily functions simultaneously (a process known as "eating your own dogfood").

Regular files

- `r`: readable
- `w`: writable
- `x`: executable

Directory files

- r: ?
- w: ?
- x: ?

Directory files

- `r`: listable
- `w`: writable
- `x`: traversable (whether you can access files in that directory)

Long paths

- /foo/bar/bletch/baz
- cat /foo/bar/bletch/baz
- /, /foo, /foo/bar/ /foo/bar/bletch must be traversable
- baz must be readable

Long paths

- /foo/bar/bletch/baz

```
$ pwd
/foo/bar
```

- cat bletch/baz
- What must be traversable?
- What must be readable?

Long paths

- /foo/bar/bletch/baz

```
$ pwd
/foo/bar
```

- cat bletch/baz
- /foo/bar, /foo/bar/bletch must be traversable
- baz must be readable

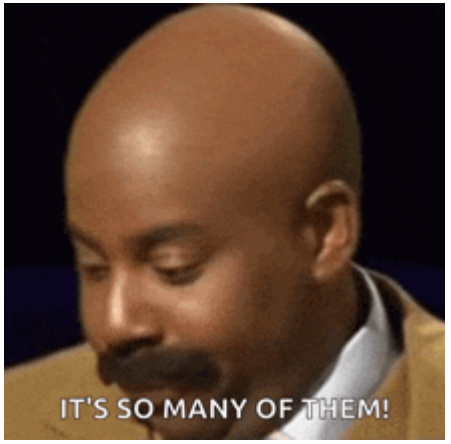
Only a fool asks what is the entropy of a password.

Entropy of a password?

- Passwords have *information content*
- Password generators have *entropy*

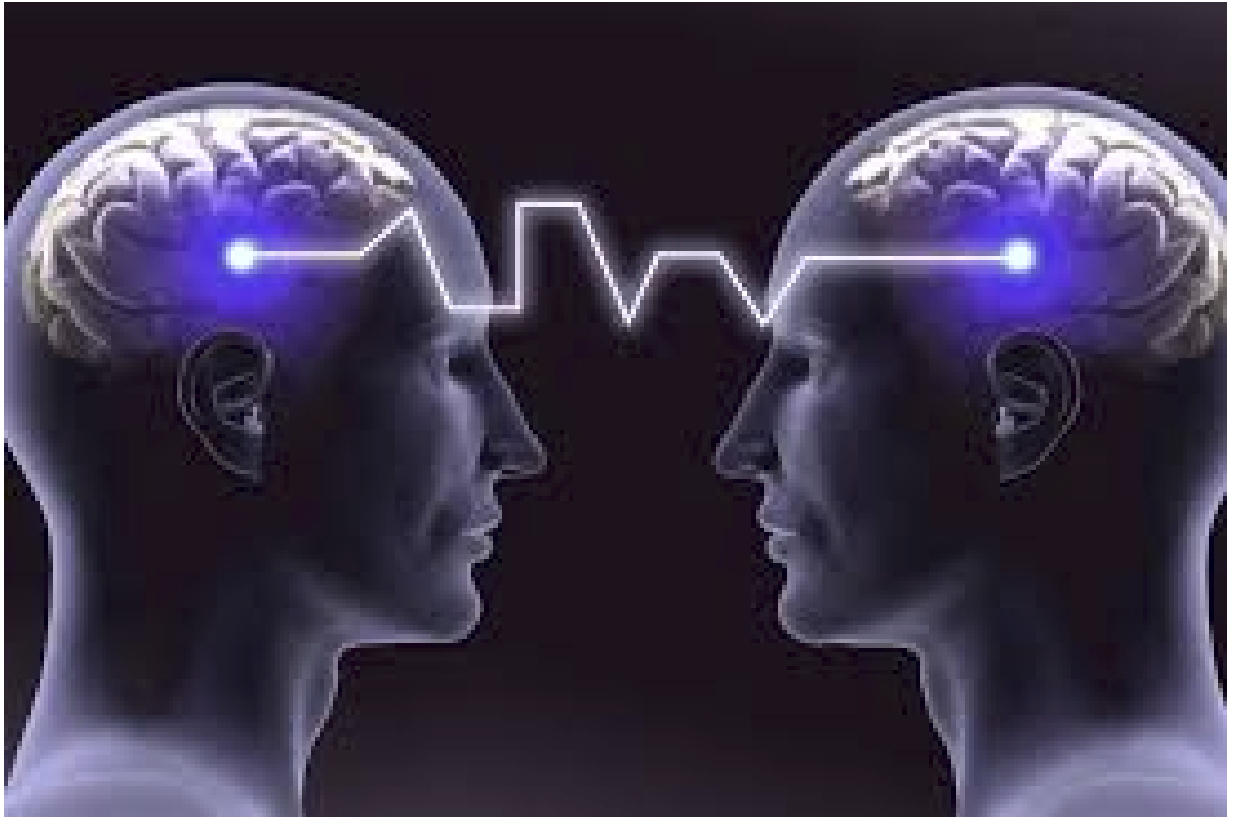
Generating phrases?

- YesterdayInTodayGreaterGood
- So long! Super secure?



Kerckhoffs's principle

- A security system should be secure, even if everything about the system, except the key (or password), is public knowledge.
- In other words: your secret should not be the algorithm

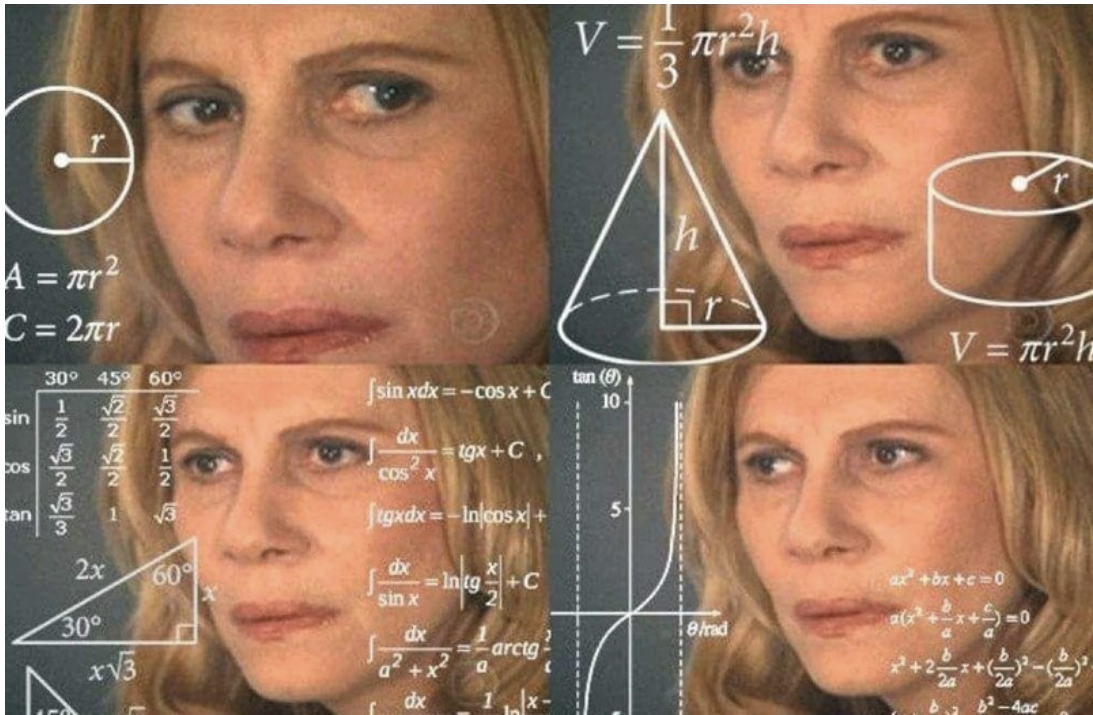


Entropy of a password?

- Assume each word is chosen from dictionary of 2000.
- And there are 5 words in the phrase.
- What is the entropy of this generator?

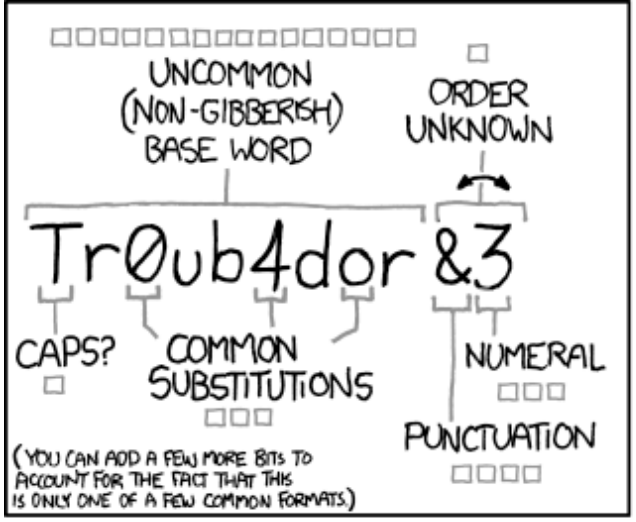
Entropy vs. memorability

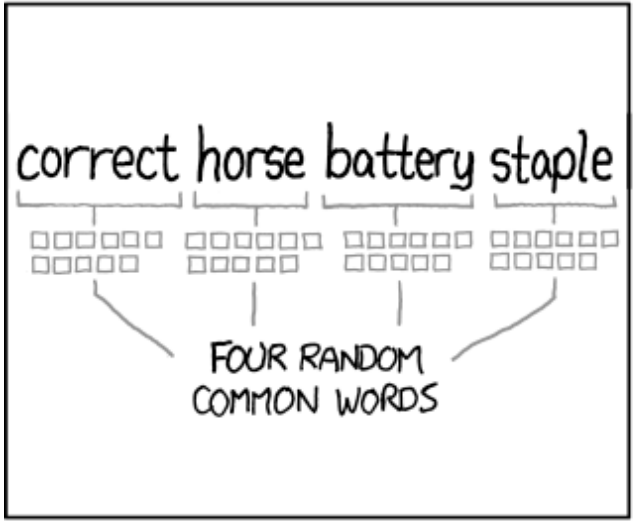
- Entropy unfortunately corresponds roughly to memorability

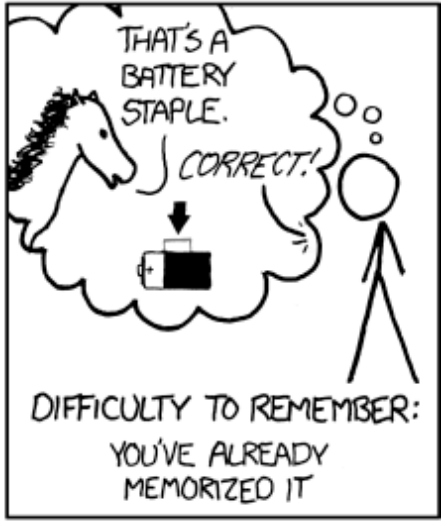


Entropy vs. memorability

- ***BUT*** research shows that phrases can be easier to remember than random letters of equivalent entropy.
- Why?







THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Your Password:

- Must be 8-32 characters long
- Must include at least two of the following elements:
    - At least one letter (upper or lowercase)
    - At least one number
    - At least one special character from the following: # $ % ' ^ , ( ) * + . : | = ? @ /] [_ ` { } \ ! ; - ~
- Must be different than your previous five Passwords
- Must not match your User ID
- Must not include more than 2 identical characters (for example: 111 or aaa)
- Must not include more than 2 consecutive characters (for example: 123 or abc)
- Must not use the name of the financial institution (for example: JPM, MORGAN, CHASE)
- Must not be a commonly used password (for example: password1)

Randomness

Is the universe random?

- Is the universe non-deterministic?
- Probably

Randomness

- Unpredictability
- Unpredictability is *relative*

https://youtu.be/zbkizy-Y3qw?t=56

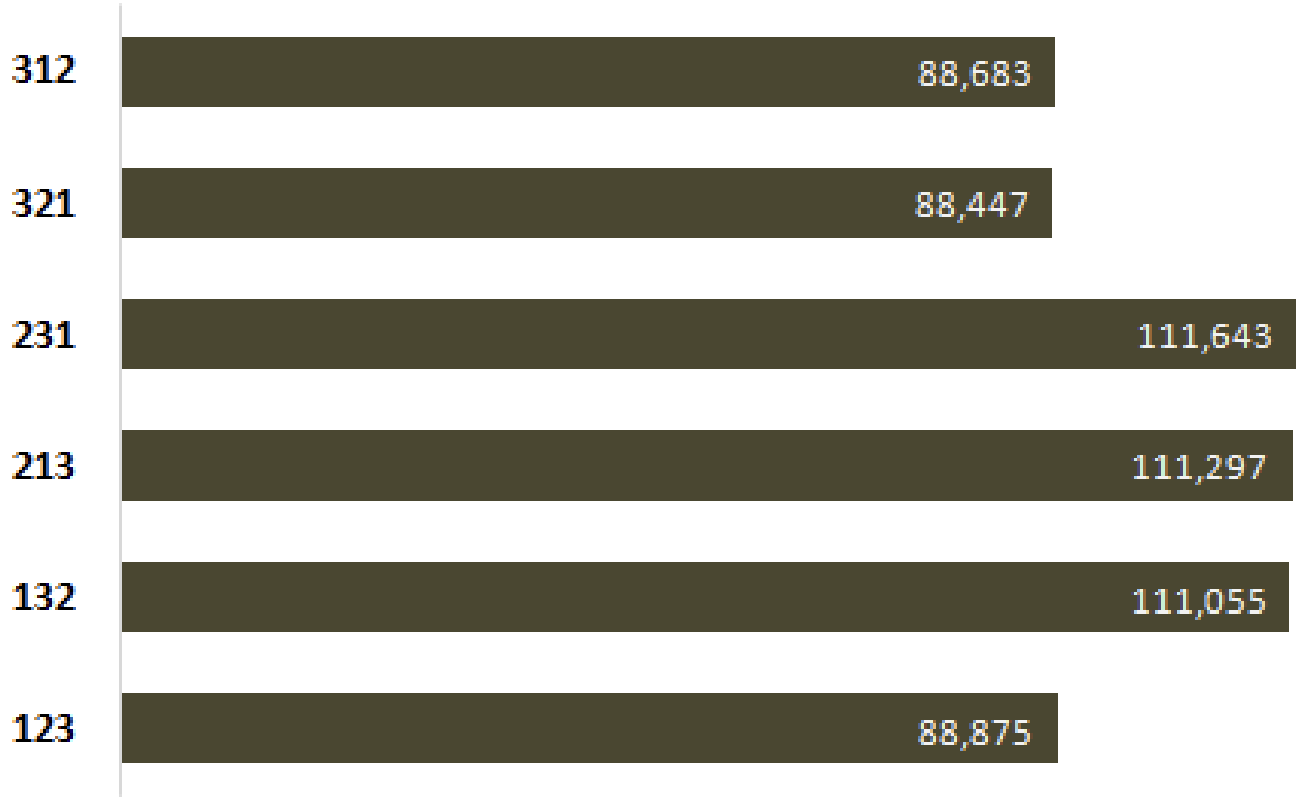
Computers are deterministic

- How can we generate random numbers?
- Pseudorandom – appearing statistically random but derived deterministically
- We derive random bits and pivot to the rest

High level example

- shuffle(): randomize the order of an array
- How hard can it be?

```
for (int i = 0; i < cards_length; ++i) {
    int r = random() % cards_length;
    swap(&cards[i], &cards[r]);
}
```



Source:
https://blog.codinghorror.com/the-danger-of-naivete/

Moral of the story

- Randomness is hard!
- Use Fisher-Yates shuffle
- OR a popular, well-tested library function

PRNG generators

- Seed – initializes the hidden PRNG state
- Ah but how do we generate that?
- “Real” randomness


ARROGANT


IBM

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilise the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key D467BE99B852085C marked as ultimately trusted
gpg: revocation certificate stored as '/home/desktop/.gnupg/openpgp-revocs.d/C53
public and secret key created and signed.

pub   rsa3072 2019-06-27 [SC] [expires: 2021-06-26]
      C5307C7223CBEE06DCB21777D467BE99B852085C
uid                      This is an example key <fake@notreal.com>
sub   rsa3072 2019-06-27 [E] [expires: 2021-06-26]

desktop@vbpc:~$
```

How to access entropy pool?

“In UNIX, everything is a file.”

- /dev/random – pure entropy pool
- /dev/urandom – seeded with entropy pool

Common, ***insecure*** sources

- The current time in seconds
- The current time in nanoseconds
- The current process ID
- The current thread ID
- The current operating system uptime
- Some XOR combination of the above

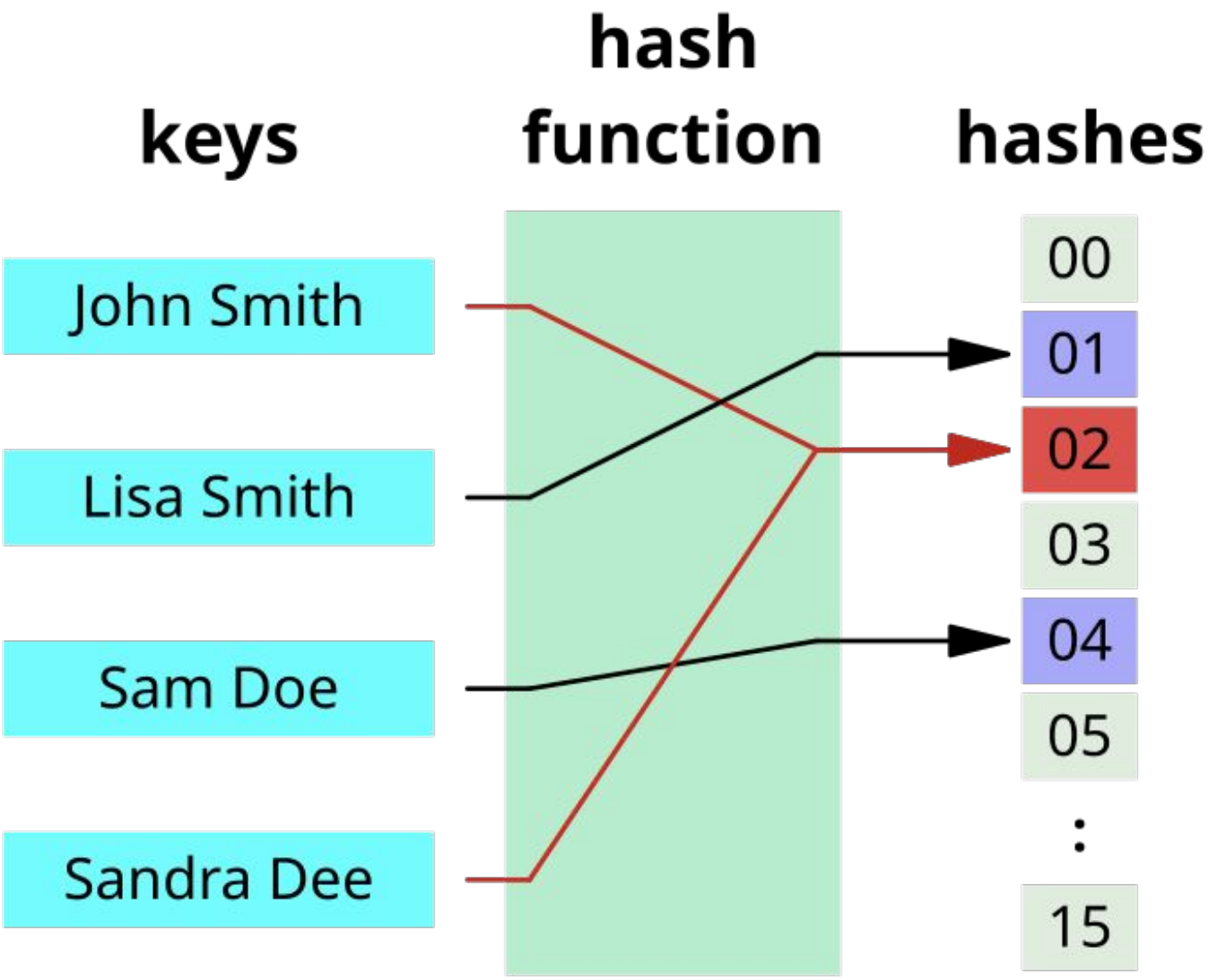
Hash

We have seen hashes…

- Used to “store” passwords
- Used to mix entropy into the entropy pool

What is a hash?

- Hash function is any function that can be used to map data of arbitrary size to fixed-size values.
- The fixed-size value is called a hash.


keys
hash function
hashes
John Smith
Lisa Smith
Sam Doe
Sandra Dee
00
01
02
03
04
05
:
15

Java’s hashCode(): nothing cryptic

- Integer.valueOf(1).hashCode() → 1
- Integer.valueOf(2).hashCode() → 2
- Long.valueOf(0x100000000L).hashCode() → 1

Cryptographic hashes

- One way / irreversible
- Collision resistant

Pre-image resistance

- Given *h*, it should be infeasible to find a message *m* such that *h* = hash(*m*).

Second pre-image resistance

- Given m_1, it should be infeasible to find an m_2 such that hash(m_1) = hash(m_2).

Collision resistance

- It should be infeasible to find two messages m_1 and m_2 such that hash(m_1) = hash(m_2).

Java’s hashCode(): nothing cryptic

- Integer.valueOf(1).hashCode() → 1
- Integer.valueOf(2).hashCode() → 2
- Long.valueOf(0x100000000L).hashCode() → 1

Java’s hashCode(): nothing cryptic

- Integer.valueOf(1).hashCode() → 1
- Integer.valueOf(2).hashCode() → 2
- Long.valueOf(0x100000000L).hashCode() → 1

- Given *h*, it should be infeasible to find a message *m* such that *h* = hash(*m*).

Java’s hashCode(): nothing cryptic

- Integer.valueOf(1).hashCode() → 1
- Integer.valueOf(2).hashCode() → 2
- Long.valueOf(0x100000000L).hashCode() → 1

- Given m_1, it should be infeasible to find an m_2 such that hash(m_1) = hash(m_2).

Java’s hashCode(): nothing cryptic

- Integer.valueOf(1).hashCode() → 1
- Integer.valueOf(2).hashCode() → 2
- Long.valueOf(0x100000000L).hashCode() → 1

- It should be infeasible to find two messages m_1 and m_2 such that hash(m_1) = hash(m_2).

Implications

- Satisfying collision resistance → Satisfying second pre-image resistance
- Failing second pre-image resitance → Failing collision resistance

Popular cryptographic hash f()s

- MD5 (1991)
- SHA-1 (1995)
- SHA-2 (2001)
- SHA-3 (2016)
- BLAKE (2008), BLAKE2 (2012), BLAKE3 (2020)

NOT cryptographic hash f()s

- Object.hashCode() (Java)
- hash() (python)
- CRC32 (or, more broadly, checksums)
- Jenkins Hash

Some hashes are slow on purpose!

- bcrypt
- Argon2

Most languages built-in

- python: hashlib
- java: java.security.MessageDigest

Problem

- What if two users have the same password?

Salt

- Store ***NON***-secret string with password
- Concatenate when hashing
- hash($salt . $password)

Salt

- Protects against accidental collisions
  - Birthday paradox!
  - In a group of just 23 people, there’s a greater than 50% chance that at least two of them share the same birthday
- Protects against rainbow table lookups

When does birthday paradox apply?

- Hash collisions?
- Brute-forcing passwords?
- Buying a lottery ticket?
- Two people buying the same lottery ticket?
- Two people buying the winning ticket?

Assignment 1

- Deadline: September 18, 2025, 10:00pm
- Permissions, entropy, randomness
- https://tildesites.bowdoin.edu/~j.knockel/csci3330f25/assignment1.pdf