Authentication in practice

CS 1660: Introduction to Computer Systems Security

Authetication (Recap)

	Hash (SHA1 SHA256)	MAC Message Authentication Code	Digital signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Crypto system	None	Symmetric (AES)	Asymmetric (RSA, ECC)

How to authenticate *humans*?

Beyond CIA…

Identification

Authentication

Authorization

Identification

- Humans are generally indistinguishable in front of a computer
- A subject should provide a identifier (e.g. email has to be unique)
- The system will verify if you have the proof to claim an identity
  - This process is called Authentication

Authentication

- Authentication is the act of confirming the truth of an attribute of a datum or entity
- There are three authentication factors:
  - Knowledge: Something you know
  - Ownership: Something you have
  - Inherence: Something you are

Knowledge

- Something the user knows (e.g., a password, or PIN, challenge/response (the user must answer a question), pattern)

Strengths

- Easy to transport
- Can be changed

Weaknesses

- Can be forgotten
- Easy to duplicate
- Verifier often learns the secret

Ownership

- Something the user has (e.g., phone number, ID card, security token, etc.)

Strengths

- Easily transferable
- More difficult to clone that what you know

Weaknesses

- Can be lost or stolen
- Can be forged
  - e.g. a key can be made from photos

Inherence

- Something the user is or does (e.g., fingerprint, retinal pattern, DNA sequence, voice, etc.).

Strengths

- Non-transferable
- Usually identifies individual

Weaknesses

- Forgeable (ie, fingerprint from picture or from a glass)
- Can be lost (ie, loss or degradation)
- Can’t be changed

Dynamic Authentication with Devices

Time-Varying Codes

(One Time Password)

- Physical/Tamper proof
- Precise clock
- Hash chain inside

Challenge response

- U2F (Universal Second Factor) authentication protocol
- challenge/response protocol
- https://developers.yubico.com/U2F/Protocol_details/Overview.html

More authentication factors?

- Location factor
  - Where you are (ie. Gps, Mobile Cell, etc.)
- Ability factor
  - What you can do (ie. Keystroke Dynamics, mouse tracking, etc.)
- …
- Usually they are classified in the inherence factor, It is an open problem
  - NIST SP 800-63-1

If you need more security?

- Could you use more authentication factor to verify the identity of a user?
  - **Multi Factor Authentication** is born
    - To increase the level of security, many systems will require a user to provide different types of authentication factor

      2-factor authentication

      - ATM card + PIN
      - Credit card + signature
      - Passport + fingerprint
      - …

Clicker Question 1

Which multi-factor schema is more secure?

A. ATM Card + Pin + Fingerprint
B. Passport + fingerprint + face
C. Same level of security
D. It is not possible to establish

Clicker Question 1 - Answer

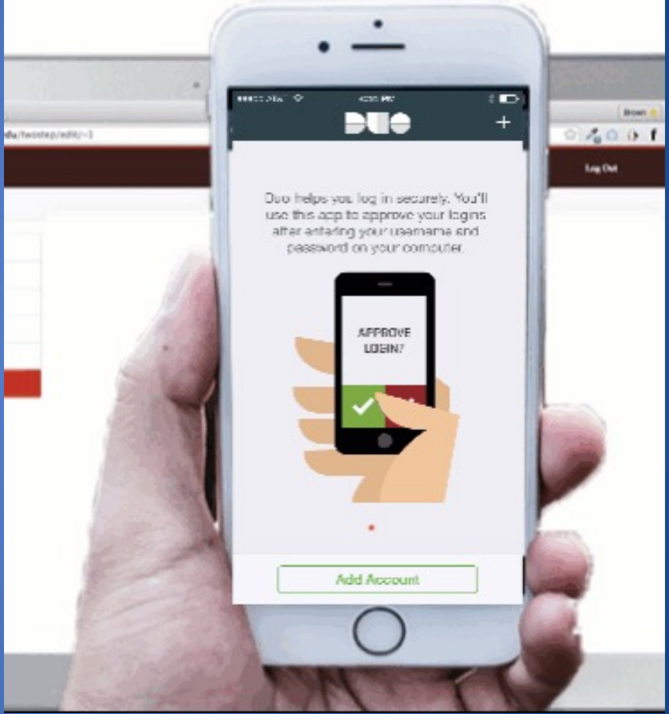
- Answer: A
  - It is the only three-factor authentication schema

  B. fingerprint and face
  - they belong both to the ownership authentication factor
  - C, D are not true

Multi-step Authentication

Federal Financial Institutions Examination Council

- User submits two or more authentication tokens
- ATM bank card (Two Factor)
  - –Physical card (something you have)
  - –PIN (something you know)
- Password + code sent to the phone (Two Step) Brown Authentication
  - –Enter password (something you know)
  - –Enter code (something you know)



Security Questions (aka Personal Questions)

- Questions about the user like city of birth, high school, first car, favorite color, etc.

- Used as supplementary authentication factor
  - When user logs in from new device
  - For password reset

- Answer selection strategies
  - Truthful answers
  - Untruthful but plausible answers
  - Randomly generated answers

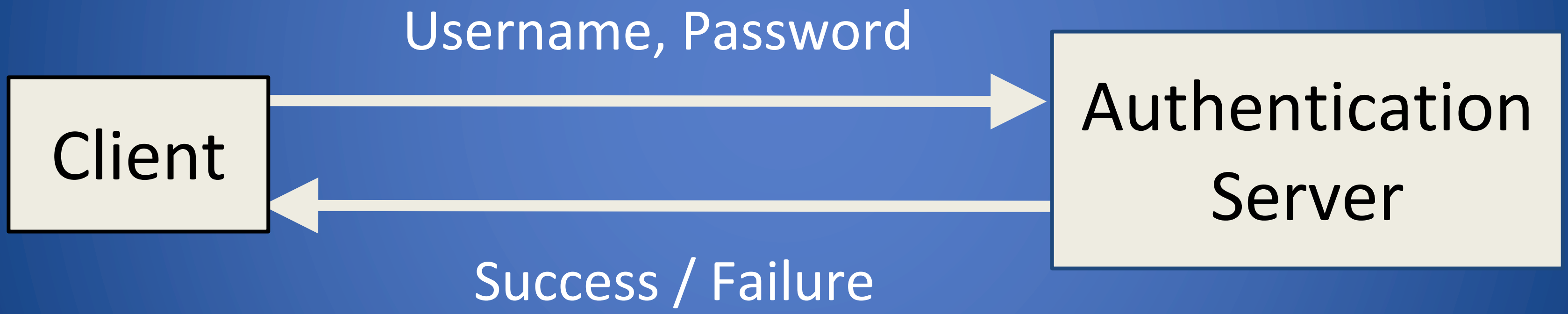
Password Authentication

What Do These Passwords Have in Common?

- 123456
- password
- 123456789
- 12345678
- 12345
- 111111
- 1234567q
- sunshine
- qwerty
- iloveyou
- princess
- admin
- welcome
- 666666
- abc123
- football
- 123123
- monkey
- 654321
- !@#$%^&*
- charlie
- aa123456
- donald
- Password1
- qwerty123

Top 25 passwords used in 2018 according to SplashData

Password Authentication



Attacks on Passwords

Information States (MC CUMBER CUBE):

Storage,Transmission, Processing

Why send the password?

- “Why not use a MAC instead?”
- Usually, establish a secure channel first with other cryptography
- Allows the server to decide how to hash passwords (which is important for later)

Password Complexity

Characters in Passwords

- Consider a standard US English keyboard
- Lower case characters: 26
- UPPER and lower case: 52
- Digits: 10
- Special characters: 32
- Standard keyboard characters: 94
- All 7-bit ASCII characters: 128

Size of Password Space

- 6-character password
- Only lower case letters, no numbers or symbols

Char	*	*	*	*	*	*

Size of Password Space

- 6-character password
- Only lower case letters, no numbers or symbols
- There are 26 x 26 … x 26 or 26^6 (≈ 309M) possible passwords
- Easy to try all of them

Char	*	*	*	*	*	*
Choices	26	26	26	26	26	26

Other Password Schemes

- How many possible 6-character passwords?
  - Digits (10): 10^6
  - UPPER and lower case (52): 52^6
  - Special characters: &, %, $, @, “, |, ^, <, … (32): 32^6
  - Standard keyboard characters (94): 94^6
  - All 7-bit ASCII characters (128): 128^6

Number of Possible Passwords

Assume a standard keyboard with 94 characters

Password length	Number of passwords
5	94^5 = 7,339,040,224
6	94^6 = 689,869,781,056
7	94^7 = 64,847,759,419,264
8	94^8 = 6,095,689,385,410,816
9	94^9 = 572,994,802,228,616,704

Brown University Password Policy

- Cannot contain your first name, last name, or username
- Cannot match your last three passwords
- Must be at least 10 characters in length
- Must contain at least one lowercase character
- Must contain at least one number
- Must contain at least one special character
- Must contain at least one uppercase character

Source: https://it.brown.edu/information-security/guard-your-privacy/strong-passwords

Strong Passwords

- Long passwords preferred
- Use all available characters
  - UPPER/lower case characters
  - Digits
  - Special characters: &, %, $, £, “, |, ^, §, …
- Which of the following passwords are strong?
  - Seattle1
  - M1ke03
  - P@$$w0rd
  - TD2k5s@}ecV87^R:@DKlksj298RLO<j;-*h

Clicker Question 2

Which password policy is more secure?

Policy A	Policy B
8 characters total: • 1 lowercase • 1 uppercase • 1 digit • 1 symbol • 4 of any keyboard characters	8 of any keyboard characters

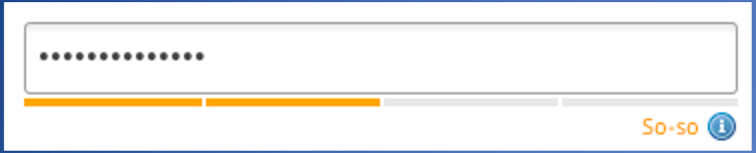
A. Policy A
B. Policy B
C. Both are equally secure
D. It is not possible to evaluate the security

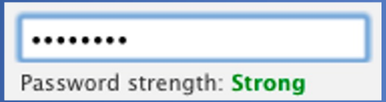
Clicker Question 2 - Answer

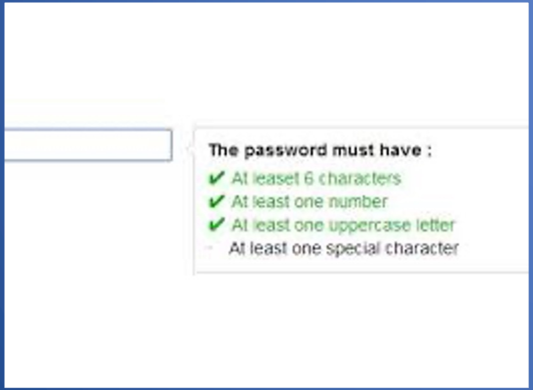
- Answer: A, B, or C
  - Depends on the assumptions you make
    - Policy B has bigger password space, though users can pick bad passwords (i.e. `password`, `12345678`)
    - What if both policies prevented users from picking common passwords?

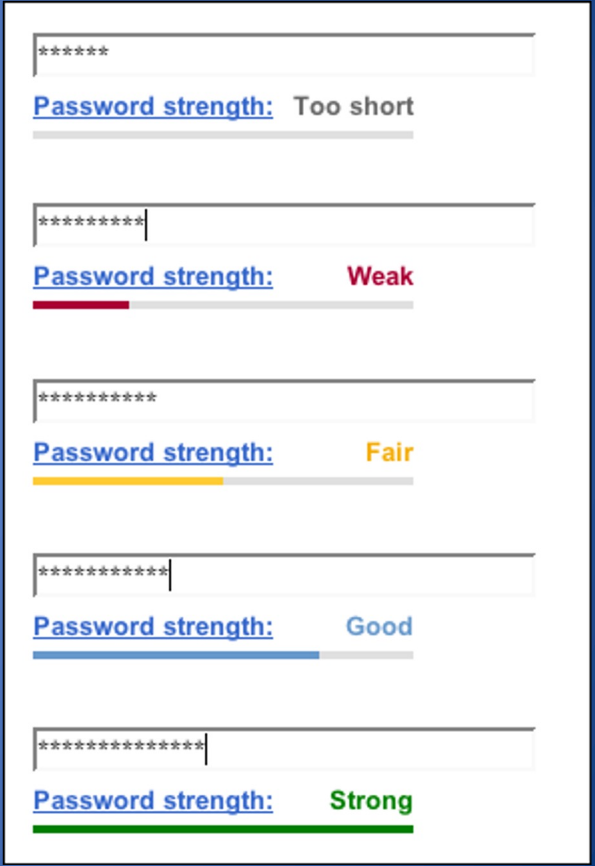
Password Complexity In Practice

Password Strength Meter









History of LUDS

- **L**ower- and **U**ppercase letters, **D**igits and **S**ymbols
- Became standard ~ 1985
- US Defense Department - *Password Management Guideline(Green Book)*
    - **Guessing space**
- NIST - *Password Usage of the Federal Information Processing Standards*
    - Take English words(dictionary) into consideration

LUDS in Practice

passwordmeter.com

Test Your Password		Minimum Requirements
Password:		• Minimum 8 characters in length • Contains 3/4 of the following items: - Uppercase Letters - Lowercase Letters - Numbers - Symbols
Hide:	☑	
Score:	0%	
Complexity:	Too Short	

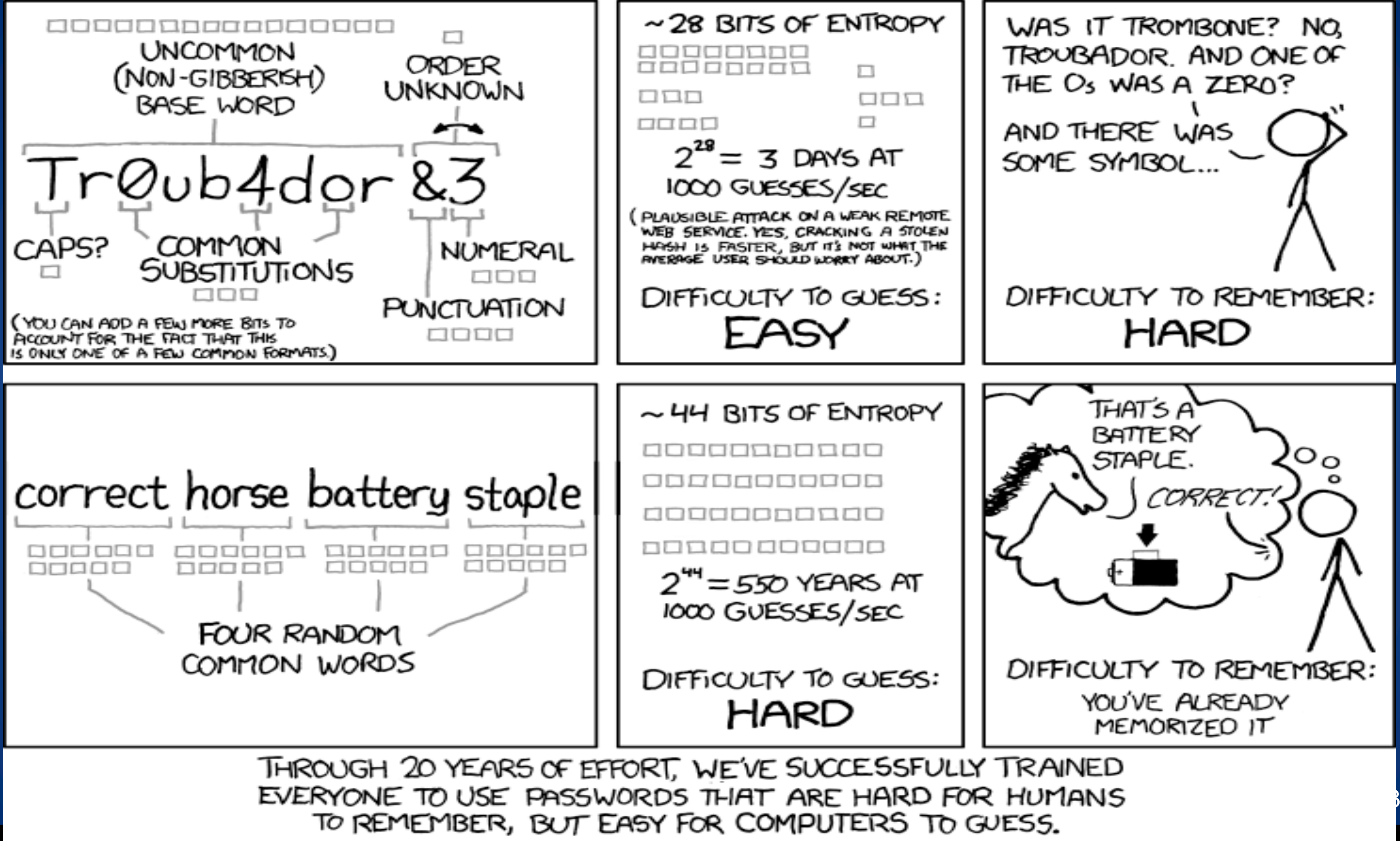
Additions	Type	Rate	Count	Bonus
Number of Characters	Flat	+(n*4)	0	0
Uppercase Letters	Cond/Incr	+((len-n)*2)	0	0
Lowercase Letters	Cond/Incr	+((len-n)*2)	0	0
Numbers	Cond	+(n*4)	0	0
Symbols	Flat	+(n*6)	0	0
Middle Numbers or Symbols	Flat	+(n*2)	0	0
Requirements	Flat	+(n*2)	0	0
Deductions				
Letters Only	Flat	-n	0	0
Numbers Only	Flat	-n	0	0
Repeat Characters (Case Insensitive)	Comp	-	0	0

Problems with LUDS

- Characters frequency is not random
- Frequently used words (password, name)
- Special Dates (Birthday of a relative)
- Keyboard Patterns
- Wrong password strength estimation
    - P@$$w0rd1

zxcvbn: realistic password strength estimation

- A model developed by Dropbox
- Easy to adopt
- A rigorous estimation
  - Estimating guess attack directly
- Non-probabilistic
  - Assume attackers know the patterns that make up a password


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor&3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT
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.

What about AI?

- Lakera is an AI security company that has developed an online game
- In this game, you have to try to get the young wizard Gandalf to reveal the password using natural language questions
- Each level requires increasingly complex techniques to deceive the implemented protection mechanisms

LAKERA

gandalf.lakera.ai

Practicing

- You can practice in class with something similar to your real password:
  - **passwordmeter.com** (LUDS)
  - zxcvbn alternatives to bit.ly/2dp7BD3 (no more public on dropbox from 9/1/2017):
    - **goo.gl/DUSwys** (Cygnius)
    - **goo.gl/epu13n**(Takecontrolbooks)

BREAK!

5 › 4 › 3 › 2 › 1

Storing Passwords

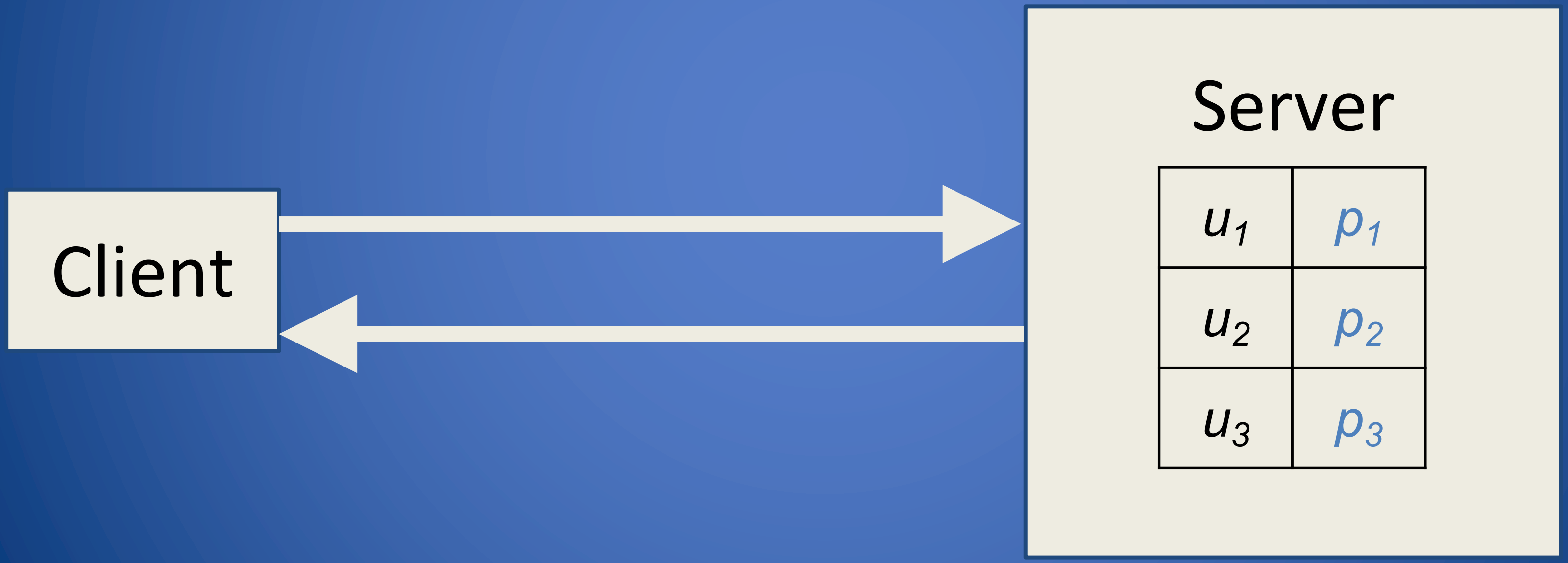
How Should the Server Store Passwords?

- Our goal is to defend from attacks that exfiltrate the password database stored by the server
  - Most common password-related attack on server
- We don’t consider other password attacks on the server
  - Eavesdropping passwords submitted by users
  - Modifying the password authentication code

Security mindset: Trust models

- Trust describes a situation in which the security of a system depends on the decisions made by other systems outside of its control
- Security of system A depends on decisions made by system B
  - "A trusts B"
- Trust ≠ trustworthiness

Attempt #1 - Plaintext

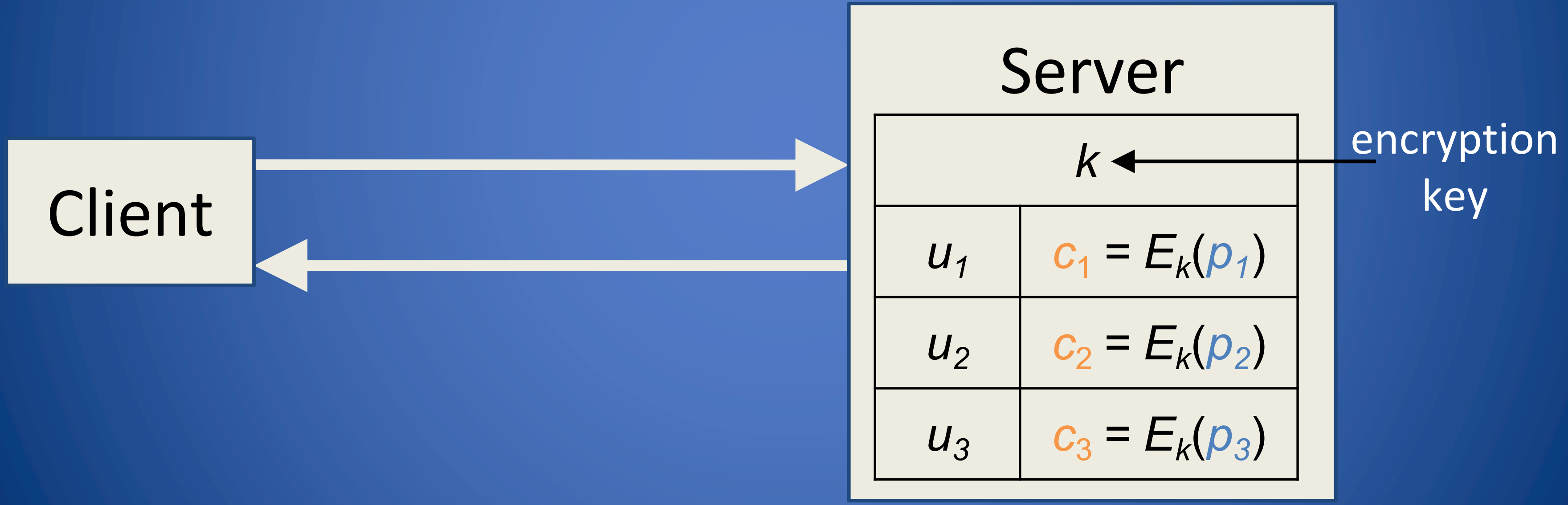


Attempt #1 - Plaintext

- Advantages
  - Easier to manage
  - Less computational needs
- What could go wrong?
  - Ifdatabase is stolen, so are passwords!
  - Admins have access to passwords.
- Ex. Reddit (2006), Twitter (2018)

Attempt #2 - Encryption

- Store encrypted passwords
- Decrypt and compare on login

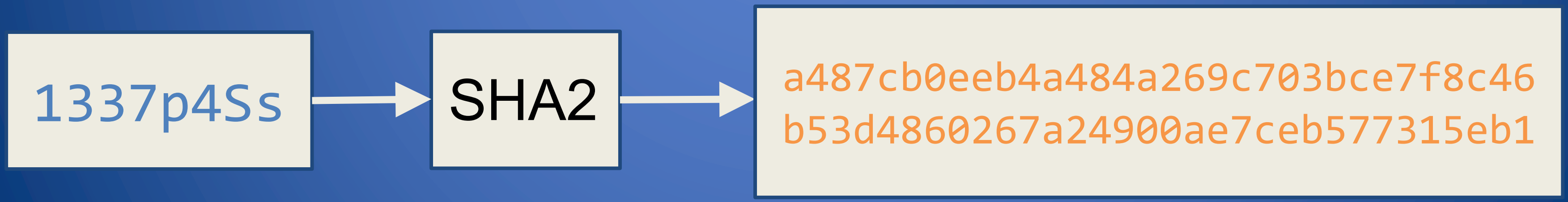


Attempt #2 - Encryption

- Advantages
  - If encrypted passwords are stolen, they can't be decrypted
  - Only administrators with key can decrypt
- What could go wrong?
  - If the encrypted passwords are stolen, what is to keep the key from also being stolen?
  - Anyone with the key (admins) can view passwords
- Ex. Adobe (2013)

Attempt #3 - Hashing

Example:

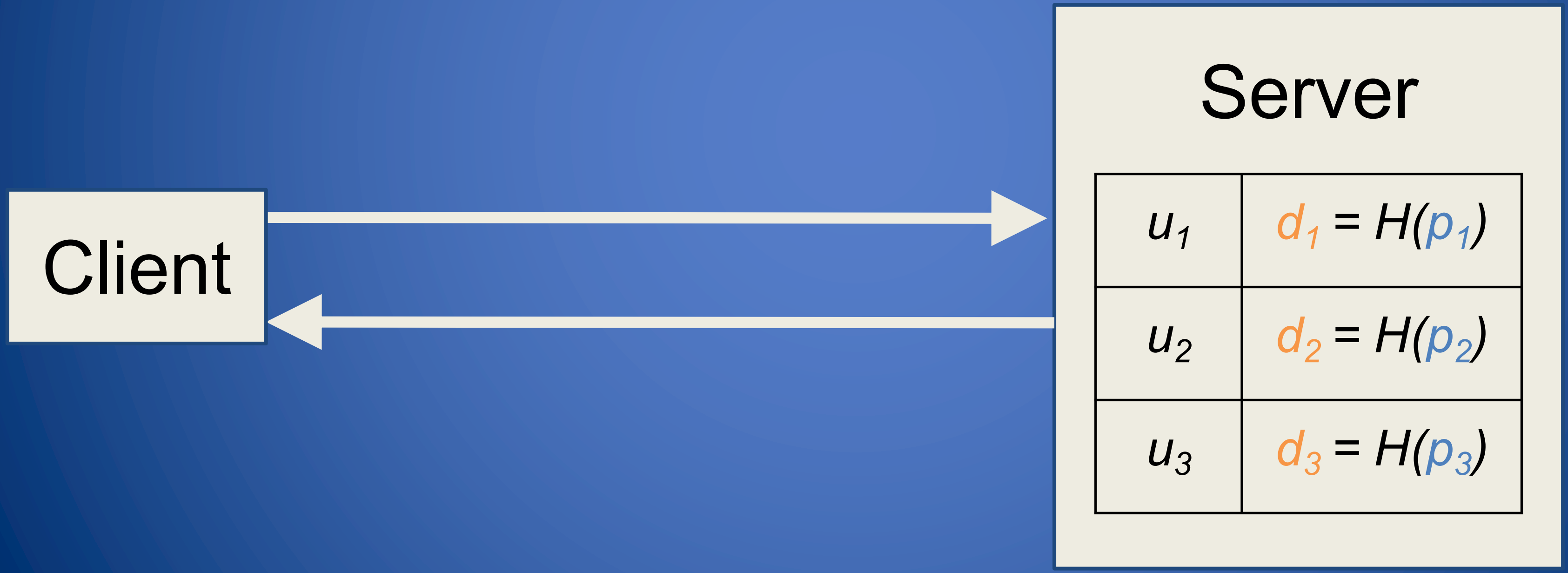


Attempt #3 - Hashing

- Recall cryptographic hashing:
  - Variable length input, fixed length “random” output
  - One-way
    - Given hash *x*, hard to find *p* such that H(*p*) = *x*
  - Weak collision resistance
    - Given input *p*, hard to find *q* such that H(*p*) = H(*q*)
  - Strong collision resistance
    - Hard to find distinct *p, q* such that H(*p*) = H(*q*)

Attempt #3 - Hashing

- Hash the password, store the hash
- Hash the user-supplied password and compare



Attempt #3 - Hashing

- Registration
  - Hash password, store hash
- Login
  - Hash user-supplied password, compare with stored hash
- What advantages does this scheme have?
  - If database is stolen, hashes need to be cracked
    - Correct
  - Cracking must be done brute-force for every password
    - Is this accurate?

Clicker Question 3

- Which hash will you try to decode first?

fa80328eaf40ecbf22943747d8fe63e3
b57bc9ee094db754ca74e034875deb1d
35e89a946952203816 1a3c173815d8e7
d6ed2c6957eb5d5228be0942cf93ea72
35e89a9469522038161a3c173815d8e7

A. fa80328eaf40ecbf22943747d8fe63e3
B. b57bc9ee094db754ca74e034875deb1d
C. 35e89a9469522038161a3c173815d8e7
D. d6ed2c6957eb5d5228be0942cf93ea72

Clicker Question 3 - Answer

C. fa80328eaf40ecbf22943747d8fe63e3

Identical hash from different users come from the same password and probably it is a common word

Attempt #3 - Hashing

- What could go wrong?
  - Identical passwords produce identical hashes
  - Once you've cracked a given hash, you can trivially crack it every time you see the same hash again
  - Humans pick bad passwords
    - Frequency analysis
    - Precompute massive tables for popular hash functions
    - Common passwords are very common!
    - Even a small table cracks most passwords
    - Updated version

Clicker Question 4

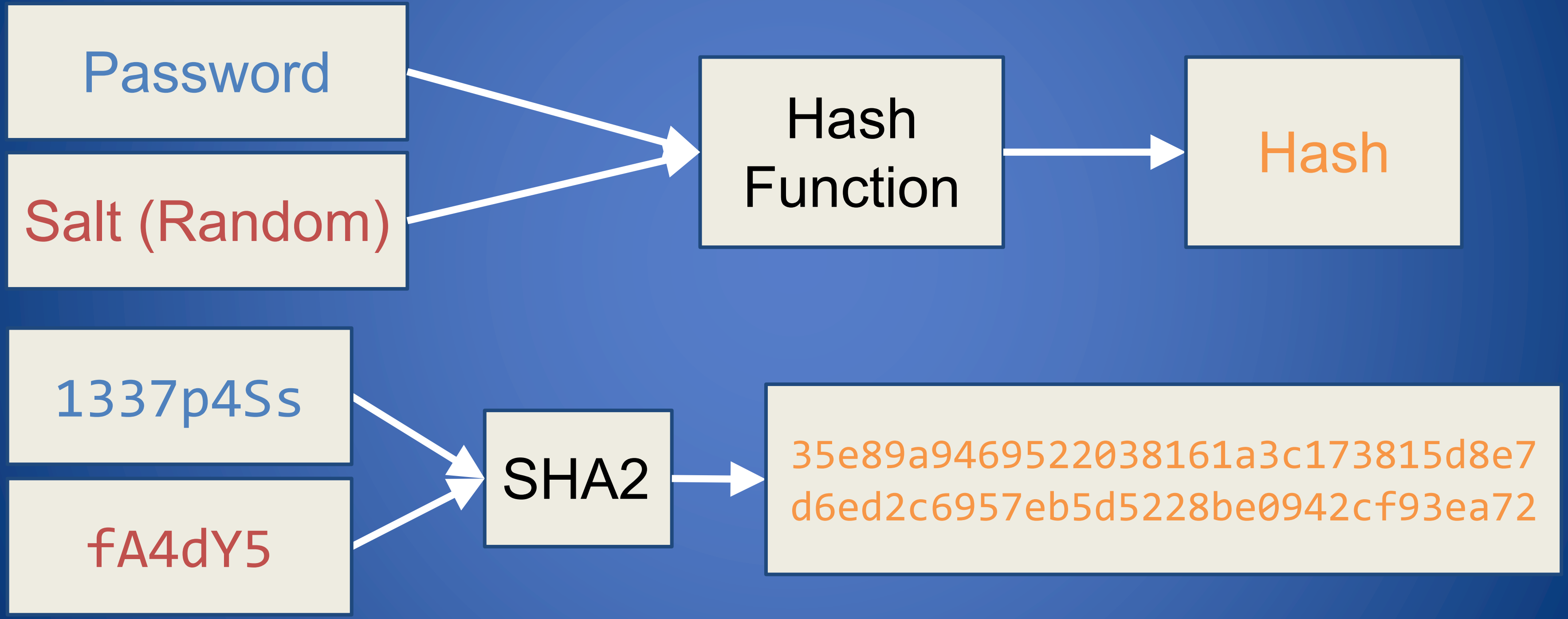
Mallory steals a database of encrypted passwords (but cannot steal the key). Could she recover the plaintext passwords?

A. Yes, all of them
B. Yes, a fraction of them
C. No, since the database is encrypted
D. No, since it is computationally infeasible

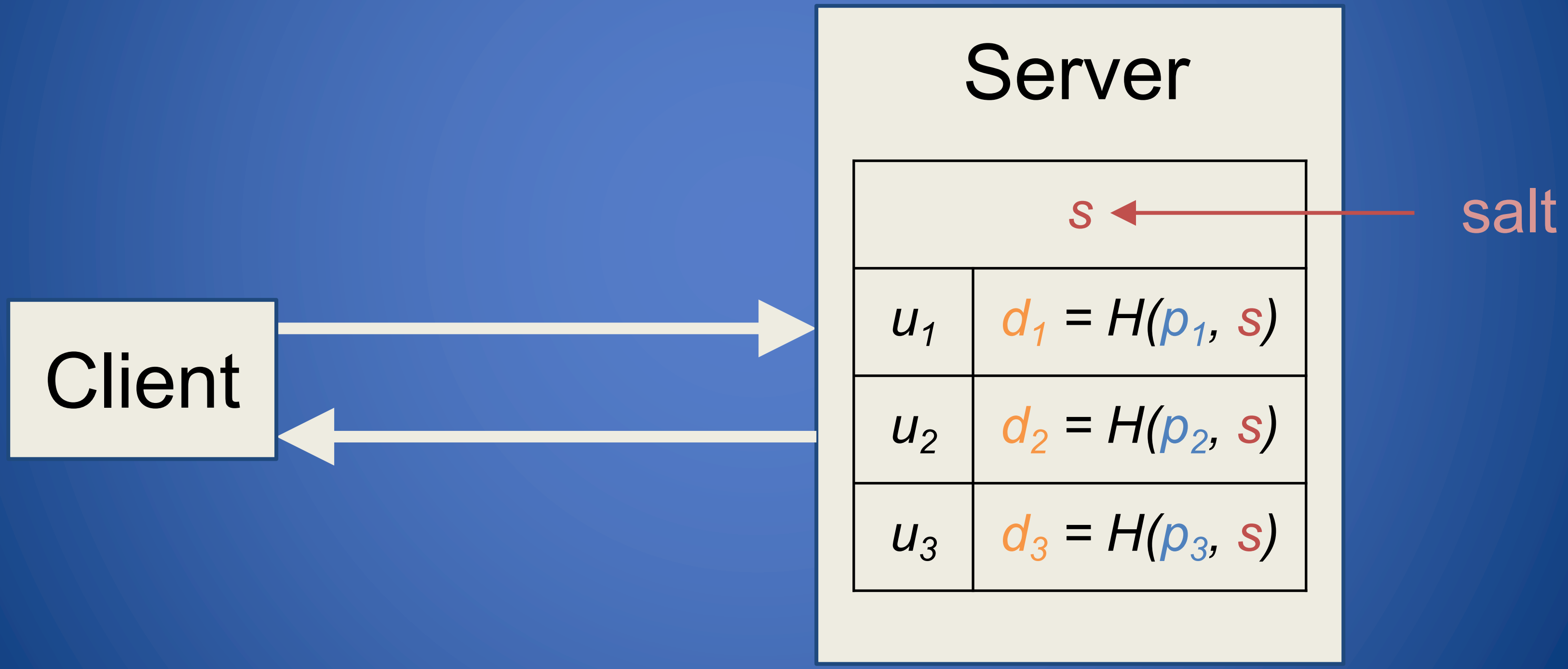
Clicker Question 4 - Answer

- Answer: B
  - Identical passwords produce identical ciphertexts
  - If you know one password, you know all passwords same ciphertext
  - Humans pick bad passwords and hints
    - Frequency analysis (0.5% of users use `password`)
    - Password hints (e.g., numbers 123456)
  - Unique passwords with good hints are safe

Attempt #4 - Salting



Attempt #4 - Salting



Attempt #4 - Salting

- Store hash of salted password
- Hash the password and salt, then compare
- Advantages
  - In order to precompute, need password and salt
  - Since salts are random, guessing salt is useless
  - Even if salt is known, computation must be redone for every site

Clicker Question 5

Using hashing and salting to store passwords, the server successfully defends over frequency analysis attacks.

A. True.
B. False.
C. Not enough information.

Clicker Question 5 - Answer

Using hashing and salting to store passwords, the server successfully defends over frequency analysis attacks.

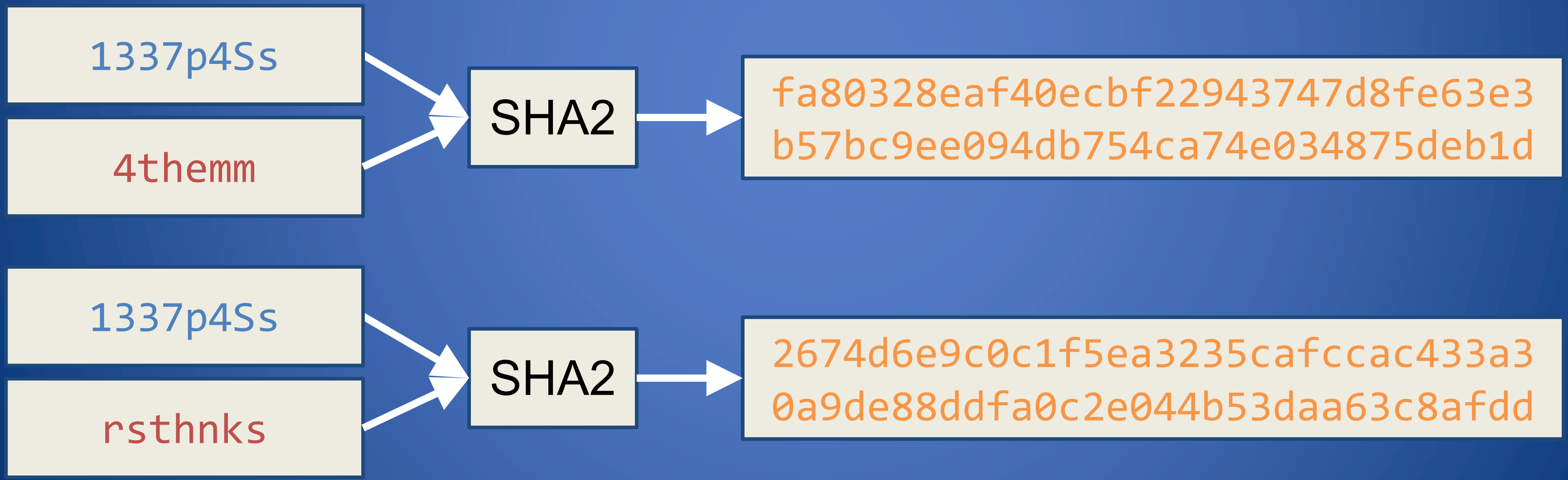
A. True.
B. False.
C. Not enough information.

Attempt #4 - Salting

- What could go wrong?
  - Identical passwords and identical salts produce identical hashes
  - Humans pick bad passwords --> frequency analysis
    - If you crack one password, you crack all identical ones
    - For big sites, precomputation is worth it

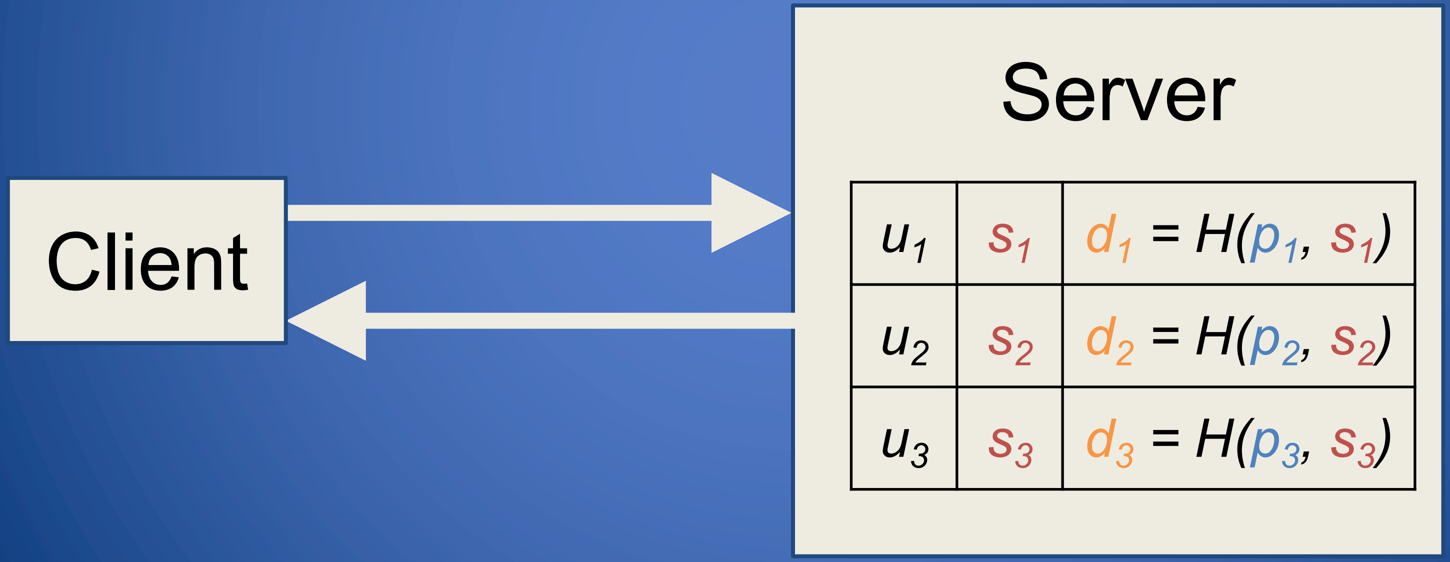
Attempt #5 – Per-User Salting

- Hashing same password with different salt will produce different hashes



Attempt #5 - Per-User Salting

- Generate a salt, hash the password, store salt and hash
- Hash the given password with the user's salt and compare



Attempt #5 - Per-User Salting

- Generate a salt, hash the password, store the hash
- Hash the given password with the user's salt and compare
- Advantages
  - Since every user has different salt, identical passwords will not have identical hashes
  - No frequency analysis
  - No using known passwords to crack other passwords
  - No precomputation, hence much harder to crack

Summary

- Human Authentication
- AAA Identification, Authentication, Authorization, Accounting
- Password
  - Authentication
  - Complexity
  - Storage (salting)