CSE467: Computer Security

21. Authentication

Seongil Wi

Department of Computer Science and Engineering

HW2

- Score will be released soon!

We will take a Quiz in Next Week

- Date: 11/28 (TUE.), Class time
- Scope:
  - Access Control
  - Authentication
- O/X quiz (3~4 problems) + some computation quiz

Security Properties

- **C**onfidentiality
- **I**ntegrity
- **A**vailability

+ **Authentication**: the ability of a computer system to *confirm the sender's identity*

+ **Non-repudiation**: the ability of a computer system to *confirm that the sender can not deny about something sent*

Today's Topic

- Confidentiality
- Integrity
- Availability

+ **Authentication**: the ability of a computer system to *confirm the sender's identity*

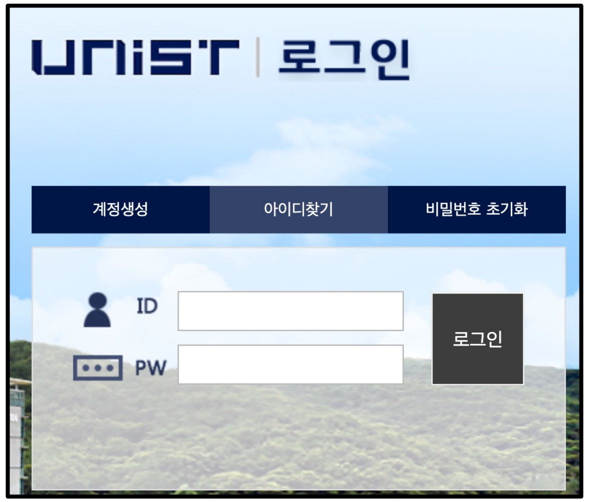
+ **Non-repudiation**: the ability of a computer system to *confirm that the sender can not deny about something sent*

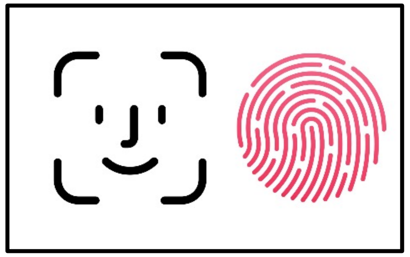
Authentication – Who Are You?

- The process by which the identity of someone or something
- Where it is used?
  - A person recognizing a person
  - Access control (PC, ATM, mobile phone)
  - Physical access control (house, building, area)
  - Identification (passport, driving license)

Authentication Methods

- Typical method
  - **Knowledge**: Something you know
    - Password, PIN, …
  - **Token**: Something you have
    - ID card, key, passport, certificate
  - **Biometrics**: Something you are
    - A physiological characteristic (e.g., fingerprint, iris pattern, form of hand)
    - A behavioral characteristic (e.g., the way you sign, the way you speak)





Outline

- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

Outline

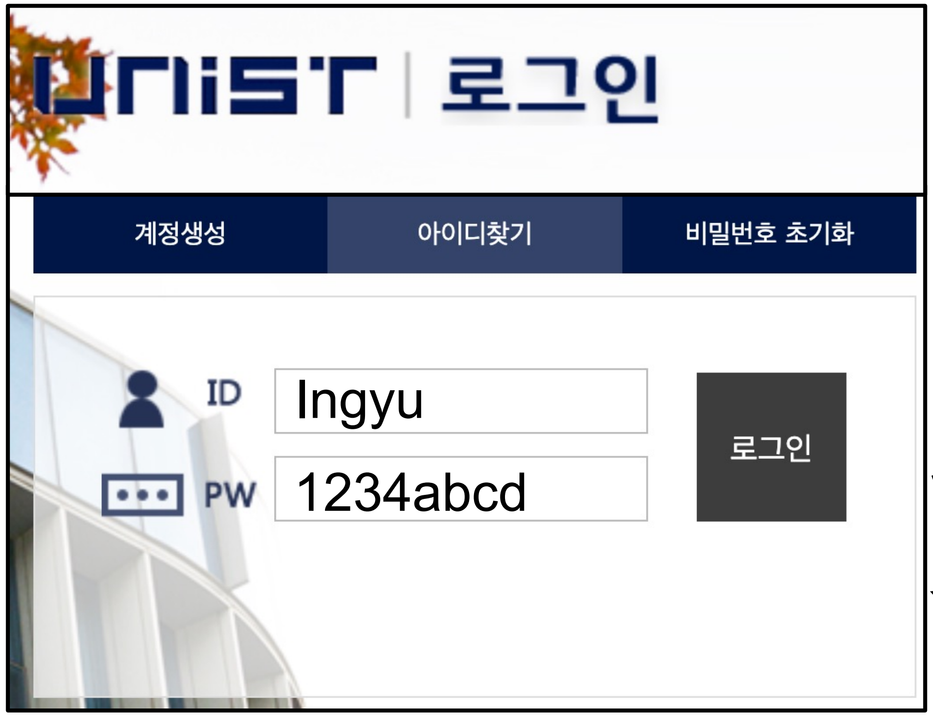
- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

Password-based Authentication – Something You Know

- User has a secret password
- System checks it to authenticate the user

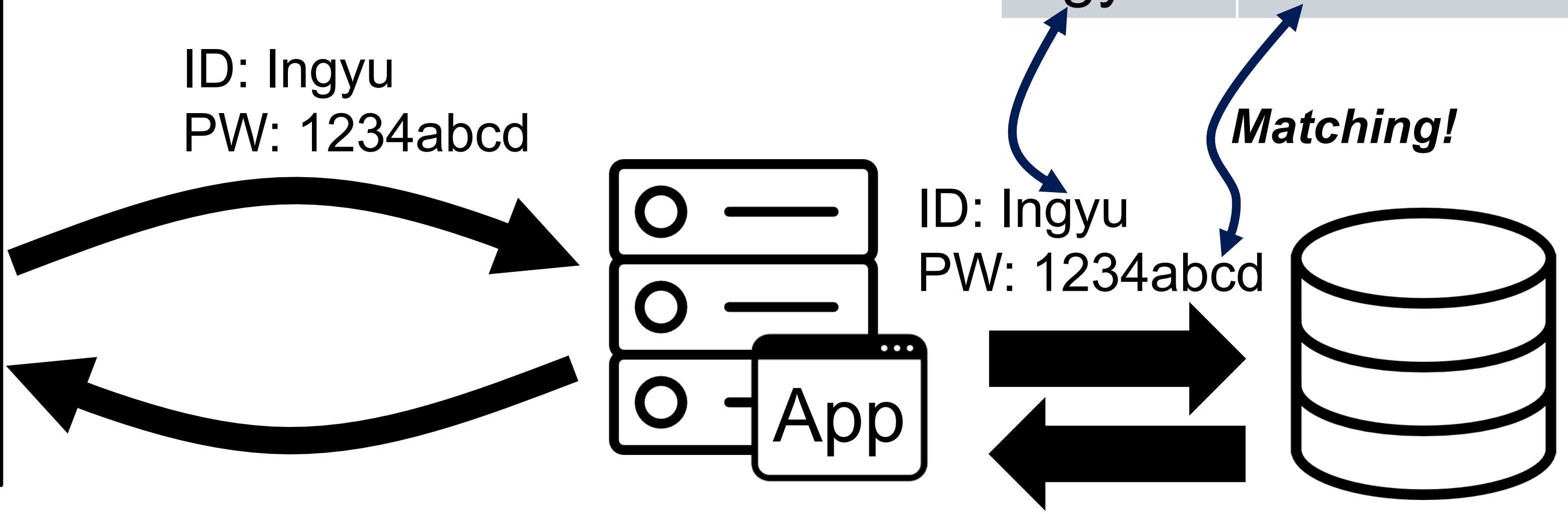
Clear Text Password

ID	Password
Jaewoo	1234abcd
Yinae	verysecure
Ingyu	1234abcd



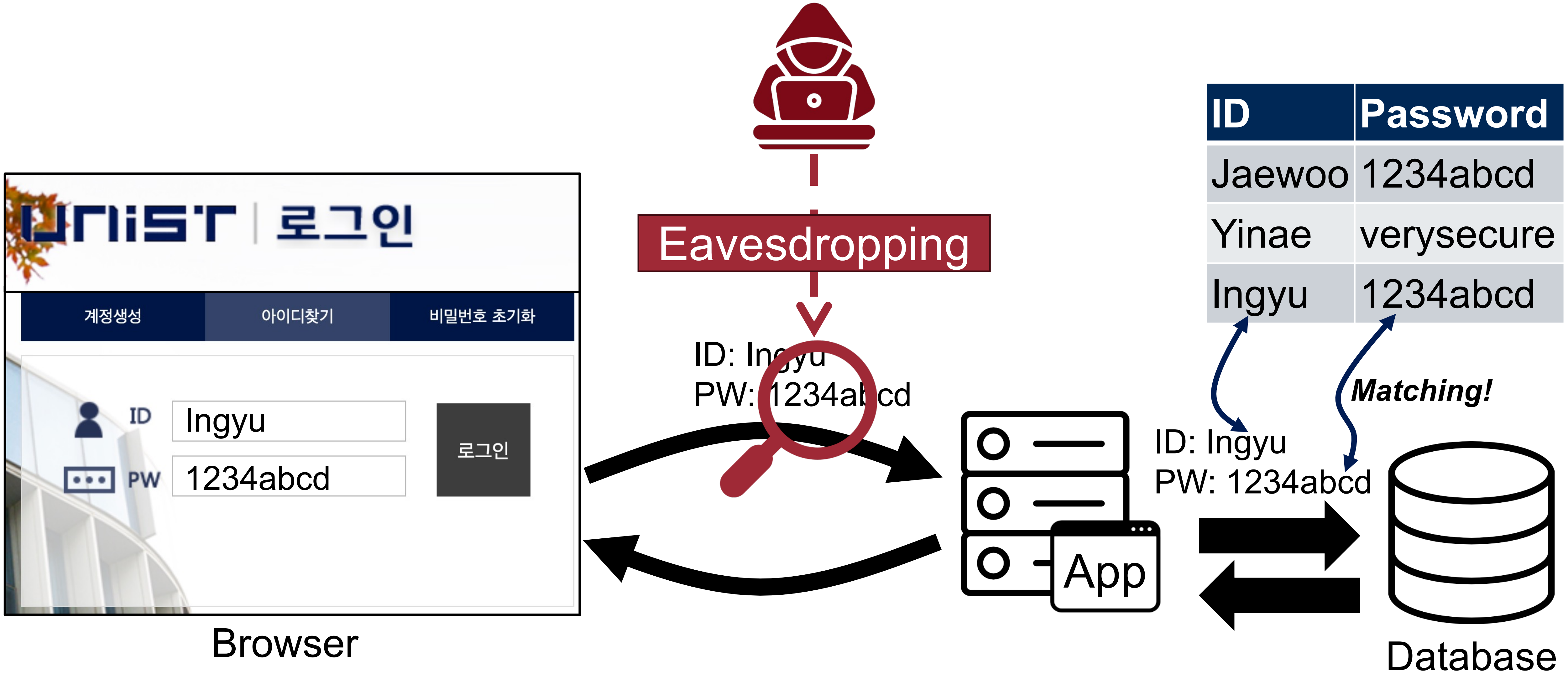
ID: Ingyu
PW: 1234abcd

Browser



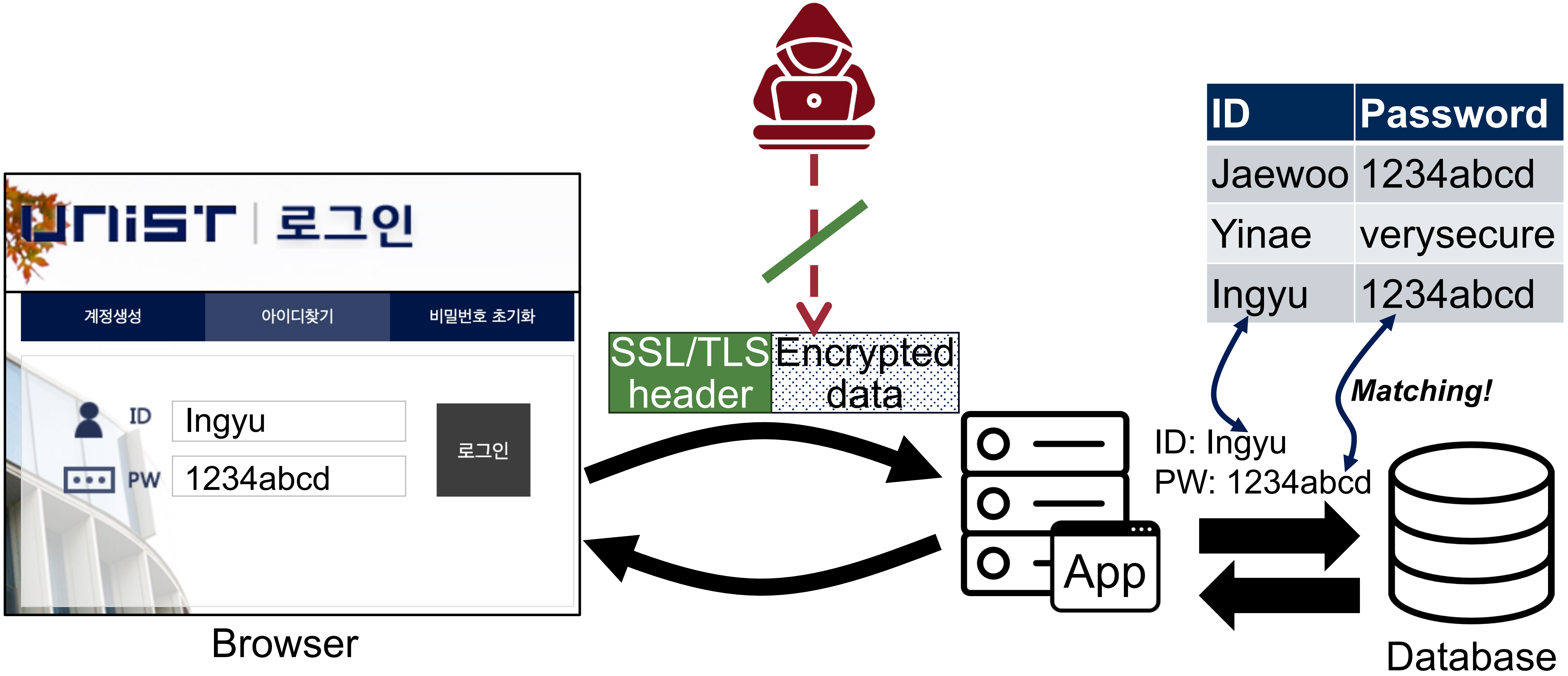
Database

Problems of Clear Text Password?



ID	Password
Jaewoo	1234abcd
Yinae	verysecure
Ingyu	1234abcd

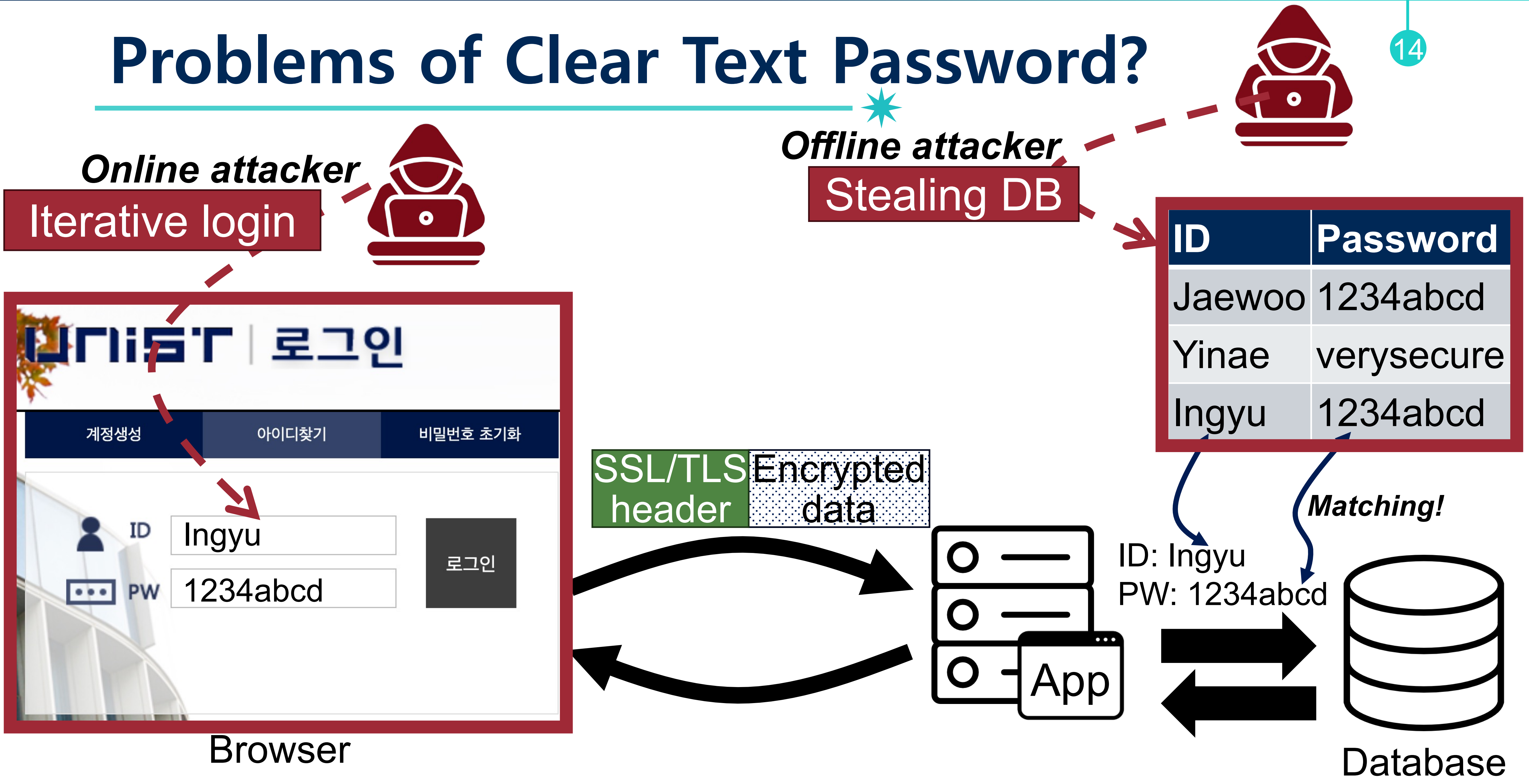
SSL/TLS Encryption! Are We Safe Now?



ID: Ingyu
PW: 1234abcd

ID	Password
Jaewoo	1234abcd
Yinae	verysecure
Ingyu	1234abcd

Problems of Clear Text Password?



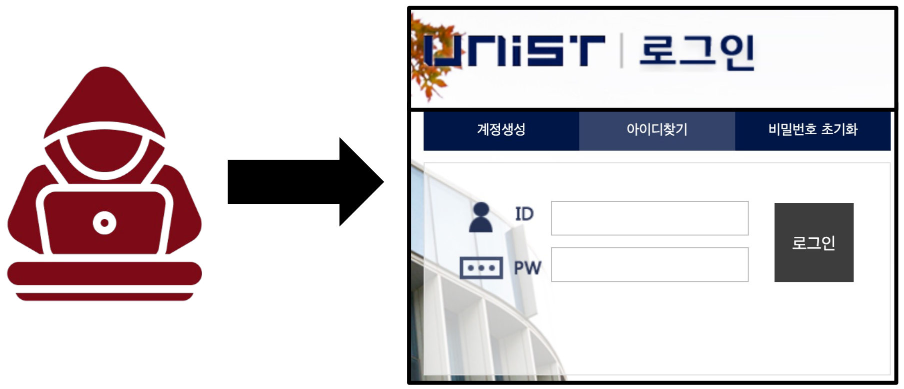
ID: Ingyu
PW: 1234abcd

ID	Password
Jaewoo	1234abcd
Yinae	verysecure
Ingyu	1234abcd

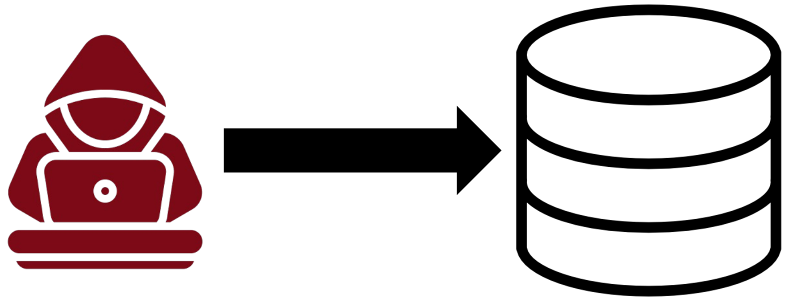
ID: Ingyu
PW: 1234abcd

Attackers

- What is the threat model?
  - ***Online attacker***
    - Tries to login to a service by iteratively trying passwords and looking whether he was successful



  - ***Offline attacker***
    - Stole password database and tries to recover the passwords
      - ✓ If the password is stored in clear text, an offline attacker can know the password of every user



How Do Attackers Use Passwords?

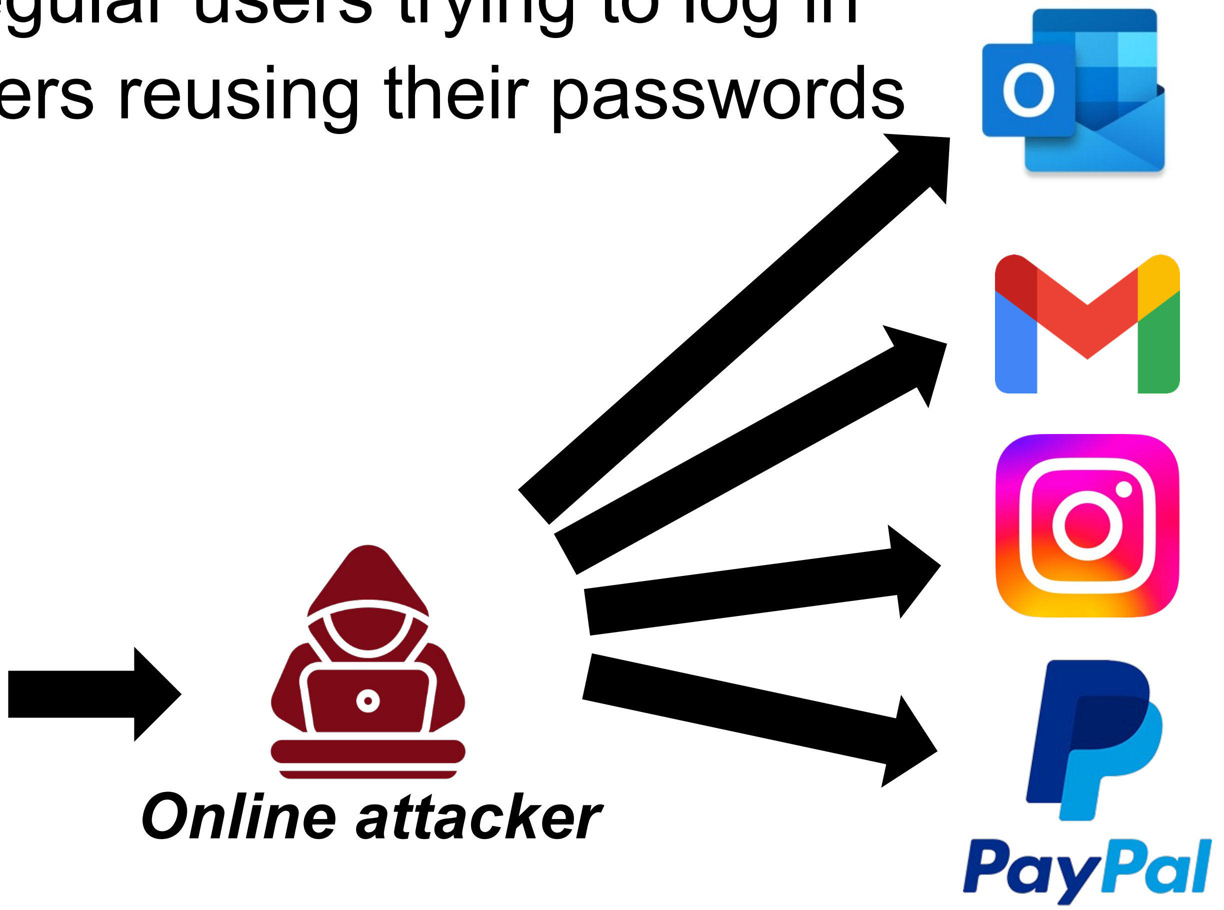
- Once a database of credentials is leaked, attackers can use them in multiple ways
  - Extract emails and usernames
  - Learn what are the most common passwords that most users use
  - Learn what are the passwords that specific users use

ID	Password
Jaewoo	1234abcd
Yinae	verysecure
Ingyu	1234abcd

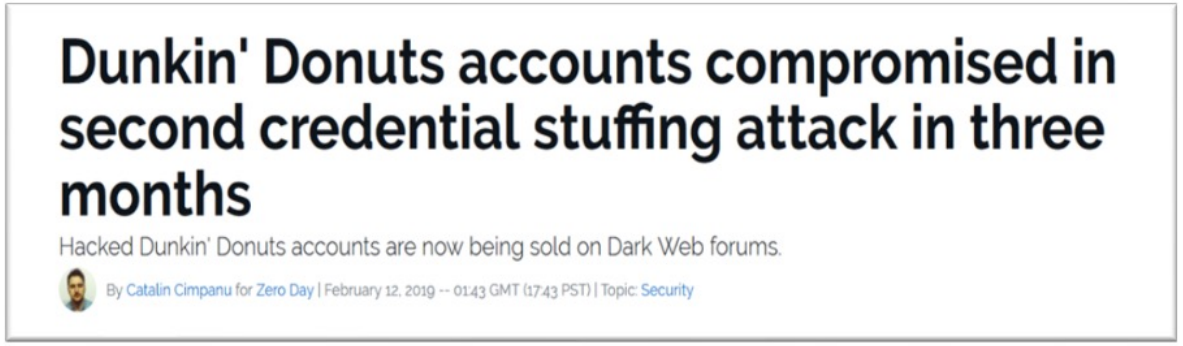
Credential Stuffing

- Attackers try these credentials against other services
  - Sometimes they utilize bots
  - Attackers act like regular users trying to log in
  - Attackers bet on users reusing their passwords

ID	Password
Jaewoo	1234abcd
Yinae	verysecure
Ingyu	1234abcd

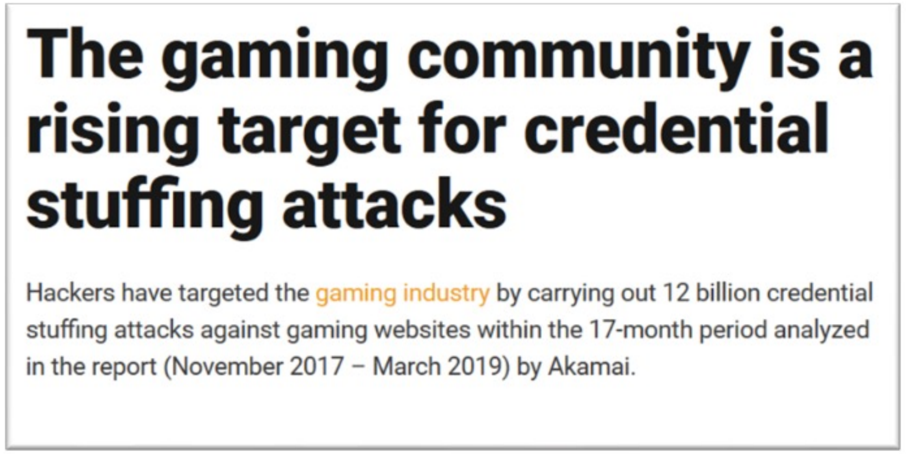


Credential Stuffing is a Real and Growing Problem

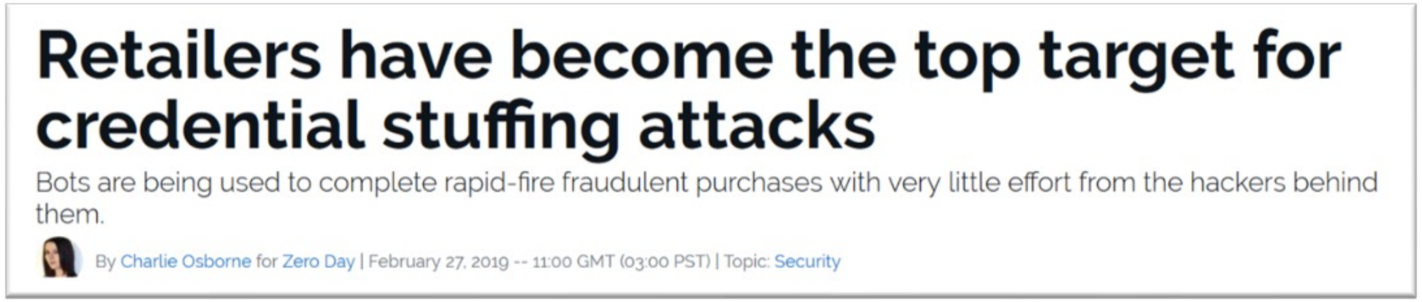

Dunkin' Donuts accounts compromised in second credential stuffing attack in three months

Hacked Dunkin' Donuts accounts are now being sold on Dark Web forums.

By Catalin Cimpanu for Zero Day | February 12, 2019 -- 01:43 GMT (17:43 PST) | Topic: Security

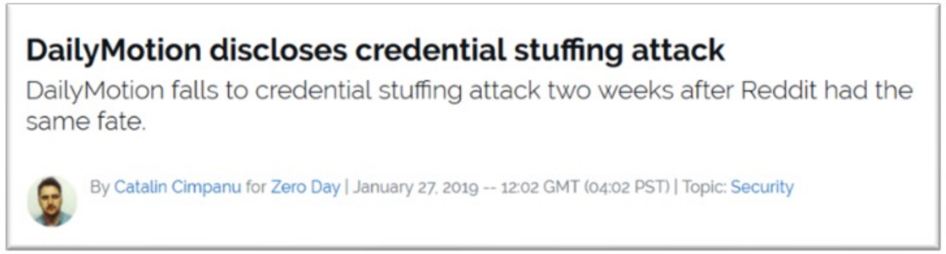

The gaming community is a rising target for credential stuffing attacks

Hackers have targeted the gaming industry by carrying out 12 billion credential stuffing attacks against gaming websites within the 17-month period analyzed in the report (November 2017 – March 2019) by Akamai.


Retailers have become the top target for credential stuffing attacks

Bots are being used to complete rapid-fire fraudulent purchases with very little effort from the hackers behind them.

By Charlie Osborne for Zero Day | February 27, 2019 -- 11:00 GMT (03:00 PST) | Topic: Security


DailyMotion discloses credential stuffing attack

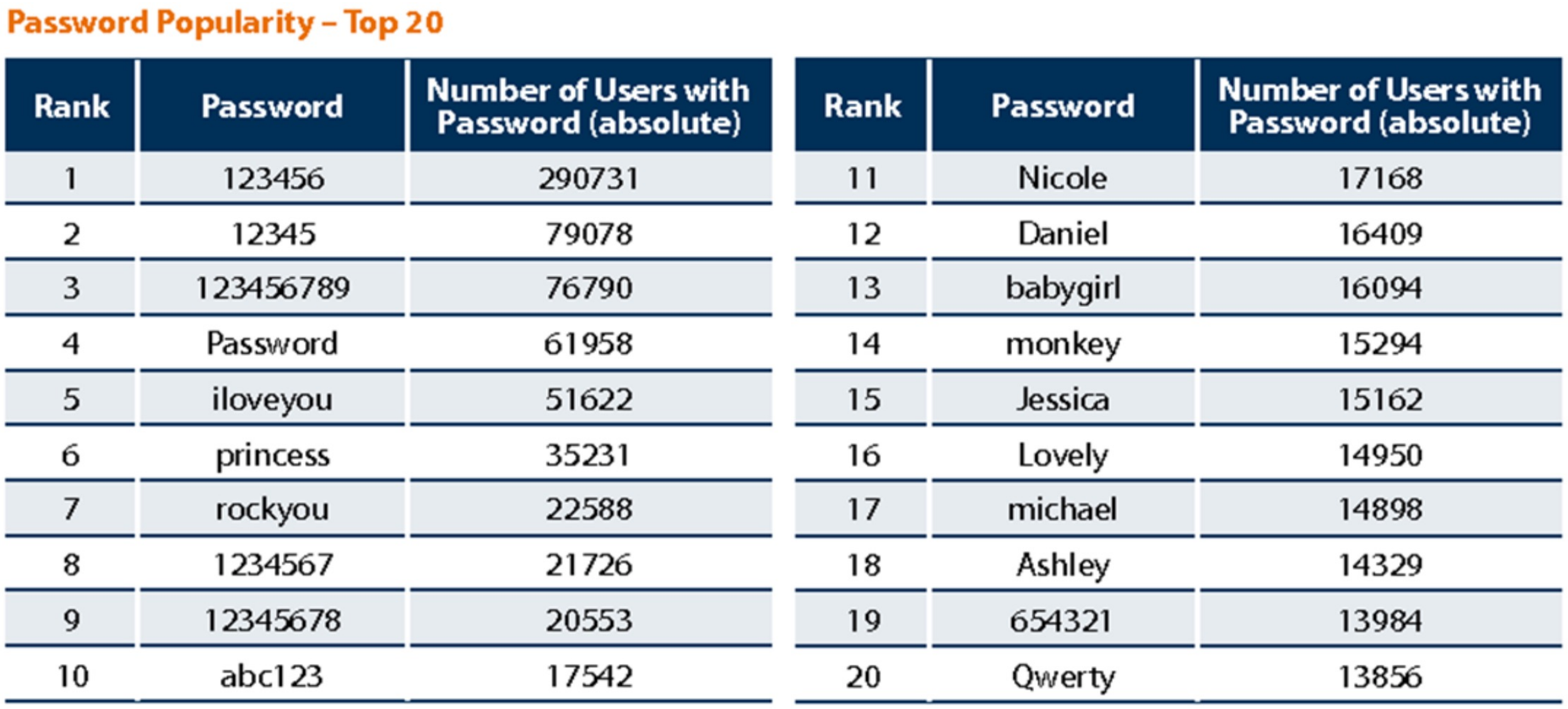
DailyMotion falls to credential stuffing attack two weeks after Reddit had the same fate.

By Catalin Cimpanu for Zero Day | January 27, 2019 -- 12:02 GMT (04:02 PST) | Topic: Security

RockYou Hack (2009)

- “Social gaming” company
- Database with 32 million user passwords from partner social networks
- Passwords stored in the clear
- December 2009: entire database hacked using an SQL injection attack and posted on the Internet

Passwords in RockYou Database



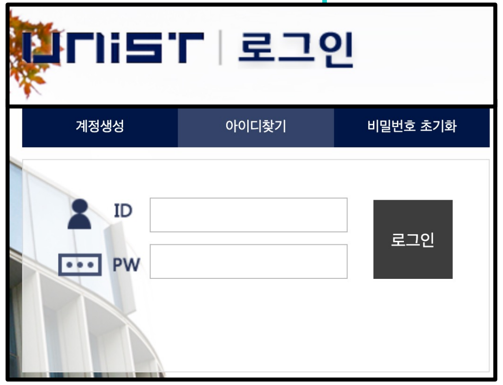
Password Popularity – Top 20

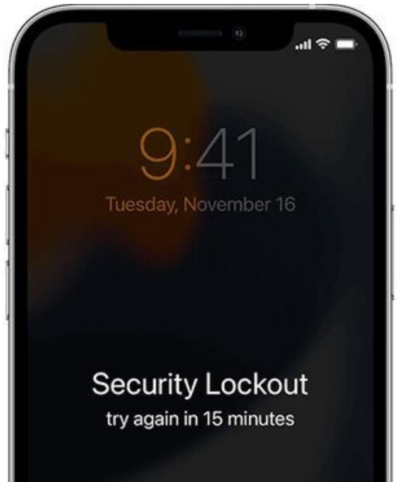
Rank	Password	Number of Users with Password (absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542
11	Nicole	17168
12	Daniel	16409
13	babygirl	16094
14	monkey	15294
15	Jessica	15162
16	Lovely	14950
17	michael	14898
18	Ashley	14329
19	654321	13984
20	Qwerty	13856

Online Attacker

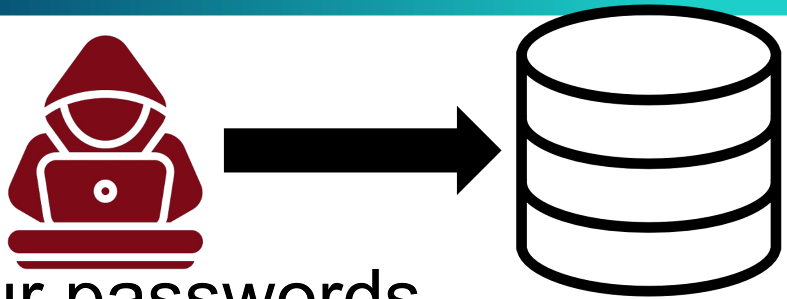
- How do we detect an online attacker?
  - Too many wrong tries
    - Distinctly different from a user who first was wrong but then was right
  - Tries multiple accounts instead of just one

- What can we do?
  - CATCHAs to differentiate between bots and humans
  - Temporarily block the IP address or rate-limit the number of requests
  - Temporarily lock the account that is being attacked
    - Rarely a good solution (Harms availability property)




9:41
Tuesday, November 16
Security Lockout
try again in 15 minutes

Offline Attacker



- Attacker somehow obtains the list of our passwords
  - Break-in to server
    - Credential guessing, SQL injection, Remote-command execution

- It’s obvious that the passwords should not be stored in the clear!
  - How do we not store them in the clear, and still check them against users attempting to log in?

Should We Use Encryption?

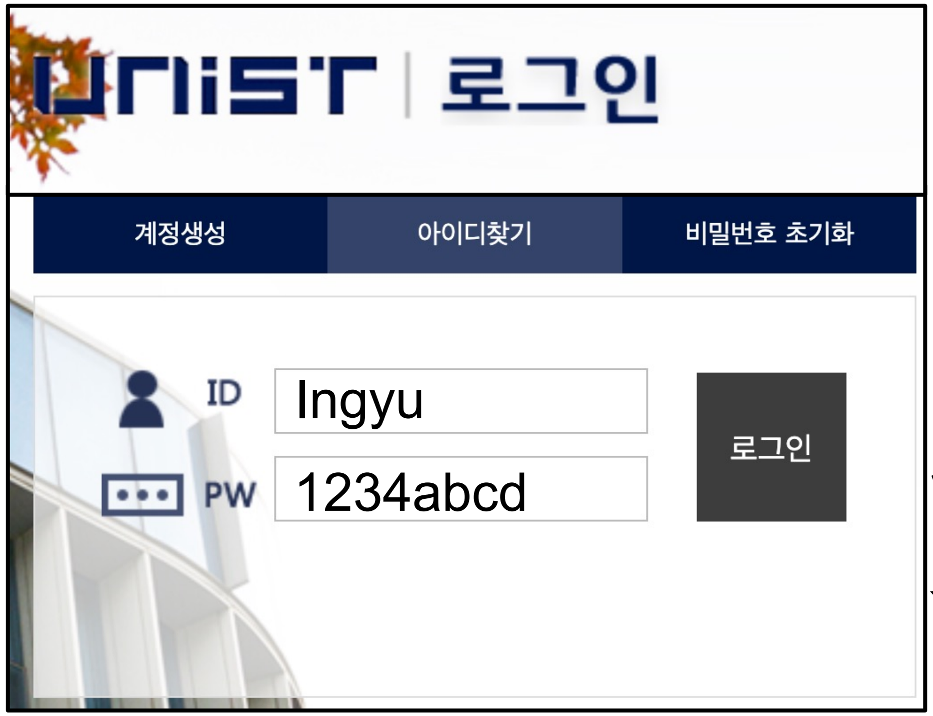
- How about encrypting each password with a secret key (e.g. only stored in the memory of the server) which is used to decrypt any single entry, on demand?

- Still a bad idea....
  - The attacker can steal your key and decrypt everything
  - The administrators can know users' passwords (no reason that they should)

Password Hashing

- Server consults database which contains **Hash(**pw**)** and validates user response

ID	Password
Jaewoo	**Hash(**1234abcd**)**
Yinae	**Hash(**verysecure**)**
Ingyu	**Hash(**1234abcd**)**


UNIST | 로그인

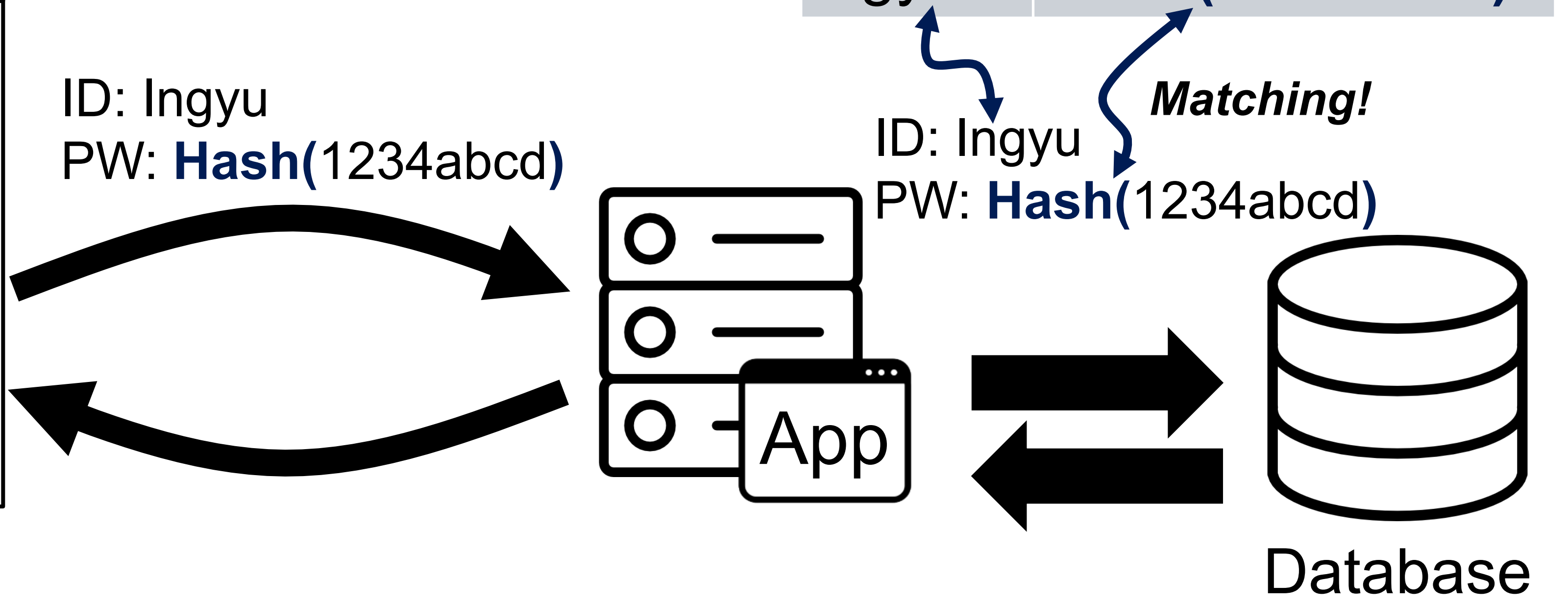
계정생성 | 아이디찾기 | 비밀번호 초기화

ID: Ingyu

PW: 1234abcd

로그인

Browser

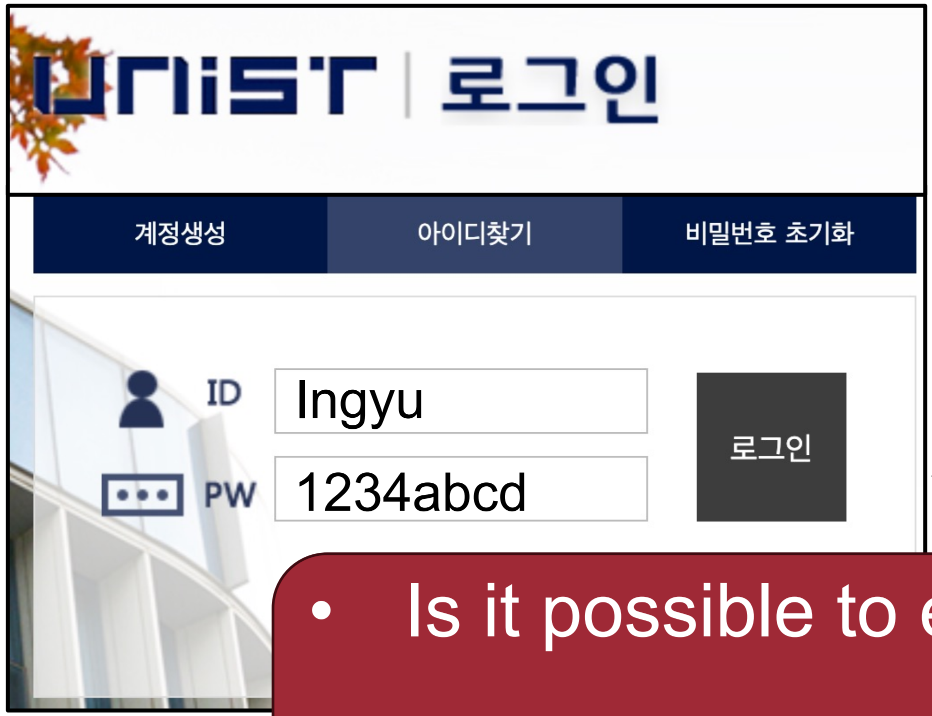


Is It Safe?

Offline attacker

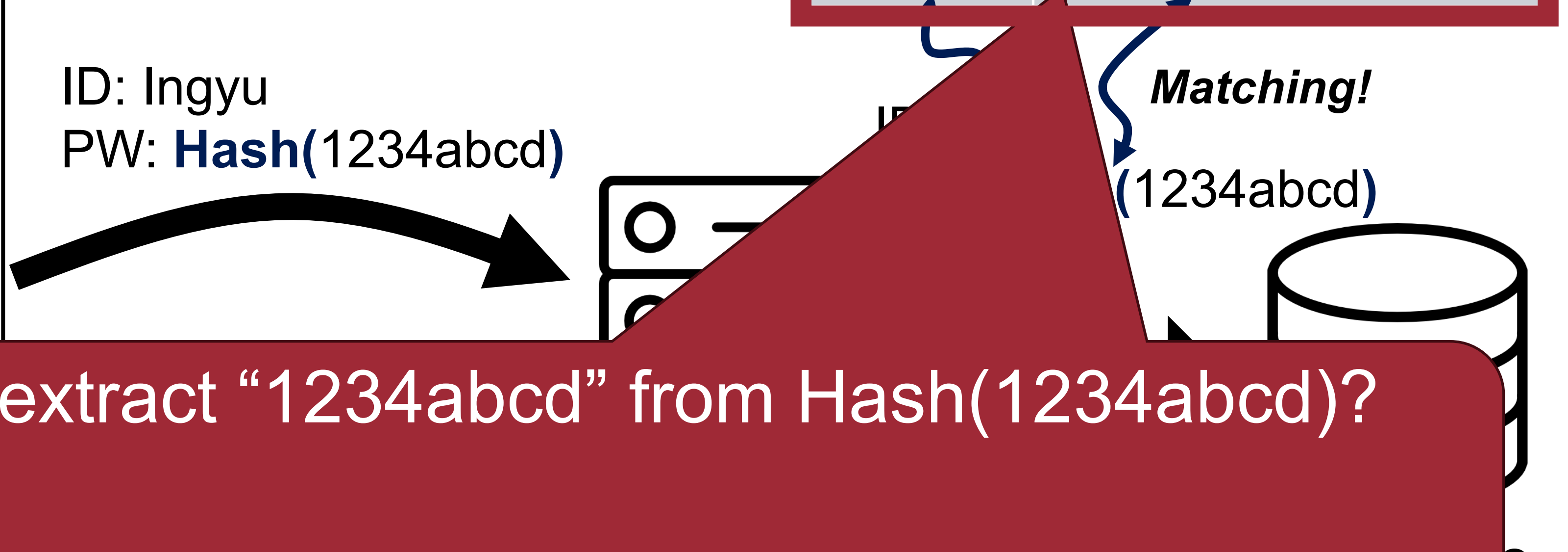
Stealing DB

ID	Password
Jaewoo	**Hash(**1234abcd**)**
Yinae	**Hash(**verysecure**)**
Ingyu	**Hash(**1234abcd**)**



ID Ingyu
PW 1234abcd

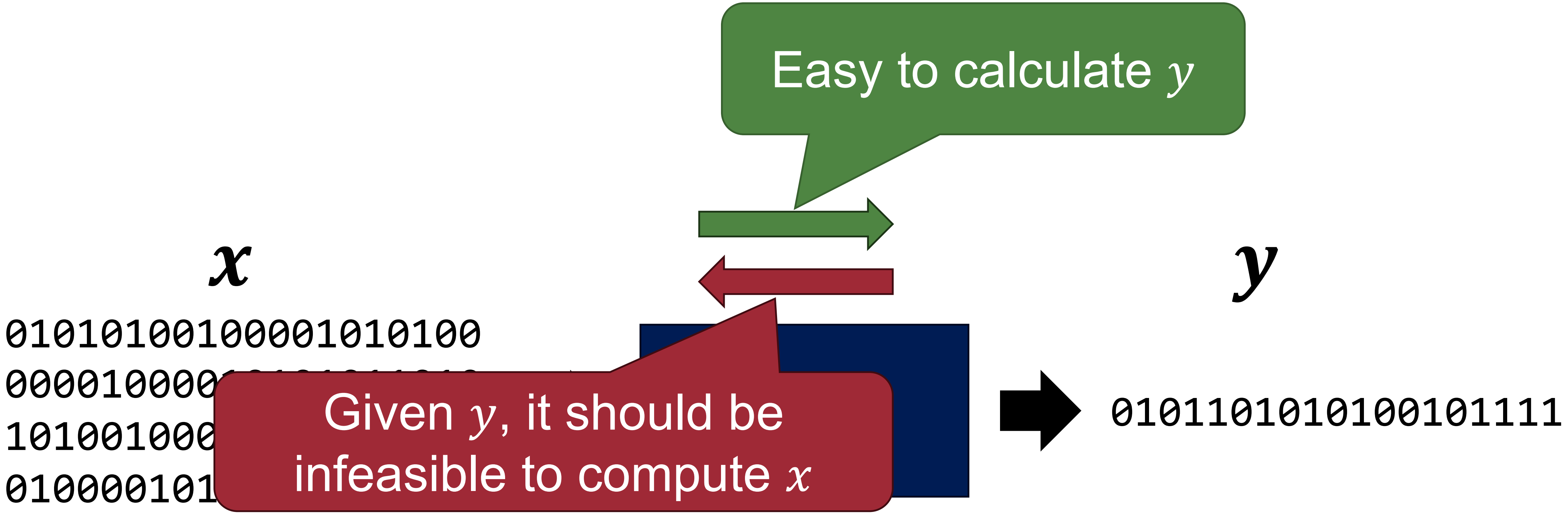
ID: Ingyu
PW: **Hash(**1234abcd**)**



- Is it possible to extract "1234abcd" from Hash(1234abcd)?

Recap - Property #1: Preimage Resistant

- Given y, computationally infeasible to find x such that $H(x) = y$
  - So-called one-way property

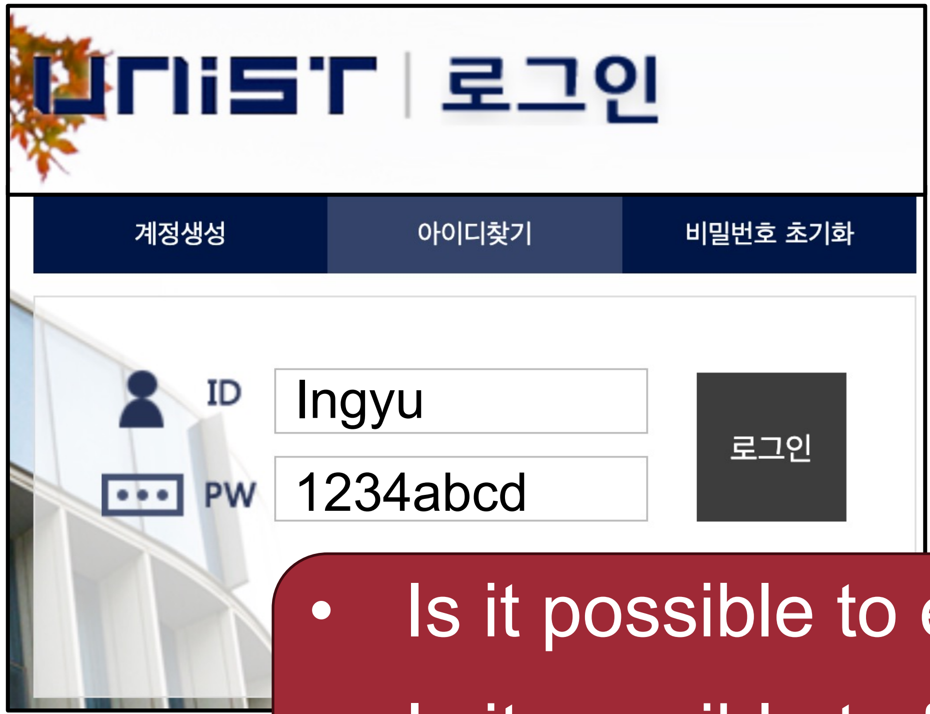


Is It Safe?

Offline attacker

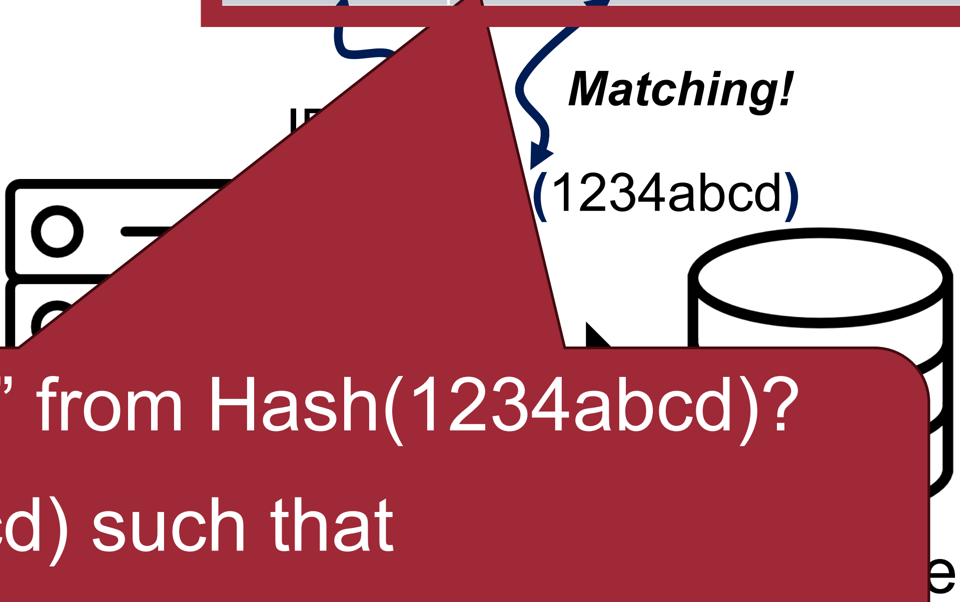
Stealing DB

ID	Password
Jaewoo	**Hash(**1234abcd**)**
Yinae	**Hash(**verysecure**)**
Ingyu	**Hash(**1234abcd**)**



ID: Ingyu
PW: 1234abcd

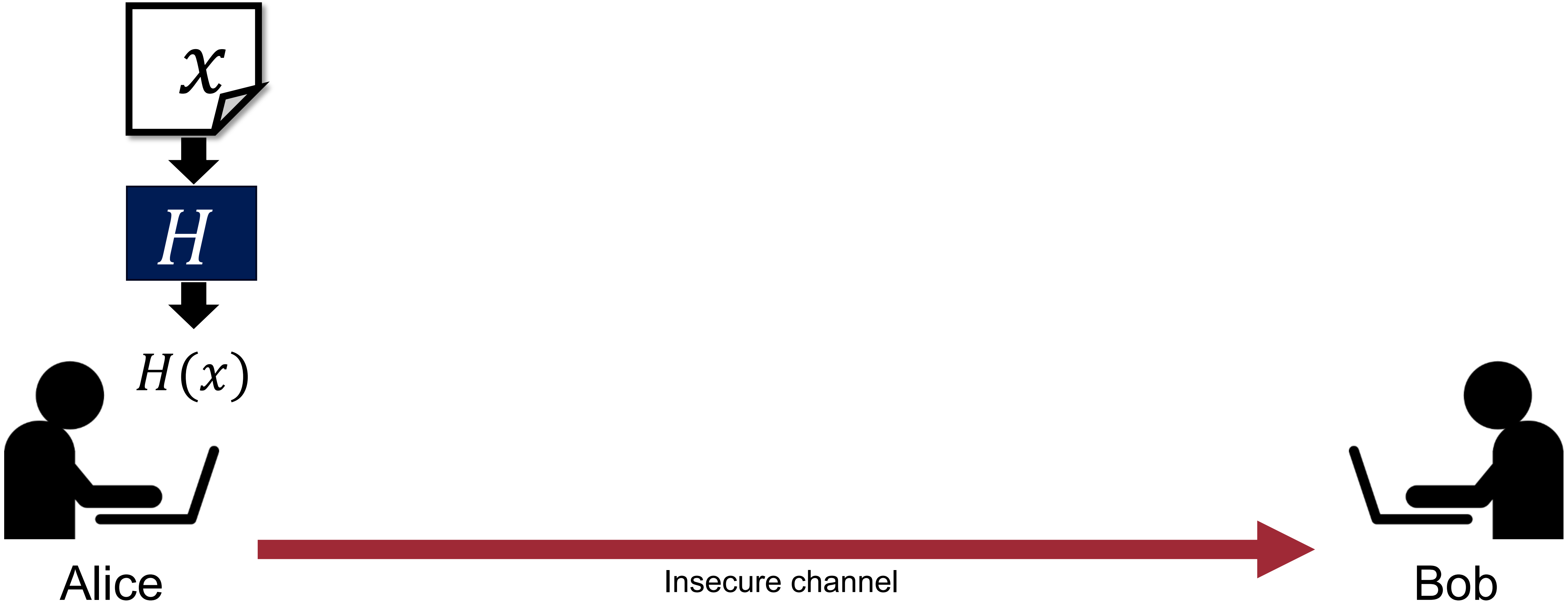
ID: Ingyu
PW: **Hash(**1234abcd**)**



- Is it possible to extract “1234abcd” from Hash(1234abcd)?
- Is it possible to find z (≠ 1234abcd) such that Hash(1234abcd) = Hash(z)?

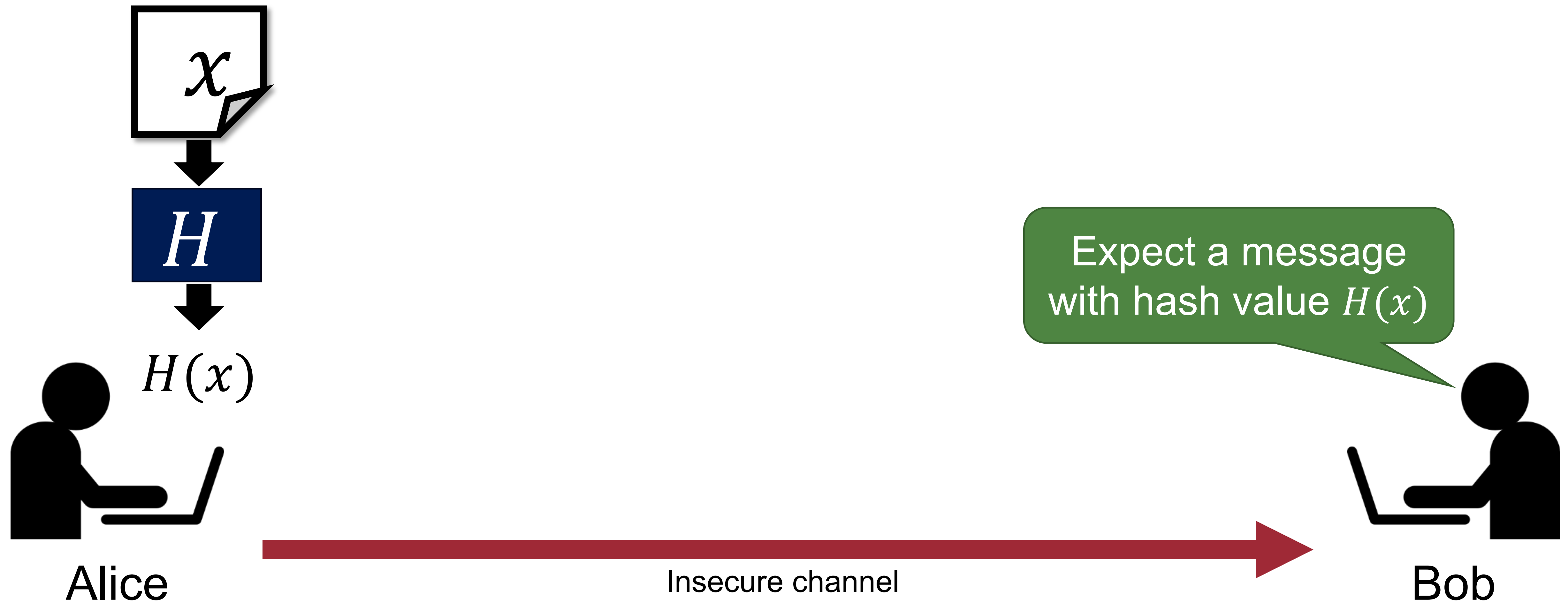
Recap - Property #2: Second Preimage Resistant

- Given x, computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$



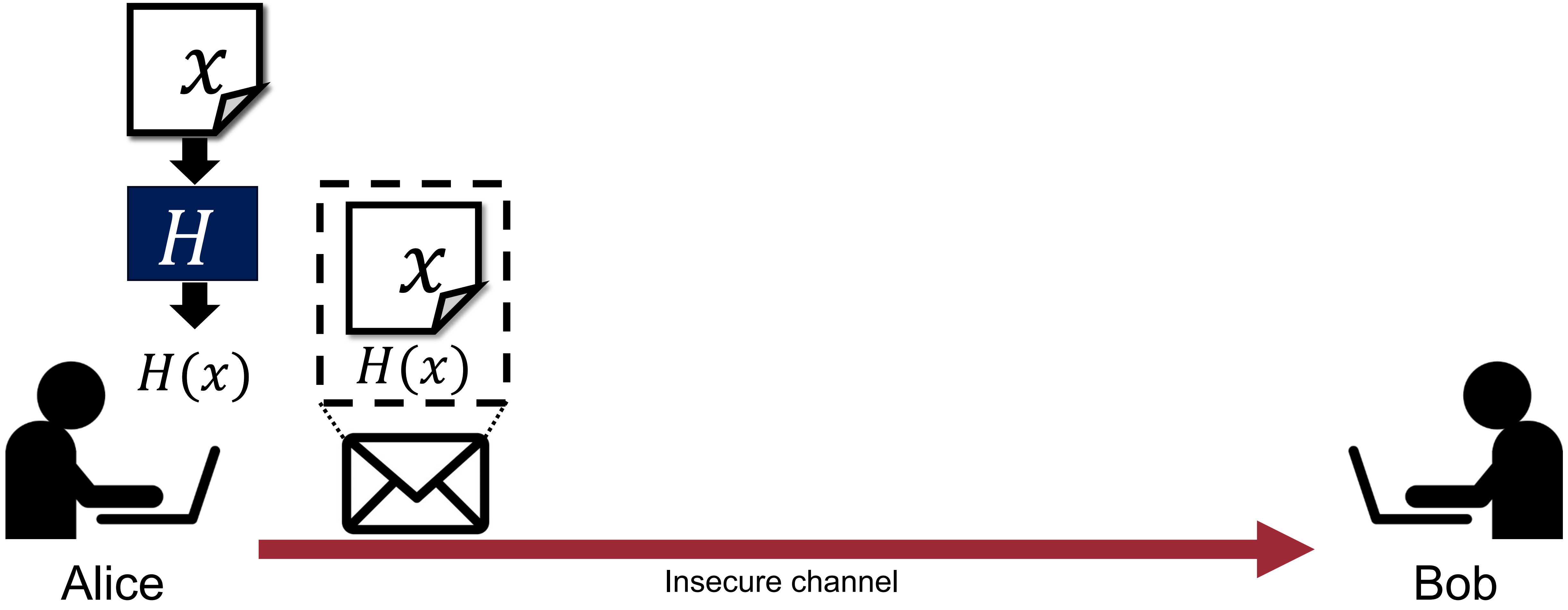
Recap - Property #2: Second Preimage Resistant

- Given x, computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$



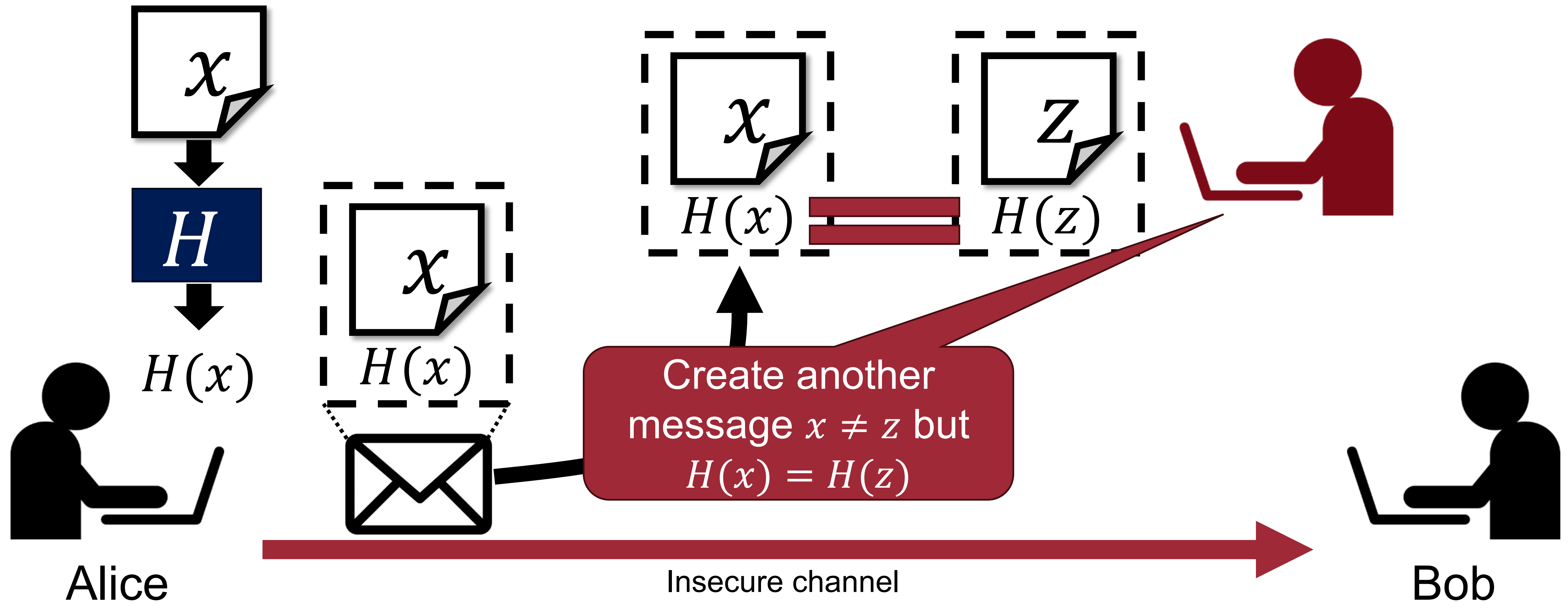
Recap - Property #2: Second Preimage Resistant

- Given x, computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$



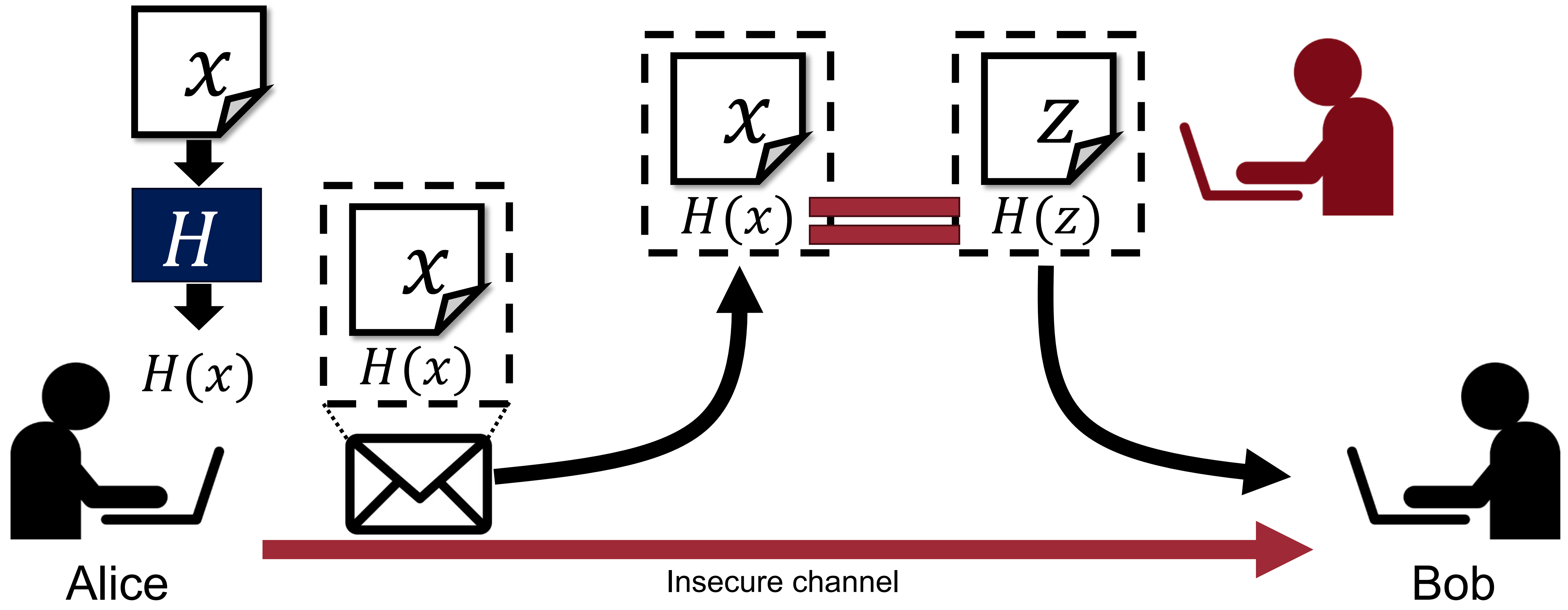
Recap - Property #2: Second Preimage Resistant

- Given x, computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$



Recap - Property #2: Second Preimage Resistant

- Given x, computationally infeasible to find z such that $x \neq z$ and $H(x) = H(z)$



Sample Cryptographic Hash Functions

Name	Year of release	Digest size (output size)
MD5 (Media Digest 5)	1992	128-bit
SHA-1 (Secure Hash Algorithm 1)	1995	160-bit
SHA-256 (Part of the SHA-2 family)	2001	256-bit

- `MD5(“helloworld”) = d73b04b0e696b0945283defa3eee4538`
- `SHA-1(“helloworld”) = e7509a8c032f3bc2a8df1df476f8ef03436185fa`
- `SHA-256(“helloworld”) = 8cd07f3a5ff98f2a78cfc366c13fb123eb8d29c1ca37c79df190425d5b9e424d`

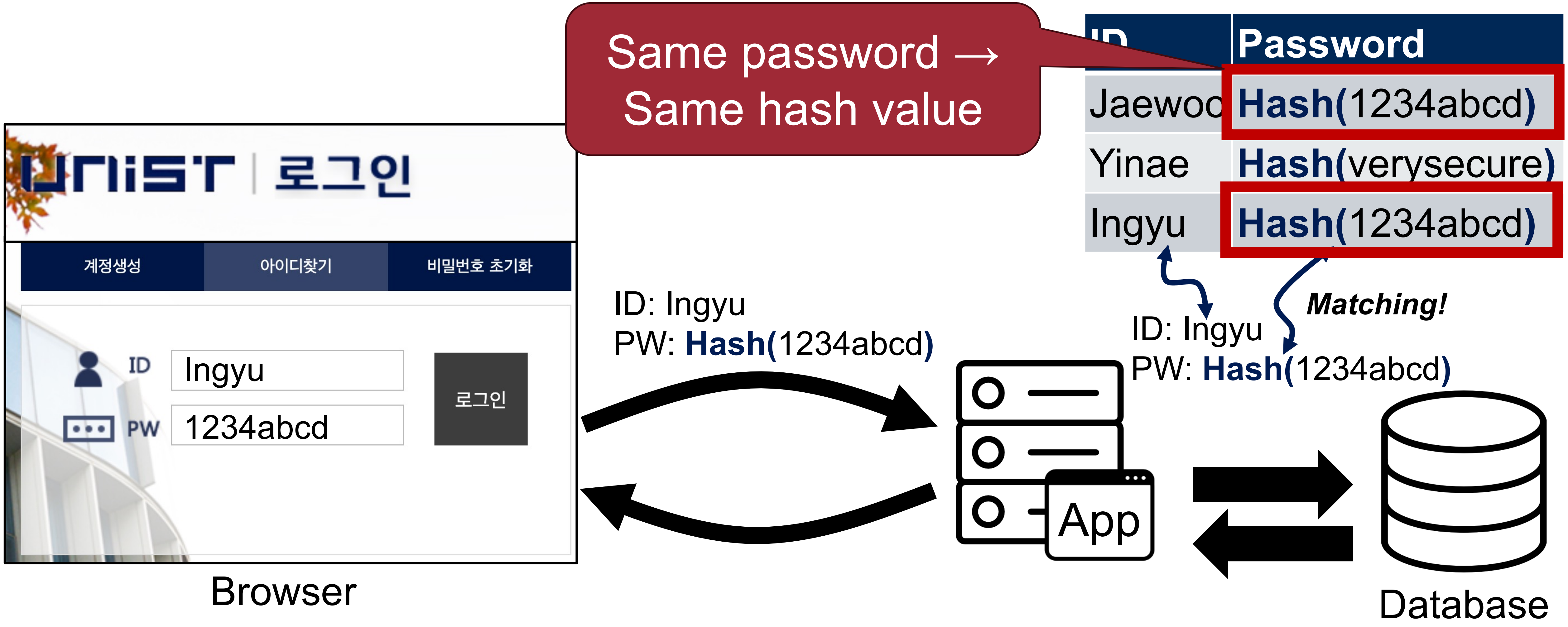
Examples

- SHA1("mysecretpasswo") =
  27c2d31b648cf7773032d1a06c8ee610c3f5b32c
- SHA1("mysecretpasswor") =
  0c894b9cd0fef7d1ccfe0729d5ff7af9509731ed
- SHA1("mysecretpassword") =
  08cd923367890009657eab812753379bdb321eeb

Small changes in input

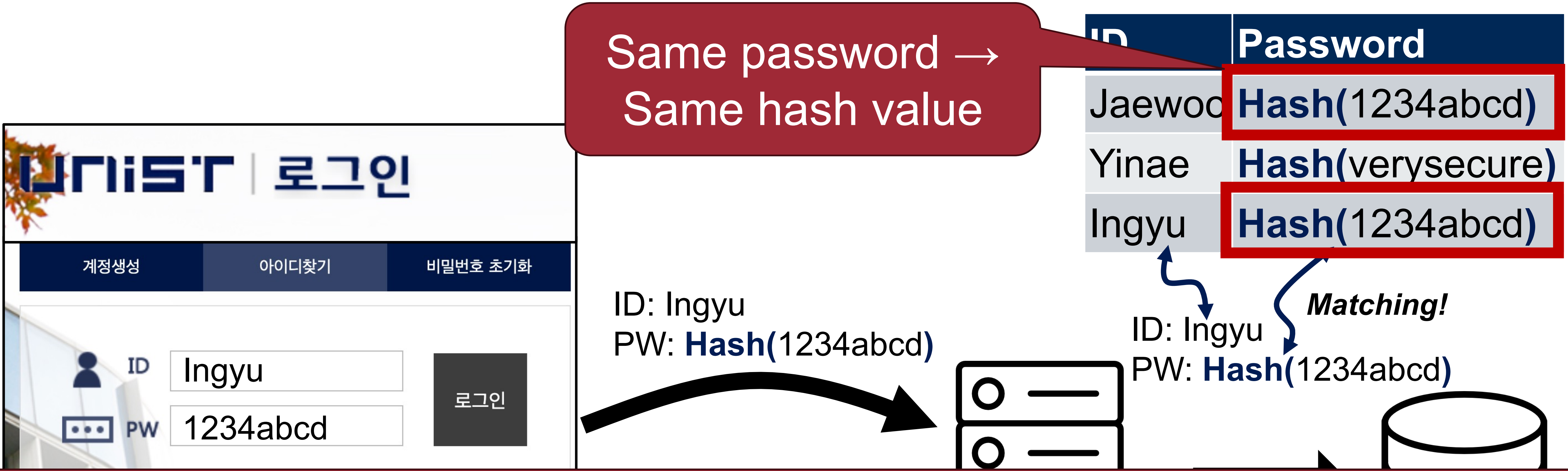
Large differences in output

Problems of Password Hashing?



ID	Password
Jaewoo	Hash(1234abcd)
Yinae	Hash(verysecure)
Ingyu	Hash(1234abcd)

Problems of Password Hashing?



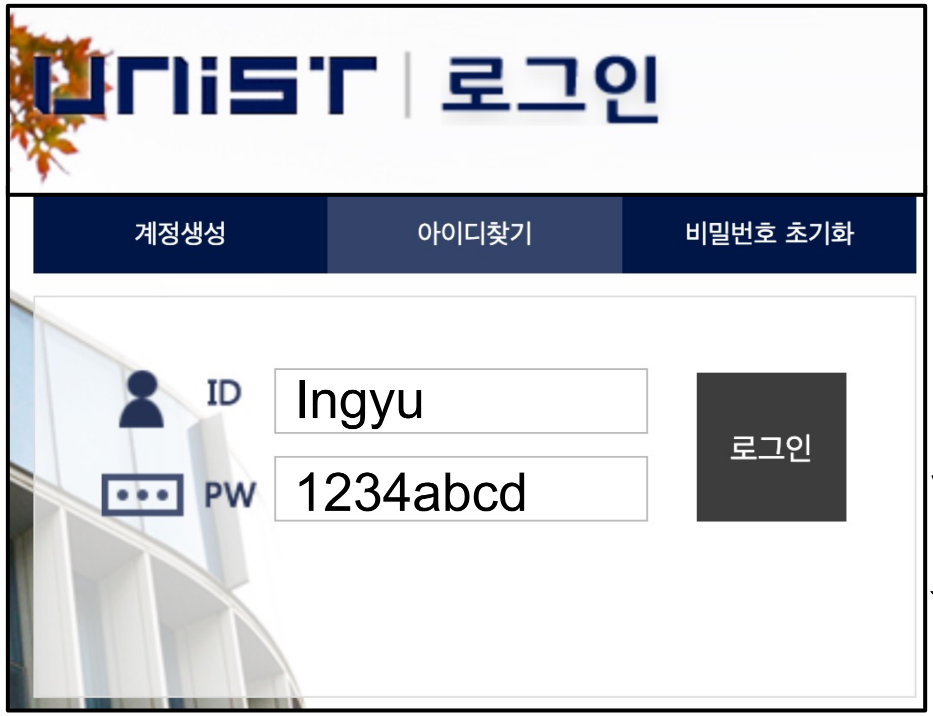
ID	Password
Jaewoo	Hash(1234abcd)
Yinae	Hash(verysecure)
Ingyu	Hash(1234abcd)

ID: Ingyu
PW: 1234abcd

Attacker can precompute hashes of ***popular words*** and try them against all accounts

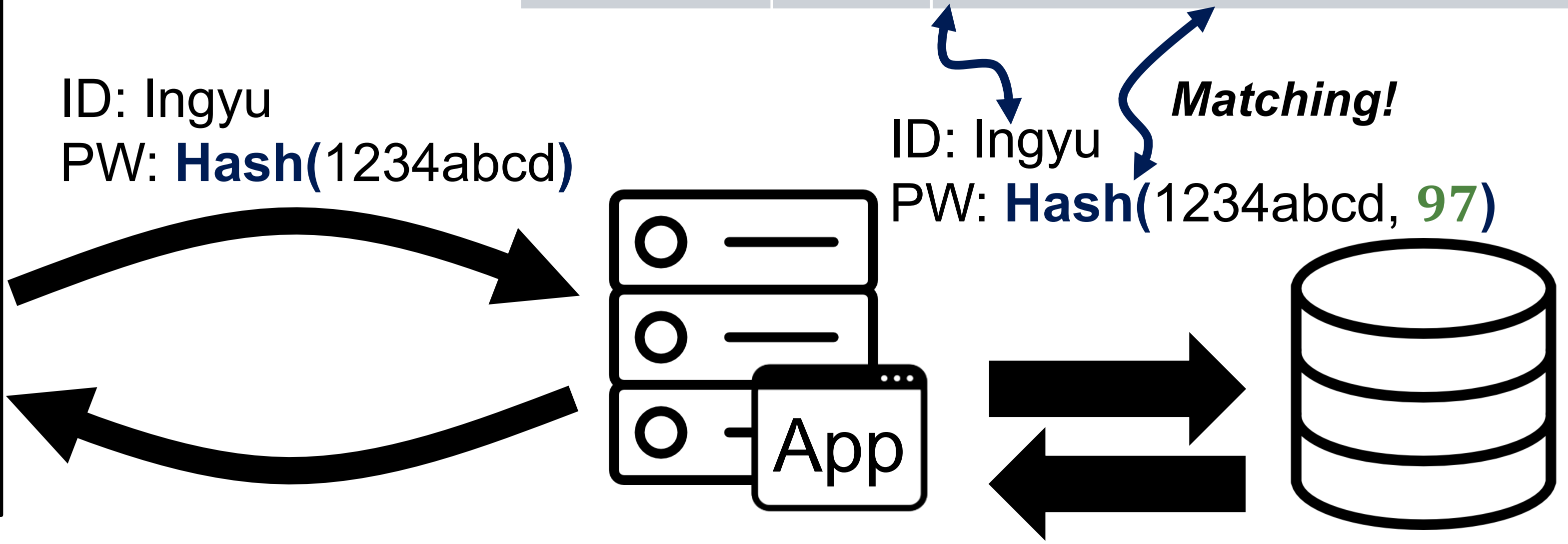
Recap: Salted Hash

ID	Salt	Password
Jaewoo	23	**Hash(**1234abcd, 23**)**
Yinae	51	**Hash(**verysecure, 51**)**
Ingyu	97	**Hash(**1234abcd, 97**)**



ID: Ingyu
PW: 1234abcd

Browser



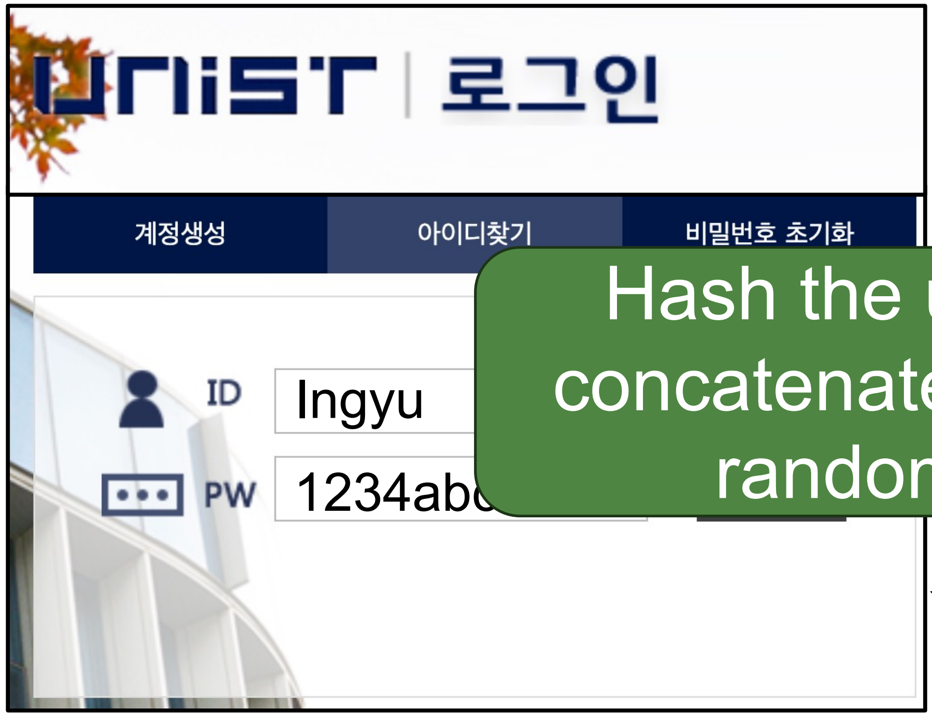
Database

Recap: Salted Hash

Same password → Different hash value

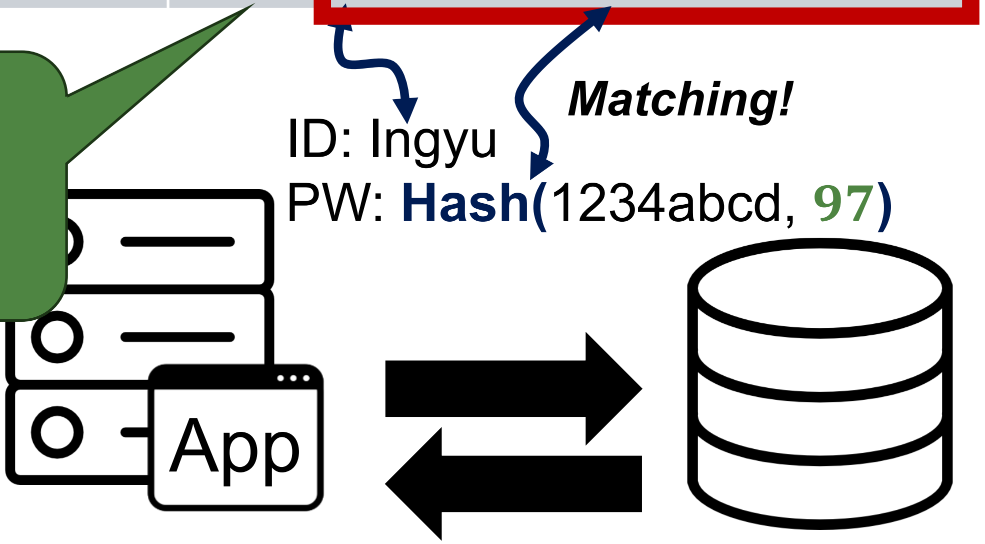
	Salt	Password
Jaewoo	23	Hash(1234abcd, 23)
Yinae	51	Hash(verysecure, 51)
Ingyu	97	Hash(1234abcd, 97)

Hash the user's password concatenated with a per-user random value (salt)



ID: Ingyu
PW: 1234abcd

Browser



ID: Ingyu
PW: Hash(1234abcd, 97)

Database

Problems of Salted Hash?

- Our steps so far allow us the following guarantees:
  - User passwords should not be recoverable from a database
  - Identical/similar passwords will have different hashes
  - The database does not “leak” the length of a user’s password

- Still has a problem of password guessing attack!
  - ***Offline attackers*** can still brute-force their way into users with weak passwords (if they are dedicated enough)

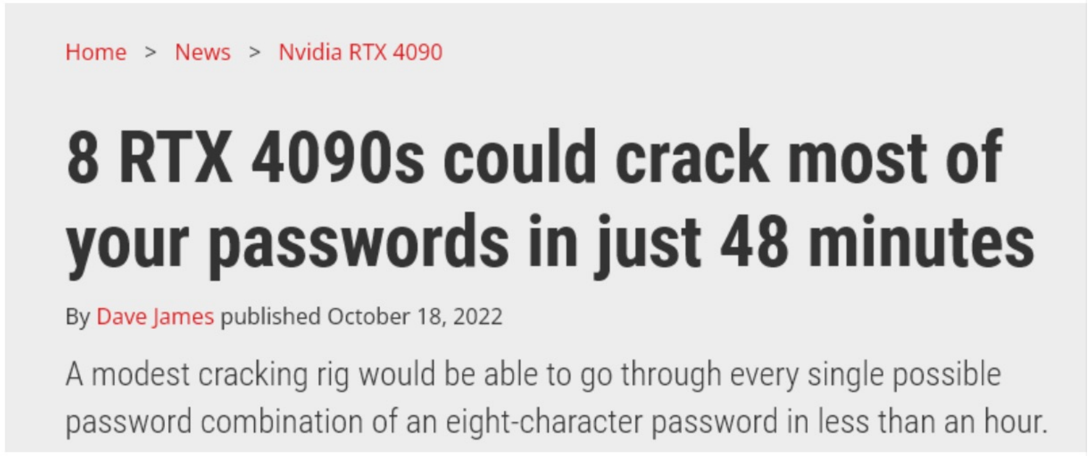
Password Guessing Techniques

- Dictionary with words spelled backwards
- First and last names, streets, cities
- Same with upper-case initials
- Room numbers, telephone numbers, etc.
- Letter substitutions and other tricks

If you can think of it, attacker will, too!

Password Hash Cracking

- Custom GPU-based hardware
  - GPUs are great for playing games and hashing
  - Most recent number for Nvidia RTX 4090
    - 300 Gigahashes per second for Windows NTLM hashes



Home > News > Nvidia RTX 4090

8 RTX 4090s could crack most of your passwords in just 48 minutes

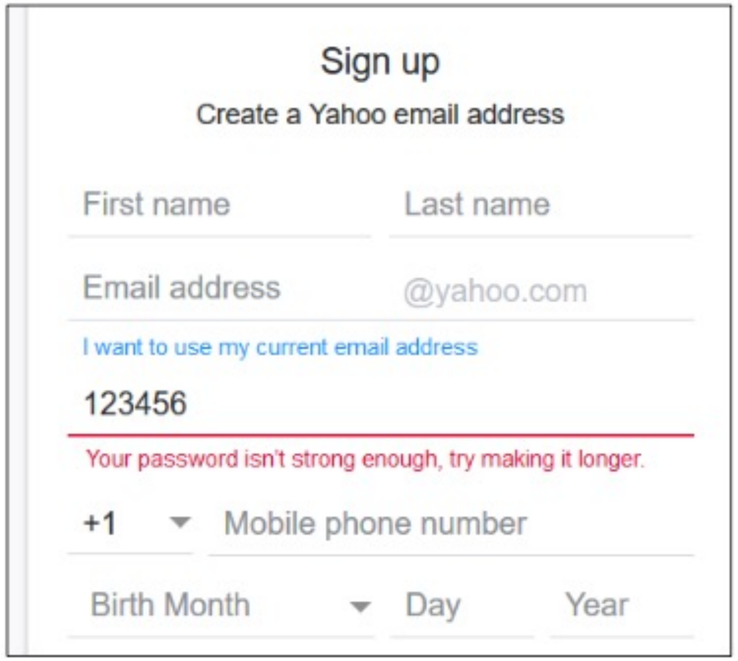
By Dave James published October 18, 2022

A modest cracking rig would be able to go through every single possible password combination of an eight-character password in less than an hour.

- Cloud-based cracking tools
  - Crackq
  - Password-cracking as a service

Defense: Password Requirements

- Systems can enforce password requirements when users register/change their passwords
  - Not a dictionary word
  - Must be at least X characters long
  - Must contain special characters
  - Is not part of a recently compromised database



Sign up

Create a Yahoo email address

First name | Last name

Email address @yahoo.com

I want to use my current email address

123456

Your password isn't strong enough, try making it longer.

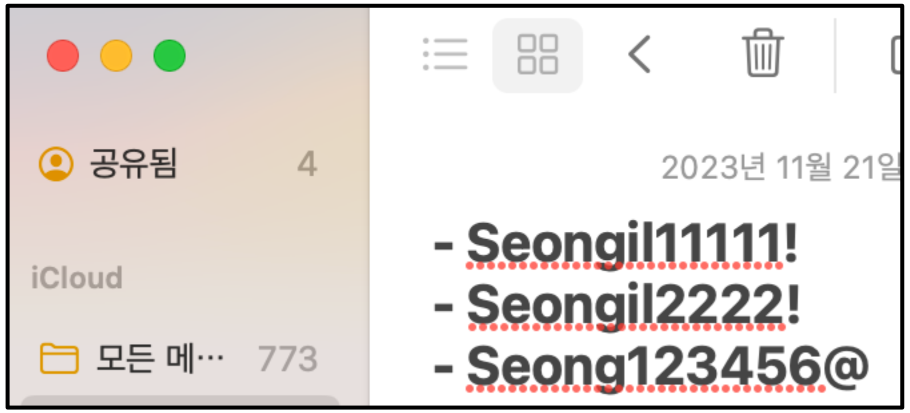
+1 Mobile phone number

Birth Month Day Year

- Other requirements are popular but not actually good
  - Change password every N months
    - NIST (National Institute of Standards and Technology) does not recommend forced password changes when passwords are not compromised

Limitations of Password Requirements (1)

- Overly restrictive password policies…
  - 7 or 8 characters, at least 3 out of {digits, upper-case, lower-case, non-alphanumeric}, no dictionary words, change every 4 months, password may not be similar to previous 12 passwords...
- … result in frustrated users and less security
  - Burdens of devising, learning, forgetting passwords
  - Users construct passwords insecurely, write them down



2023년 11월 21일

- Seongil11111!
- Seongil2222!
- Seong123456@

  - Heavy password re-use across systems
    - Do you use the same password for UNIST Portal and Google?

Limitations of Password Requirements (2)

- Typically, **strength** of a password and **memorability** are working against each other (Trade off)
  - People can’t remember arbitrary 13-character sequences
  - You can likely remember “jack123” better than “399%(mJjaweee”

Outline

- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

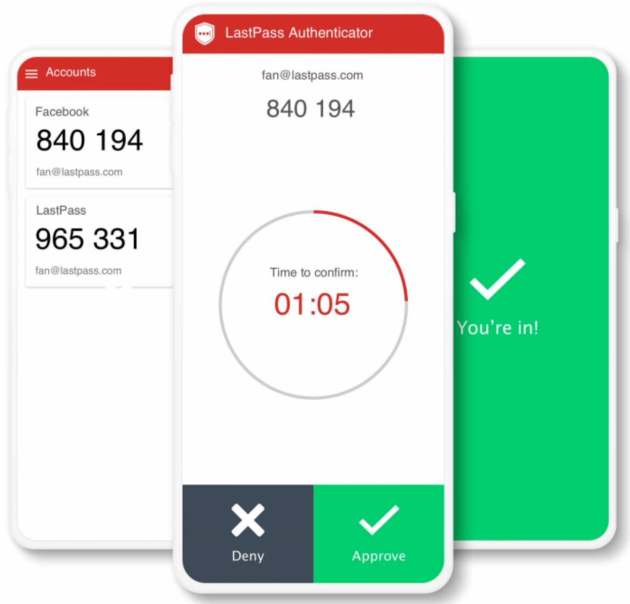
Outline

- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

Authentication Token – Something You Have

- Things one can have
  - Access to your smartphone
  - A back card
  - A secret token
    - Hardware: OTP tokens
    - Software: JWT, OAuth
  - A encryption/decryption keys
  - A badge

- Problems:
  - Stolen / forgotten / lost / duplicated
    - Higher cost to change than passwords
  - Cost of user education and support

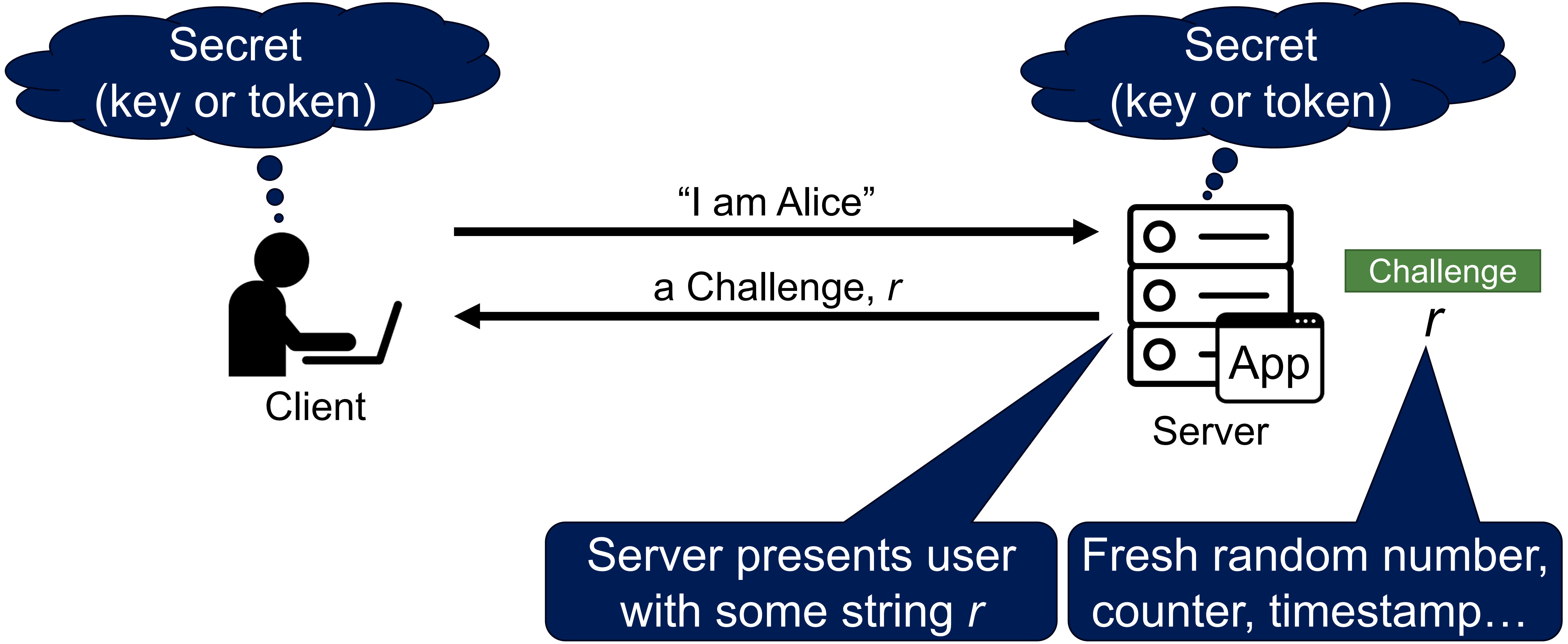


Authentication Token – Something You Have

- Types
  - Challenge-response authentication
  - Time-based authentication

Challenge-Response Authentication

- Key idea: avoid sending password-related information

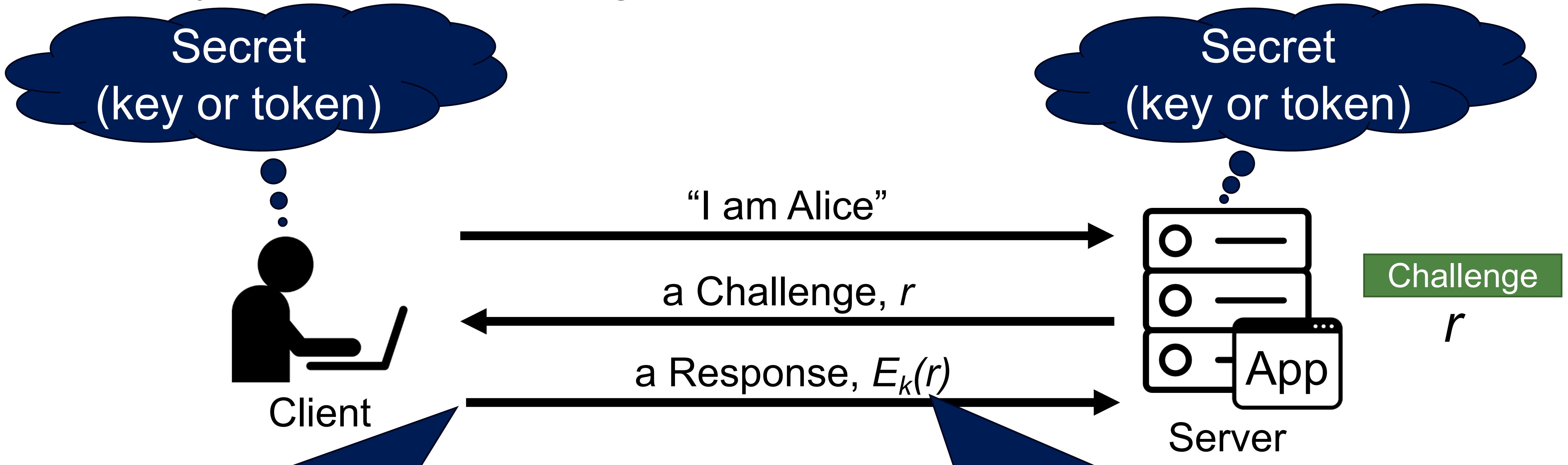


Server presents user with some string *r*

Fresh random number, counter, timestamp…

Challenge-Response Authentication

- Key idea: avoid sending password-related information



Computes the repones based on the secret and the challenge

Cryptographic hashing or symmetric/asymmetric encryption

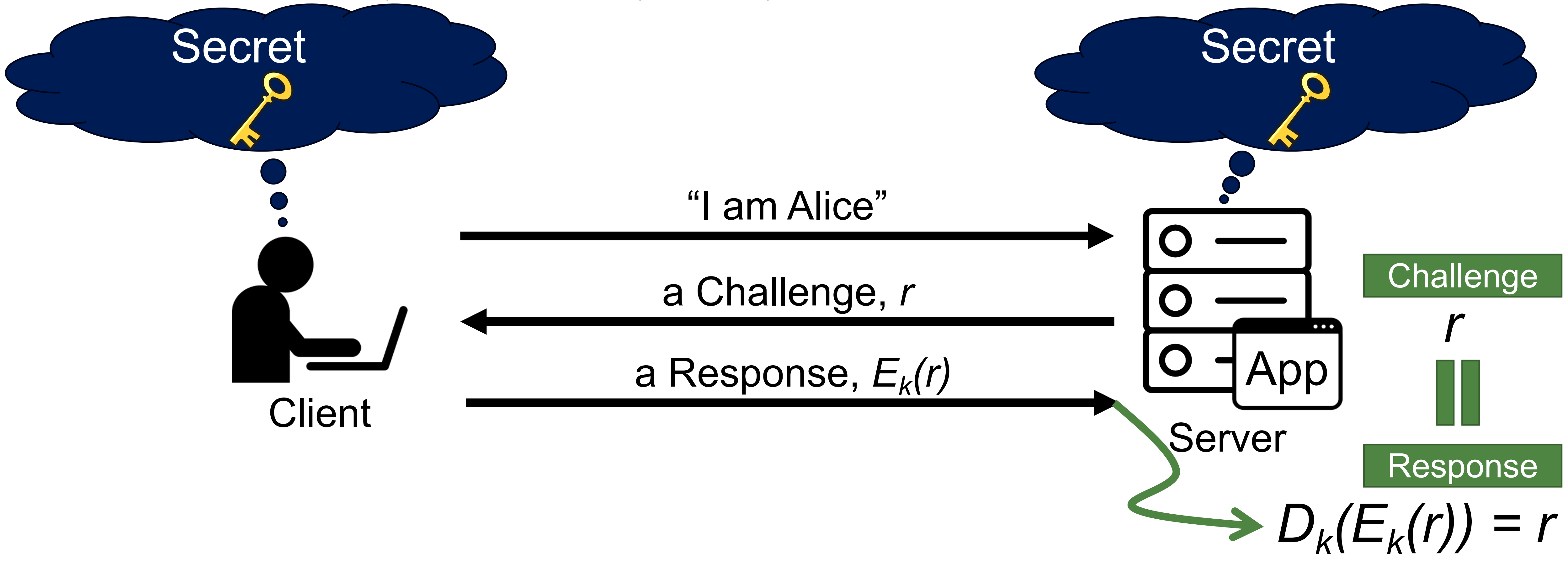
Challenge-Response Authentication

- Key idea: avoid sending password-related information

- Why is this better than the password over a network?
  - **Secrecy**: difficult to recover secret from response
    - Cryptographic hashing or symmetric encryption work well
  - **Freshness**: if the challenge is fresh, attacker on the network cannot replay an old response
    - Fresh random number, counter, timestamp, …

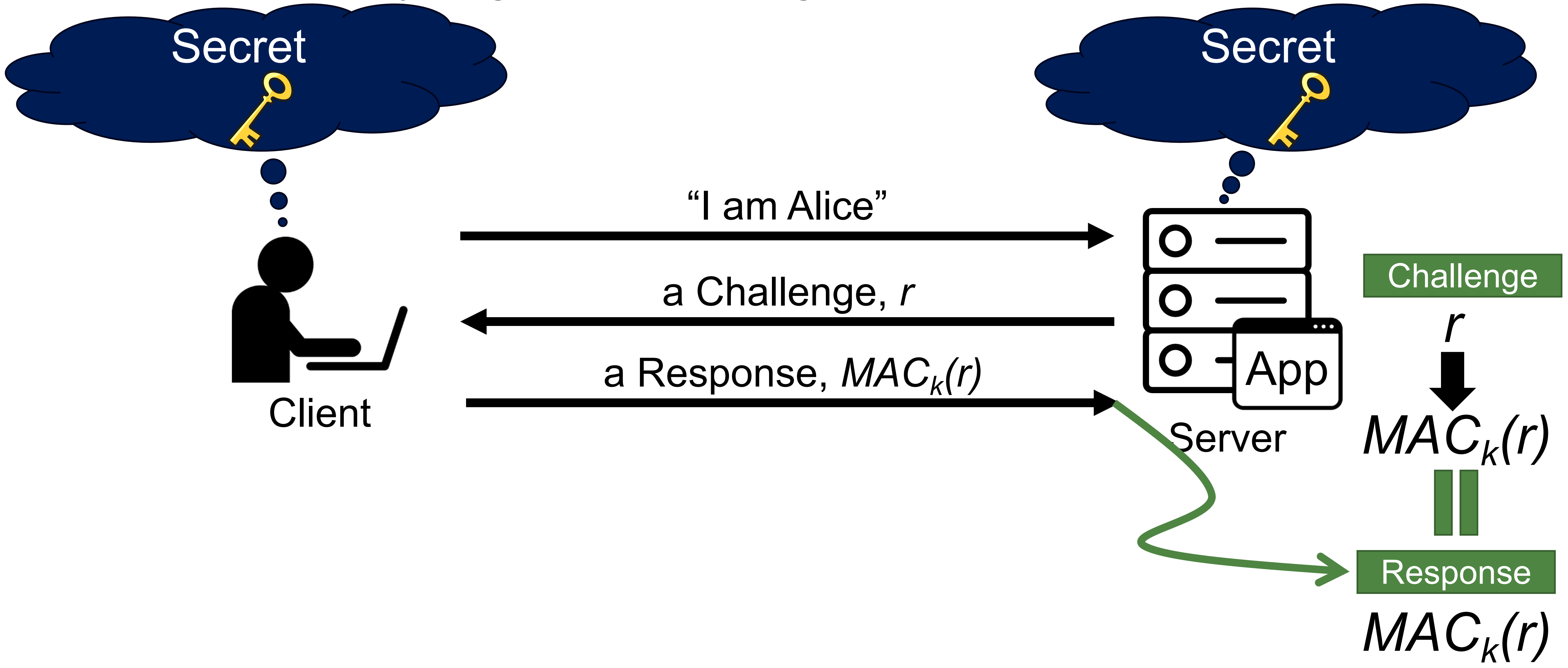
Challenge-Response Authentication

- Use case: symmetric key encryption



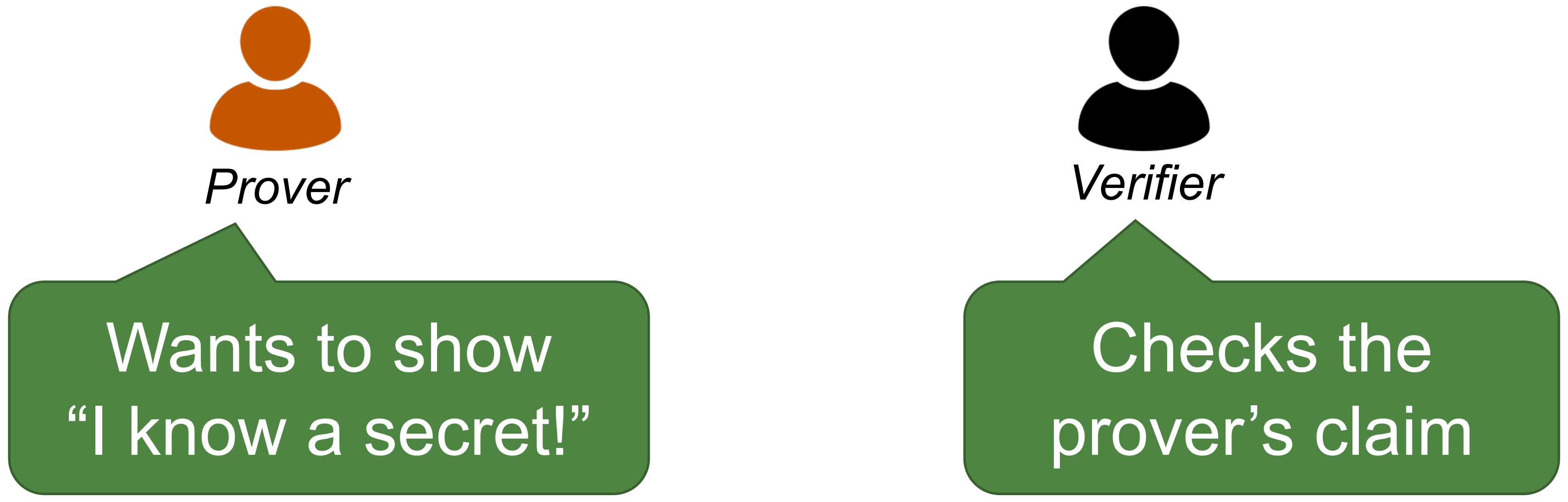
Challenge-Response Authentication

- Use case: cryptographic hashing



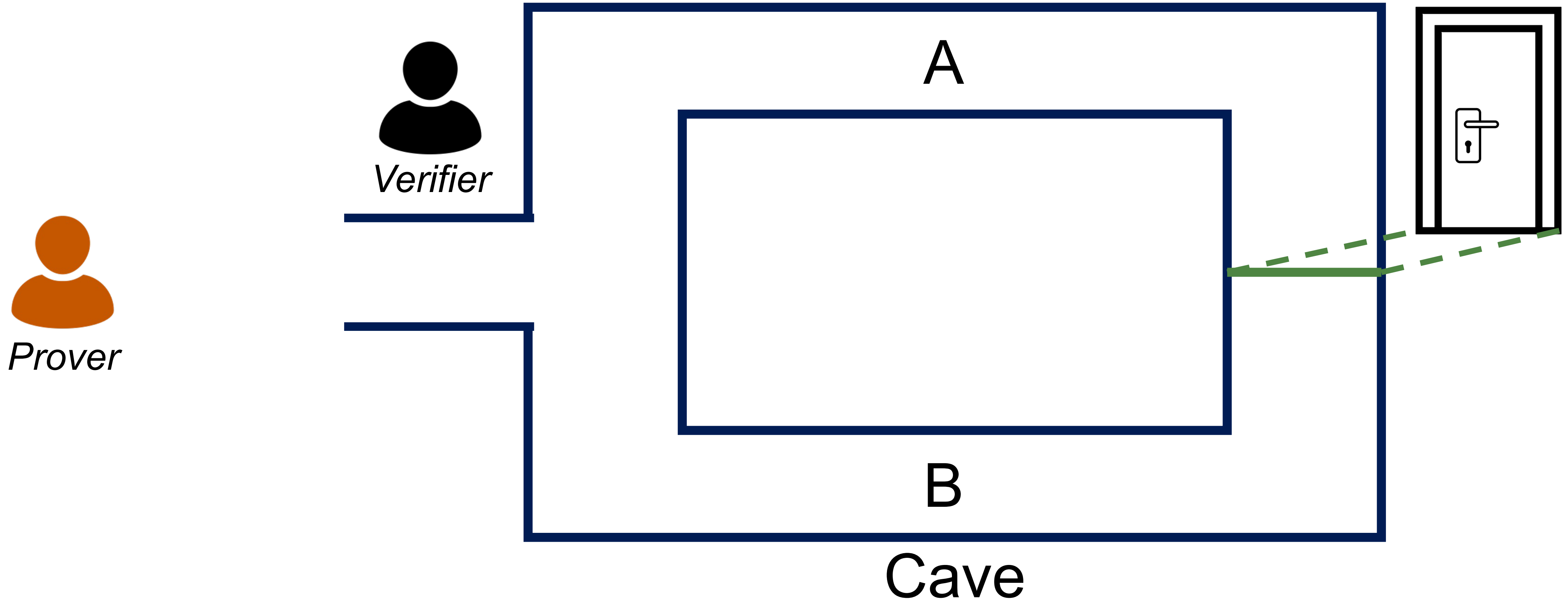
Zero-Knowledge Proof

- A method by which the *prover* can prove to *verifier* that they know a secret, <u>without revealing anything about the secret</u>



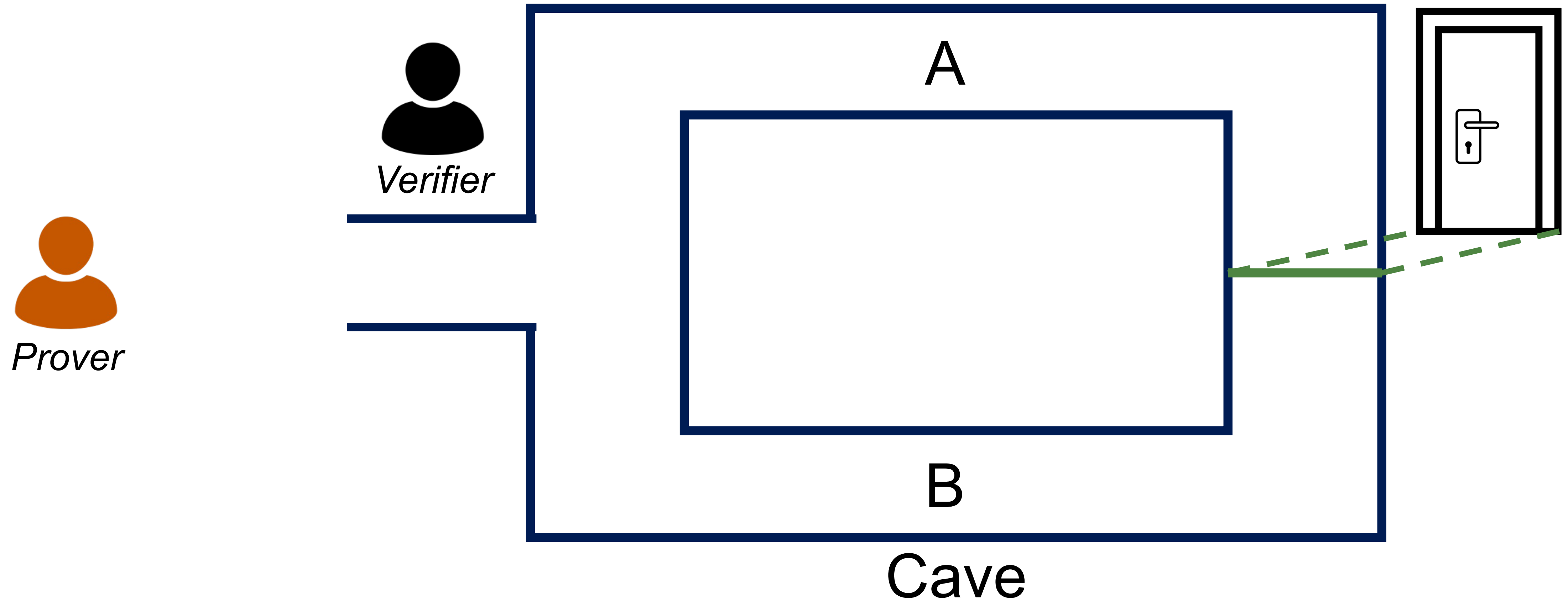
Zero-Knowledge Proof

- A method by which the *prover* can prove to *verifier* that they know a secret, without revealing anything about the secret



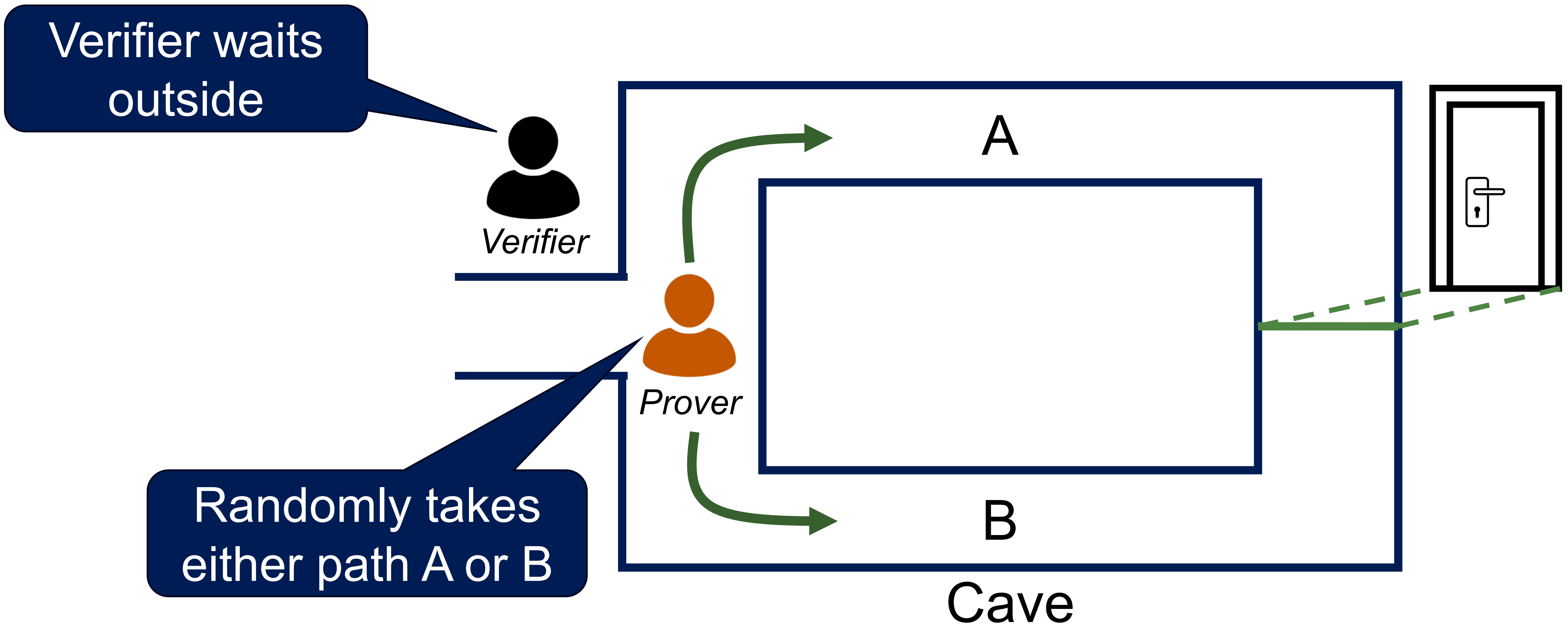
Zero-Knowledge Proof

- A method by which the *prover* can prove to *verifier* that they know a secret, without revealing anything about the secret



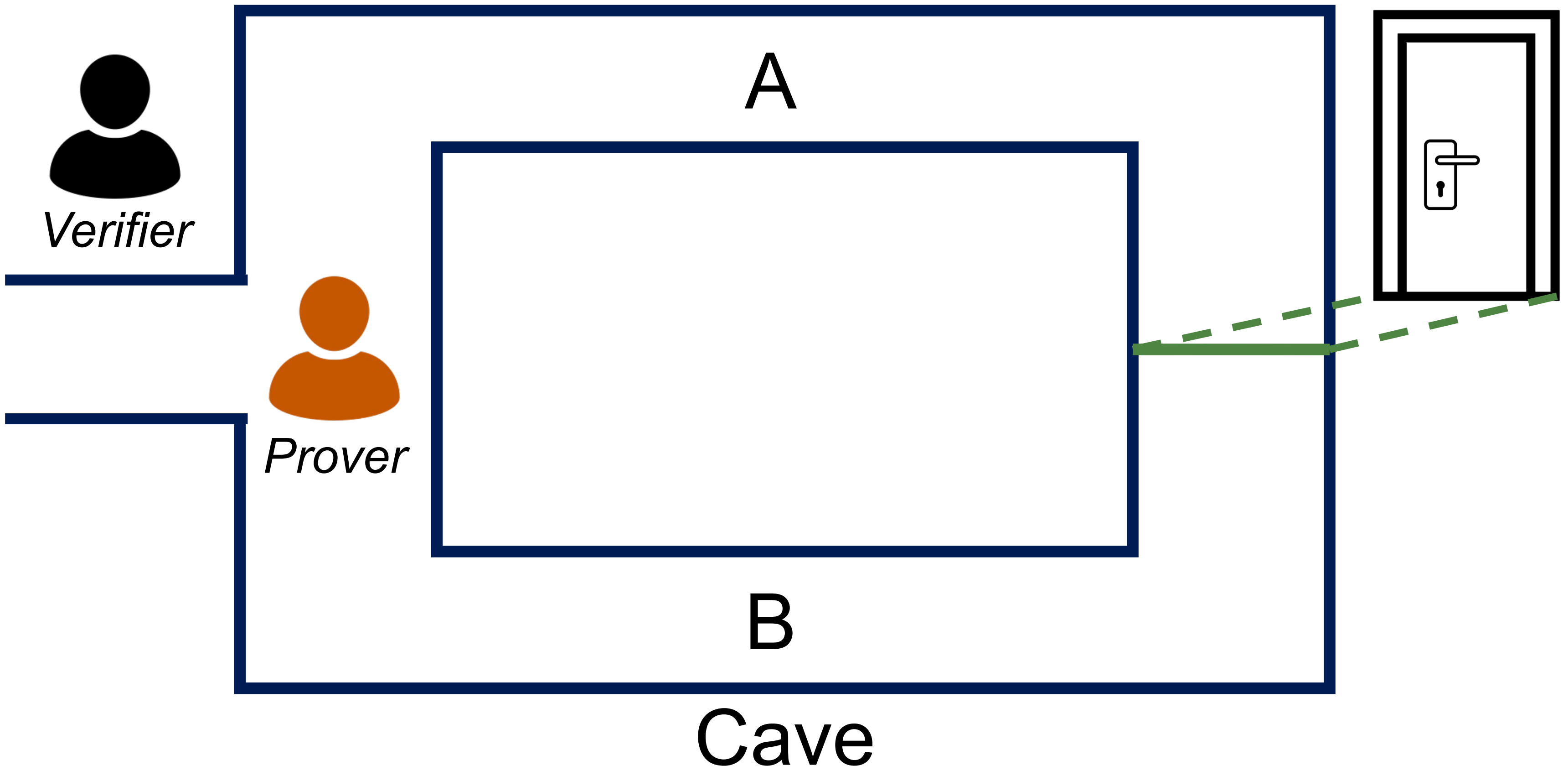
Zero-Knowledge Proof

- A method by which the *prover* can prove to *verifier* that they know a secret, without revealing anything about the secret



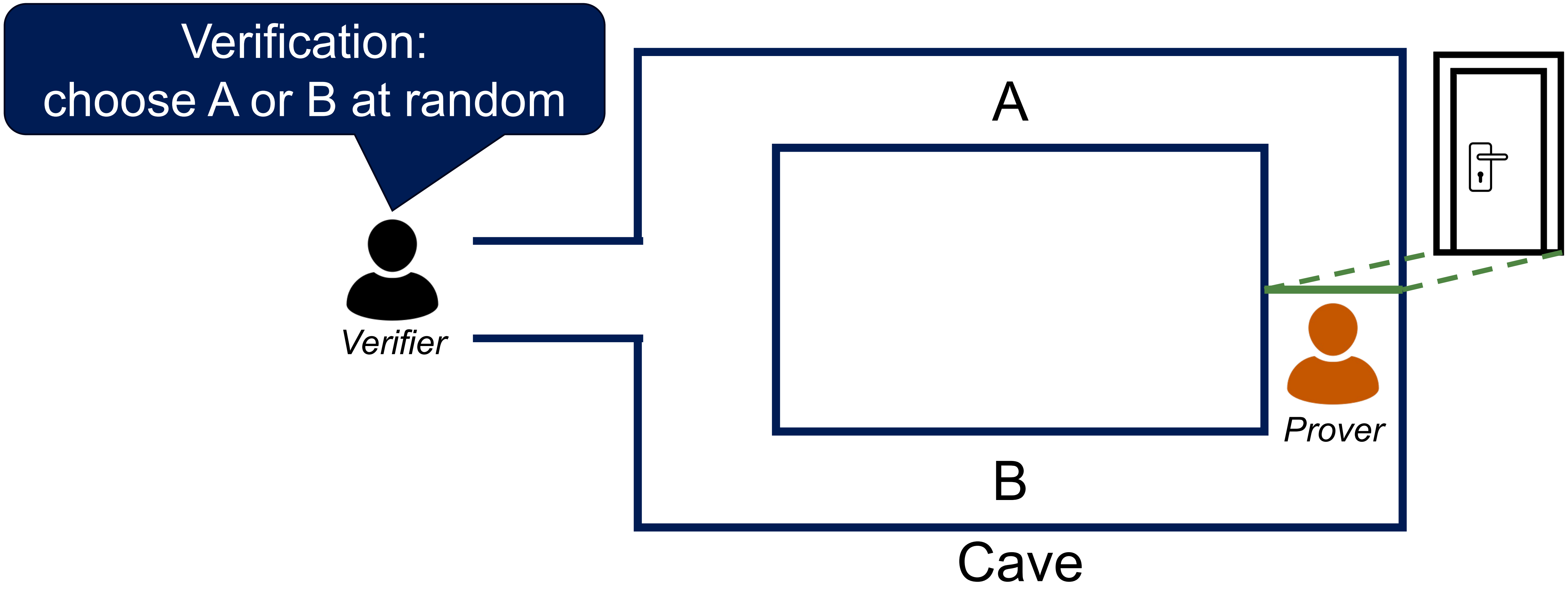
Zero-Knowledge Proof

- A method by which the *prover* can prove to *verifier* that they know a secret, without revealing anything about the secret



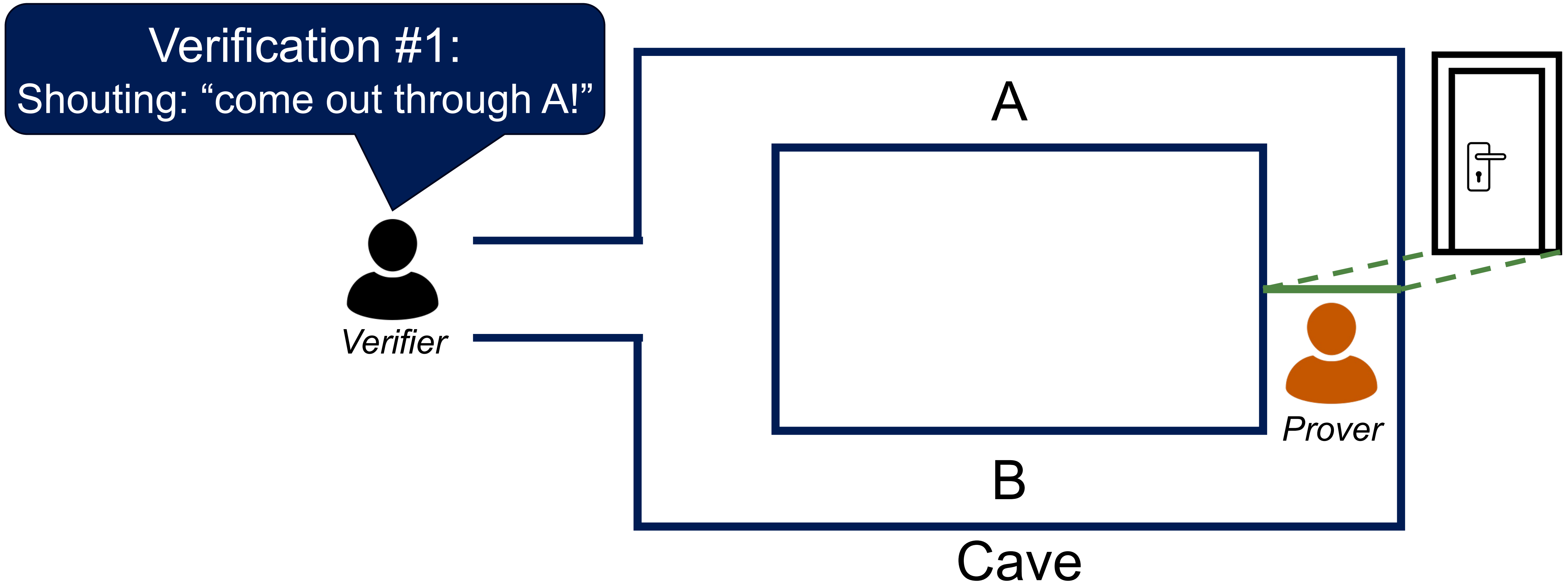
Zero-Knowledge Proof

- A method by which the *prover* can prove to *verifier* that they know a secret, <u>without revealing anything about the secret</u>



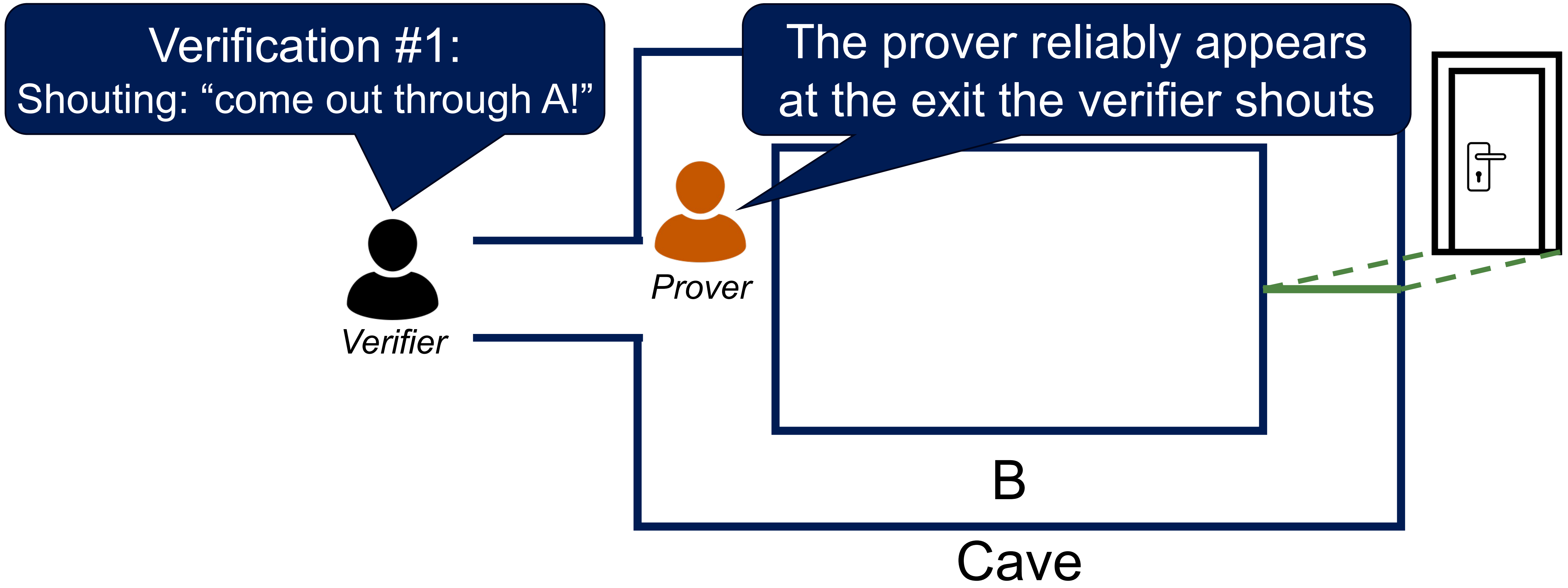
Zero-Knowledge Proof

- A method by which the *prover* can prove to *verifier* that they know a secret, without revealing anything about the secret



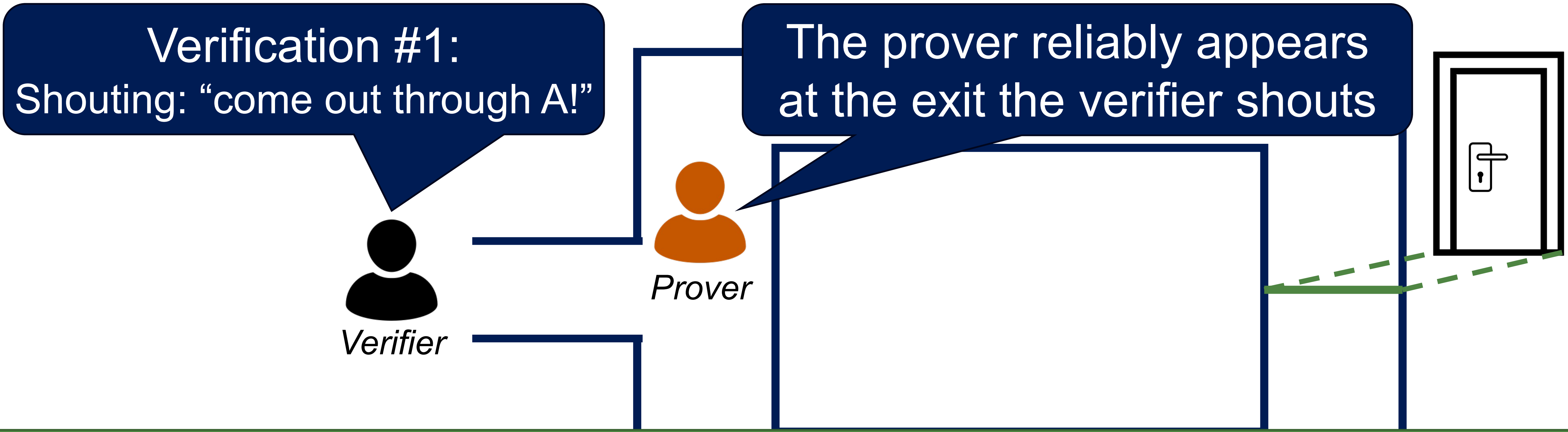
Zero-Knowledge Proof

- A method by which the *prover* can prove to *verifier* that they know a secret, without revealing anything about the secret


Verification #1:
Shouting: “come out through A!”
The prover reliably appears at the exit the verifier shouts
Prover
Verifier
B
Cave

Zero-Knowledge Proof

- A method by which the *prover* can prove to *verifier* that they know a secret, without revealing anything about the secret


Verification #1:
Shouting: “come out through A!”
The prover reliably appears at the exit the verifier shouts
Prover
Verifier

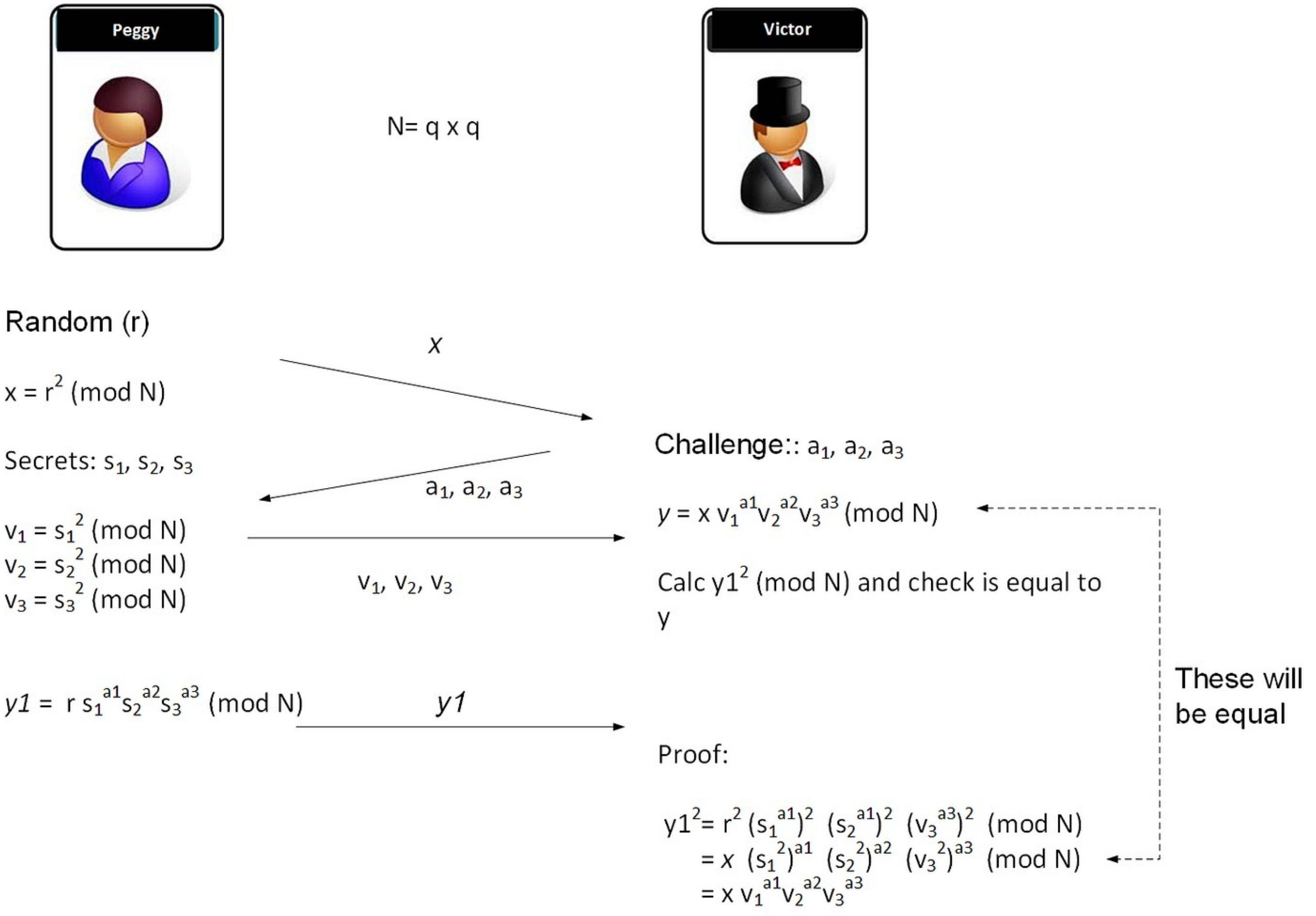
If the *Prover* **repeatedly appears** at the exit where the *Verifier* shouts, the *Verifier can* conclude that the *Prover* knows the secret

Zero-Knowledge Proof

- Chance of lucky guess
  - 1st trial: 1:2
  - 2nd trial: 1:4
  - 3rd trial: 1:8
  - 4th trial: 1:16
  - 5th trial: 1:32
  - …
  - 100th trial: 1:7.89 x 10^{-31}

Zero-Knowledge Proof – Use Case

- Fiat-Shamir Interactive Identification



Random (r)

$x = r^2 \pmod N$

Secrets: s_1, s_2, s_3

$v_1 = s_1^2 \pmod N$
$v_2 = s_2^2 \pmod N$
$v_3 = s_3^2 \pmod N$

$y1 = r\, s_1^{a1} s_2^{a2} s_3^{a3} \pmod N$

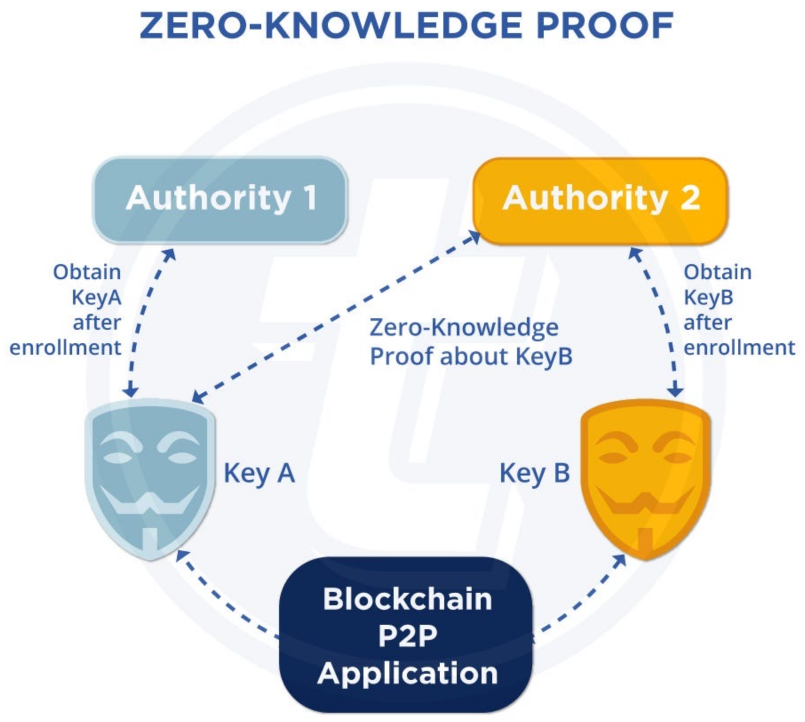
Challenge:: a_1, a_2, a_3

$y = x\, v_1^{a1} v_2^{a2} v_3^{a3} \pmod N$

Calc $y1^2 \pmod N$ and check is equal to y

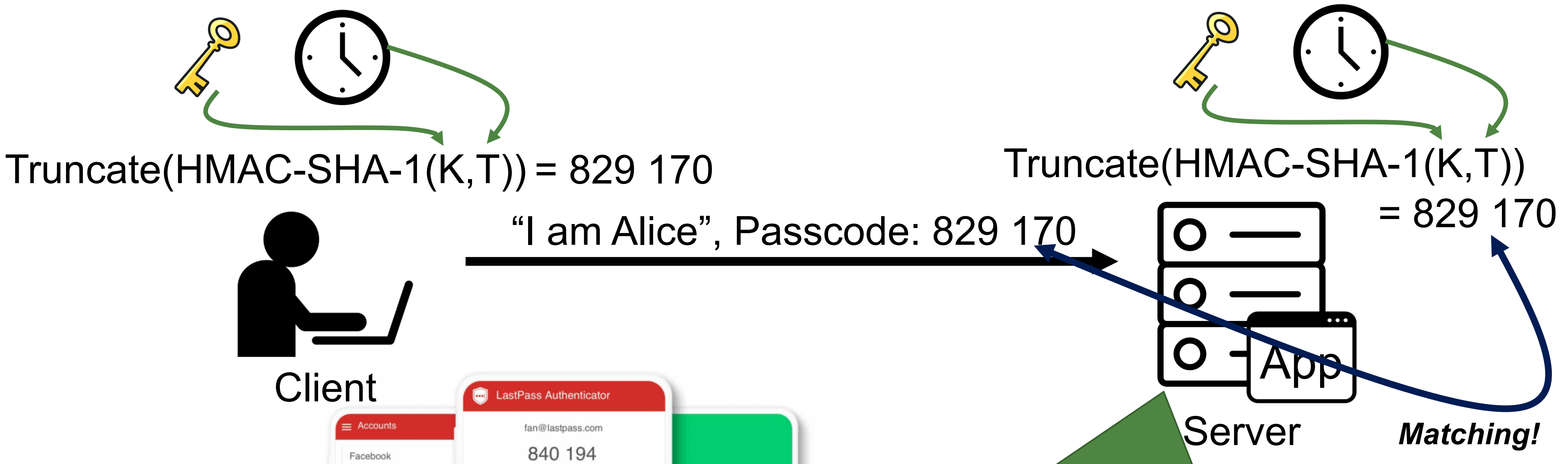
Proof:

$$\begin{aligned} y1^2 &= r^2 (s_1^{a1})^2\ (s_2^{a1})^2\ (v_3^{a3})^2 \pmod N \\ &= x\ (s_1^{2})^{a1}\ (s_2^{2})^{a2}\ (v_3^{2})^{a3} \pmod N \\ &= x\, v_1^{a1} v_2^{a2} v_3^{a3} \end{aligned}$$

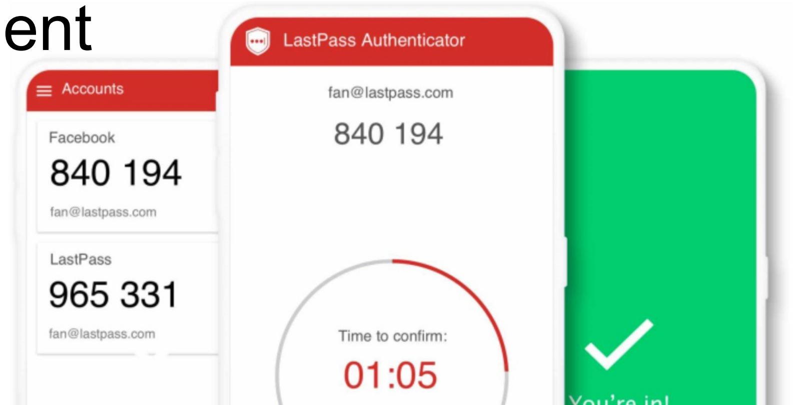

ZERO-KNOWLEDGE PROOF
Authority 1
Authority 2
Obtain KeyA after enrollment
Obtain KeyB after enrollment
Zero-Knowledge Proof about KeyB
Key A
Key B
Blockchain P2P Application

Time-based Authentication

- Client and server use [shared seed (secret key) + current time] to generate a passcode



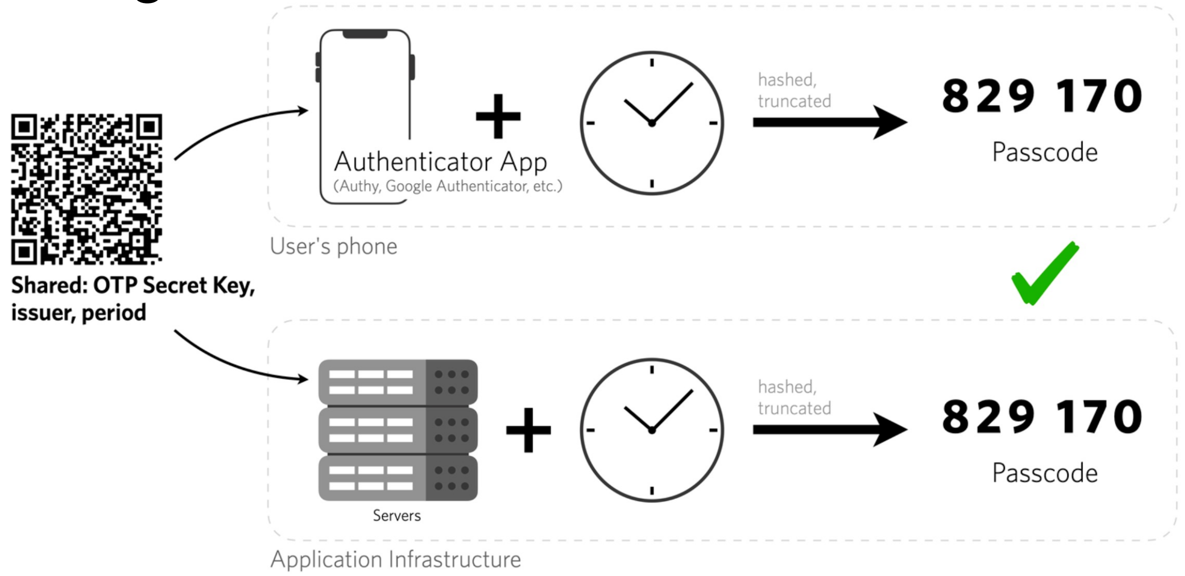
Truncate(HMAC-SHA-1(K,T)) = 829 170
Truncate(HMAC-SHA-1(K,T)) = 829 170
“I am Alice”, Passcode: 829 170
Client
App
Server
Matching!



Server need not send any random challenge to the user

Time-based Authentication

- Client and server use [shared seed (secret key) + current time] to generate a passcode
- Various methods are available for users to receive time-based one-time passwords
  - Hardware security tokens that display the password on a small screen;
  - Mobile authenticator apps, such as Google Authenticator;



- Resynchronization options
  - Default time step of 30 seconds
  - Allow for client-clocks being slightly slower / slightly faster

Outline

- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

Outline

- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

Recap: Certificate-based Authentication

Outline

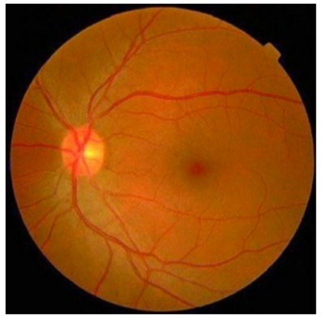
- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

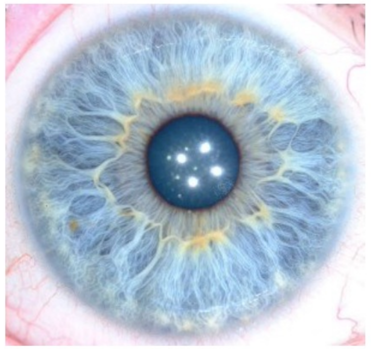
Outline

- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

Biometrics Authentication – Something You Are

- Biometrics = Bio + metric
- The measurement and statistical analysis of biological dat
  - Fingerprints
  - Palms
  - Face
  - Iris/Retina scanning
  - Voice
  - How you walk? How you type? ..
    - Research in continuous authentication



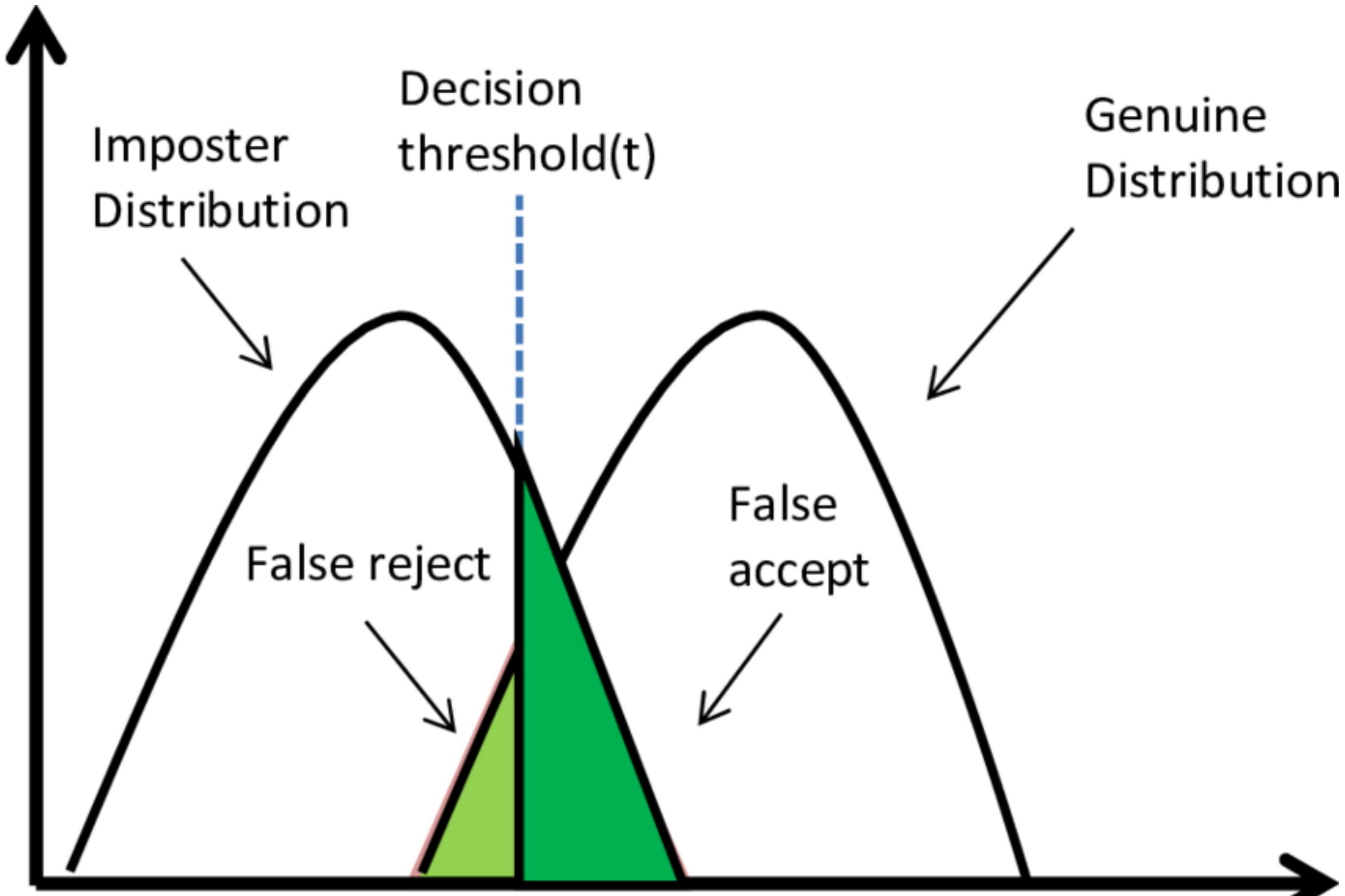


Biometrics: Pros and Cons

- Pros
  - Nothing to remember
  - Passive (nothing to type, always carrying them around)
  - Can’t share
  - Can be fairly unique
- Cons
  - Revocability
    - You can change a password but how do you change your fingerprint?
  - Are still spoofable
    - E.g., Pick fingerprints from objects and create molds
  - Cost
    - Need special devices to read them
  - **Error rates** (Major difference with something you know/have)
    - Probability of you being you, rather than certainty

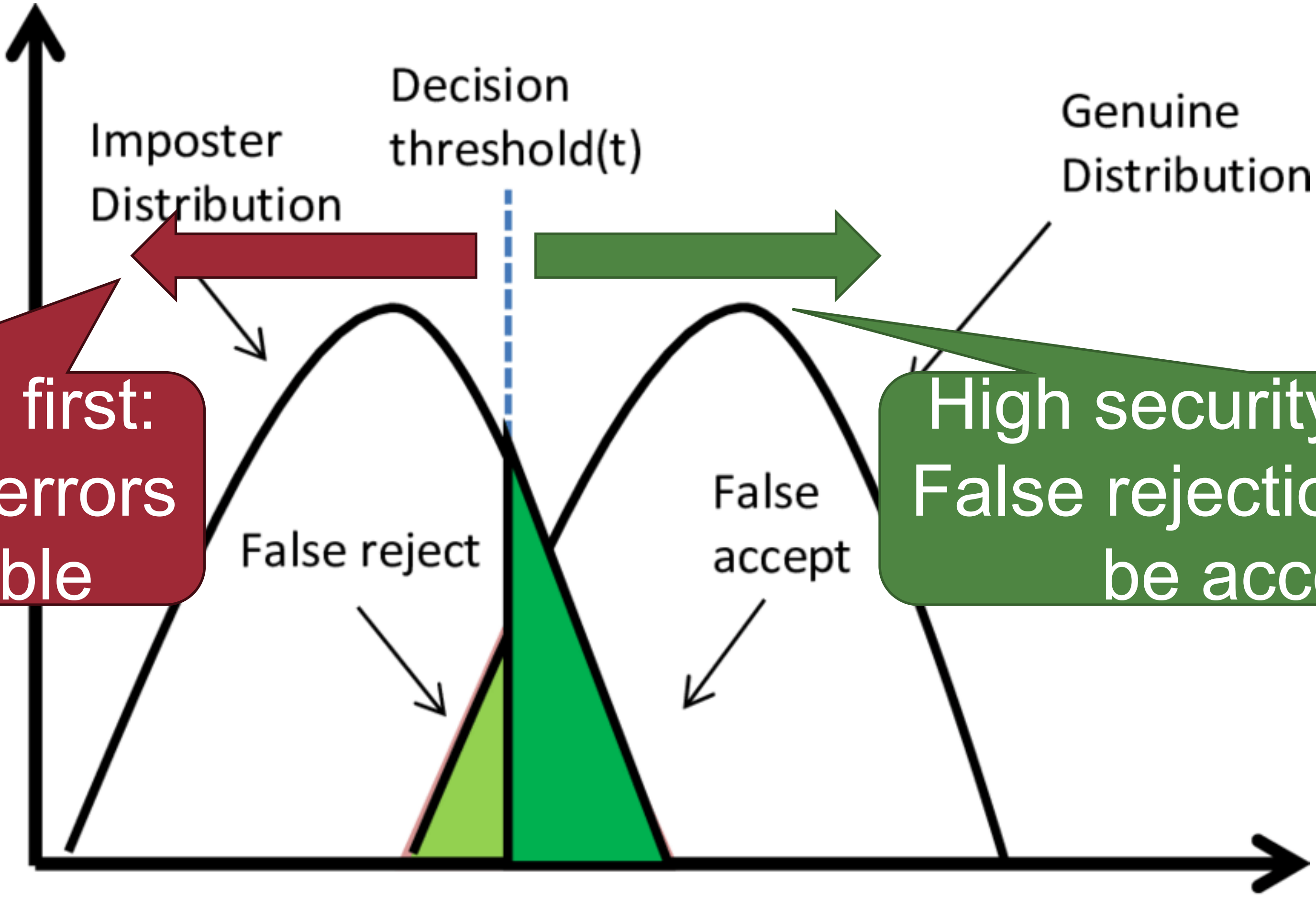
Biometric Error Rates

- False acceptance rate: system accepts a forgery
- False rejection rate: system rejects valid user



Biometric Error Rates

- False acceptance rate: system accepts a forgery
- False rejection rate: system rejects valid user



User convenience first: False acceptance errors may be acceptable

High security application: False rejection errors may be acceptable

Outline

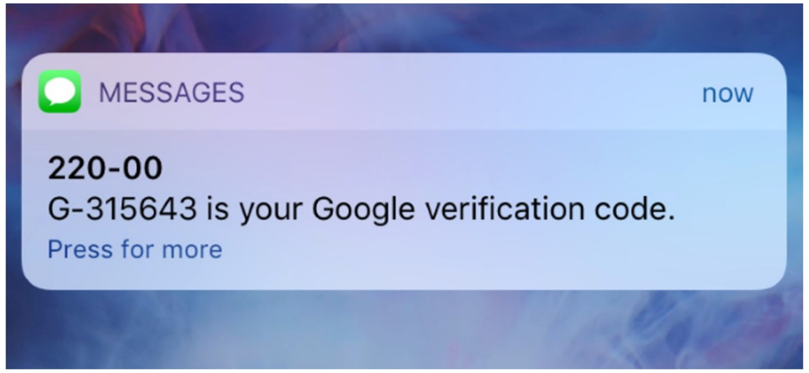
- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

Outline

- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos (skip)

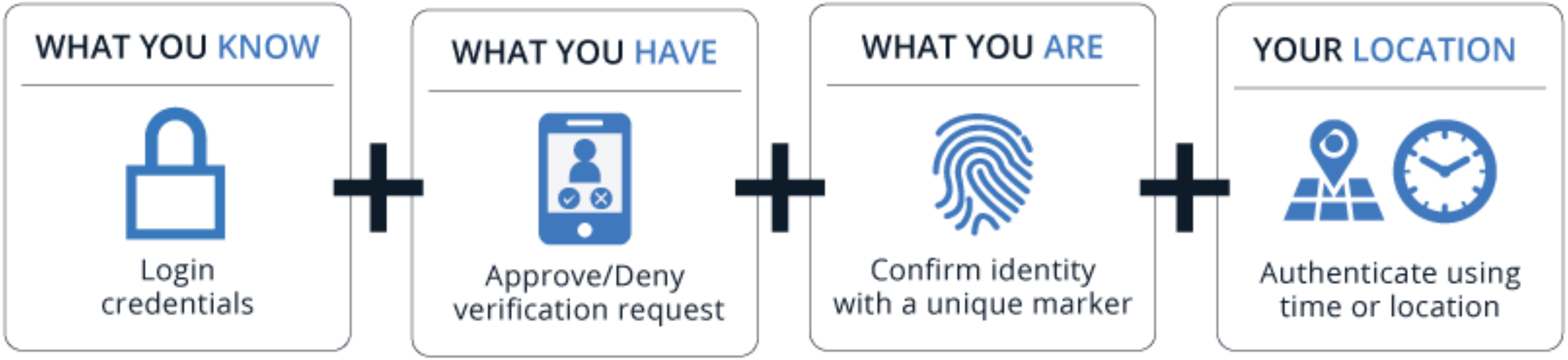
Multi-factor Authentication (MFA)

- A combination of criteria that need to be met
  - To strengthen the overall security of a system


MESSAGES now

220-00
G-315643 is your Google verification code.
Press for more

- E.g., 2 factor authentication: password (what you know) + phone (what you have)


WHAT YOU KNOW
Login credentials
+
WHAT YOU HAVE
Approve/Deny verification request
+
WHAT YOU ARE
Confirm identity with a unique marker
+
YOUR LOCATION
Authenticate using time or location

Summary: User-aspects

- Never forget that users are a critical part of securing an infrastructure
  - No matter how good your technology is, users can still ruin everything if someone convinces them that it is “okay”
- Abusing the trust of users: **social engineering or phishing**
  - We will never ask you for your password over email!
- Prevention:
  - Educating your employees
  - Setting up standard procedures

Conclusion

- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication