CSE610: Web Programming & Security

11. Password

Seongil Wi

Department of Computer Science and Engineering

Paper Presentation

- Presentation Time: 30 mins (+ QnA 5 mins)
- Check your presentation date on the website!

- Evaluation:
  - Organization/clarity
  - Quality of you criticism (You should present your opinion!)
  - Presentation skills

  + Participation points will be awarded to students asking valuable questions!

- You should start presentation with a summary of the paper
  - Problem, Goal, Contribution, and Evaluation

Midterm Exam

- April. 18 (Thursday)
- Class Time (1h 15m)

- Descriptive type questions
- Closed book

Project Checkpoint Report

- Due: April. 26 (Friday), 11:59 PM

- You should upload a single PDF file on BlackBored
  - If your team consists of two people, each member must submit a PDF file

- Add the progress made thus far in **your proposal**
  - You must write your progress/modified part in **blue font**

- The quantity and quality of progress will also be evaluated, so please write carefully!

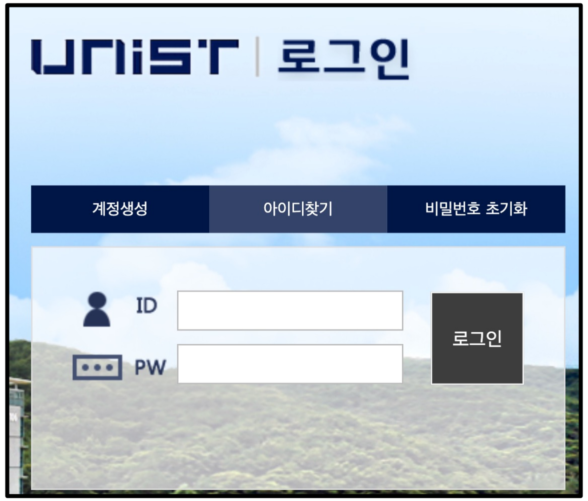
Authentication – Who Are You?

- 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)



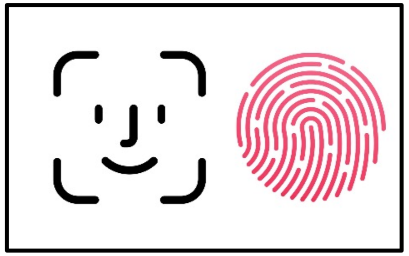
UNIST | 로그인

계정생성 | 아이디찾기 | 비밀번호 초기화

ID

PW

로그인



Types of the Authentication

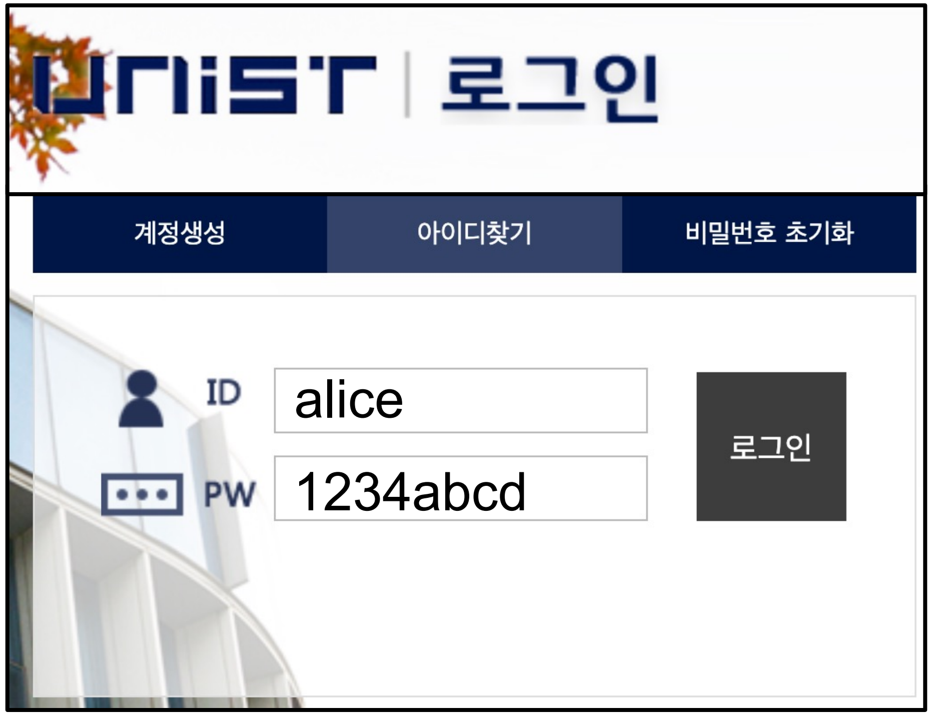
- Password-based authentication
- Token-based authentication
- Certificate-based authentication
- Biometric authentication
- Multi-factor authentication
- Kerberos
- …

Password-based Authentication – Something You Know

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

Clear Text Password

ID	Password
alice	1234abcd
bob	verysecure
charlie	1234abcd


UNIST | 로그인

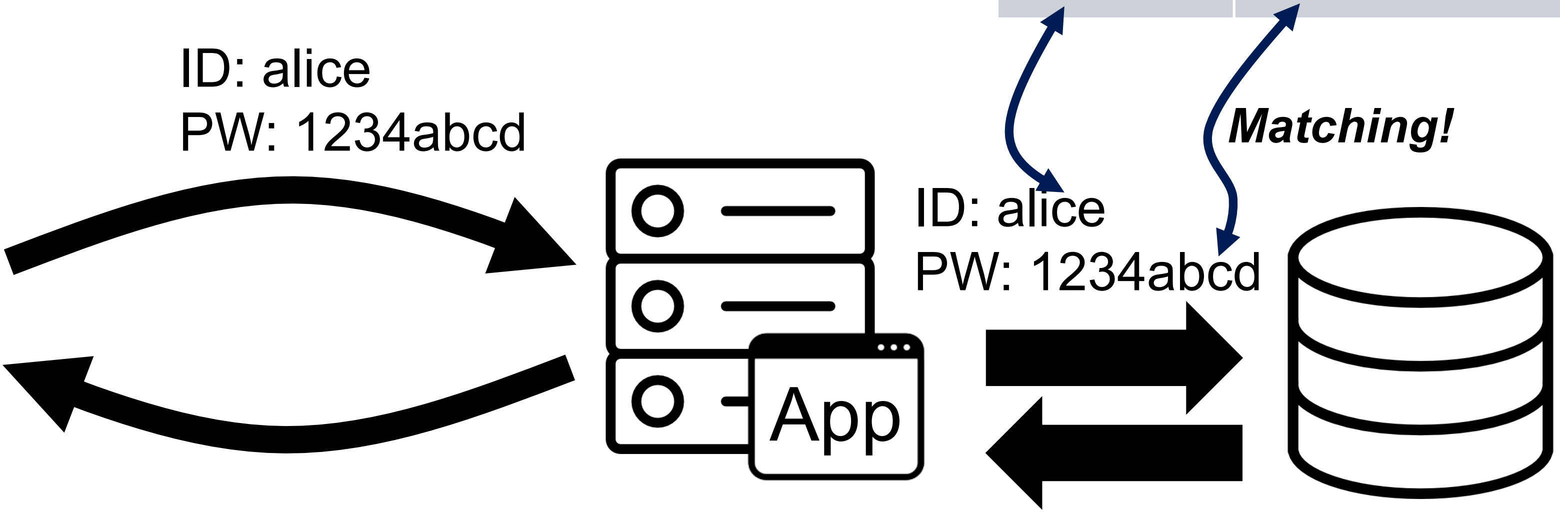
계정생성 | 아이디찾기 | 비밀번호 초기화

ID: alice

PW: 1234abcd

로그인

Browser

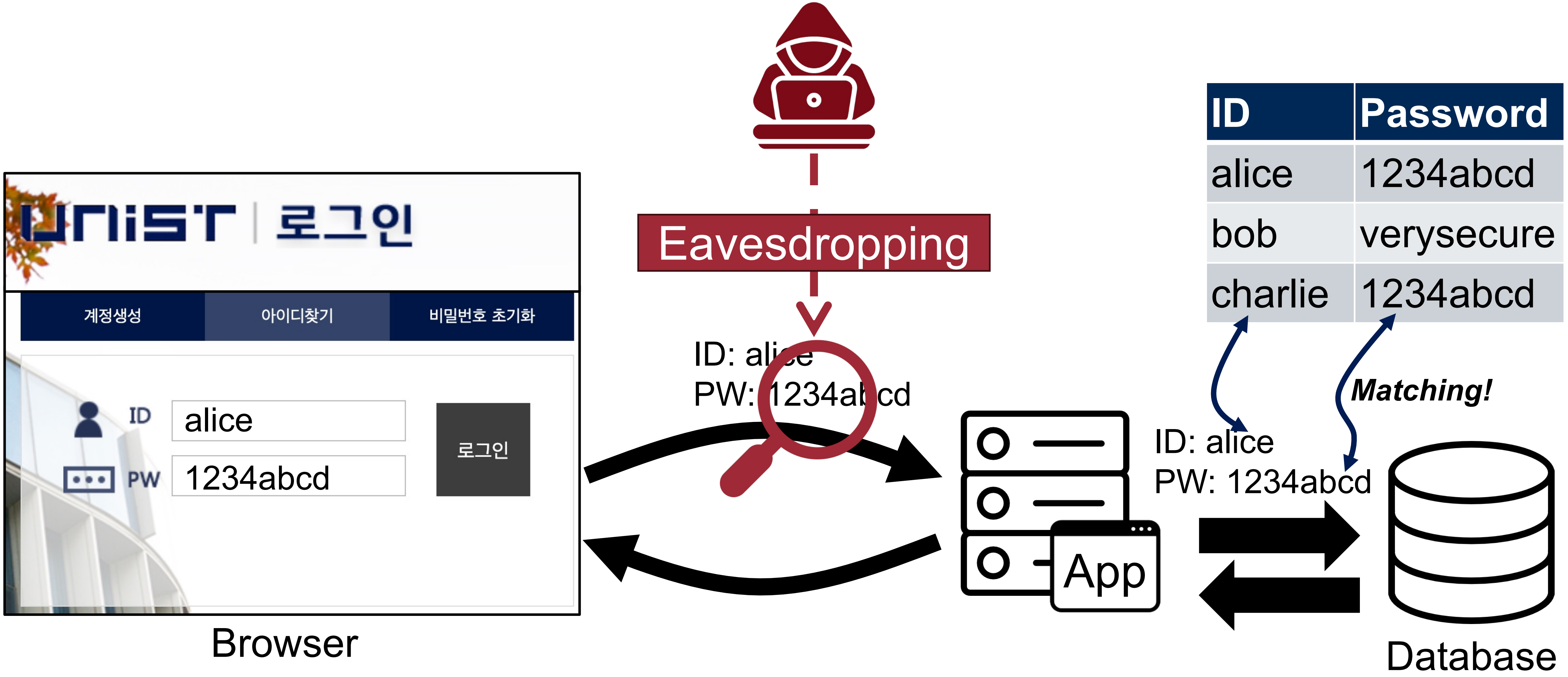

ID: alice
PW: 1234abcd

ID: alice
PW: 1234abcd

Matching!

Database

Problems of Clear Text Password?



UNIST | 로그인

ID alice

PW 1234abcd

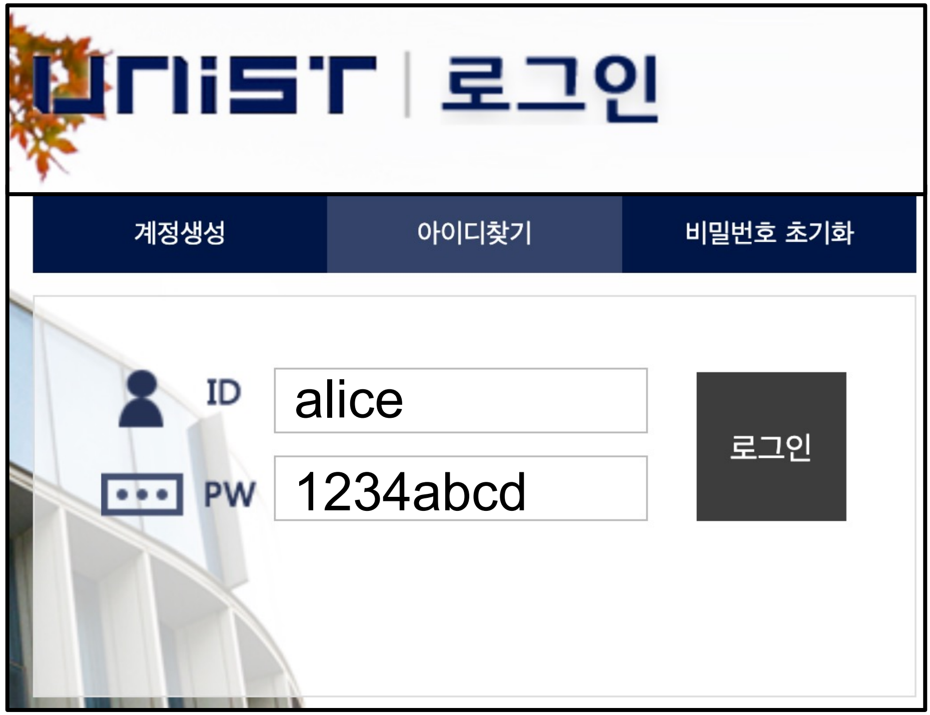
ID: alice
PW: 1234abcd

ID: alice
PW: 1234abcd

Matching!

ID	Password
alice	1234abcd
bob	verysecure
charlie	1234abcd

SSL/TLS Encryption! Are We Safe Now?



UNIST | 로그인

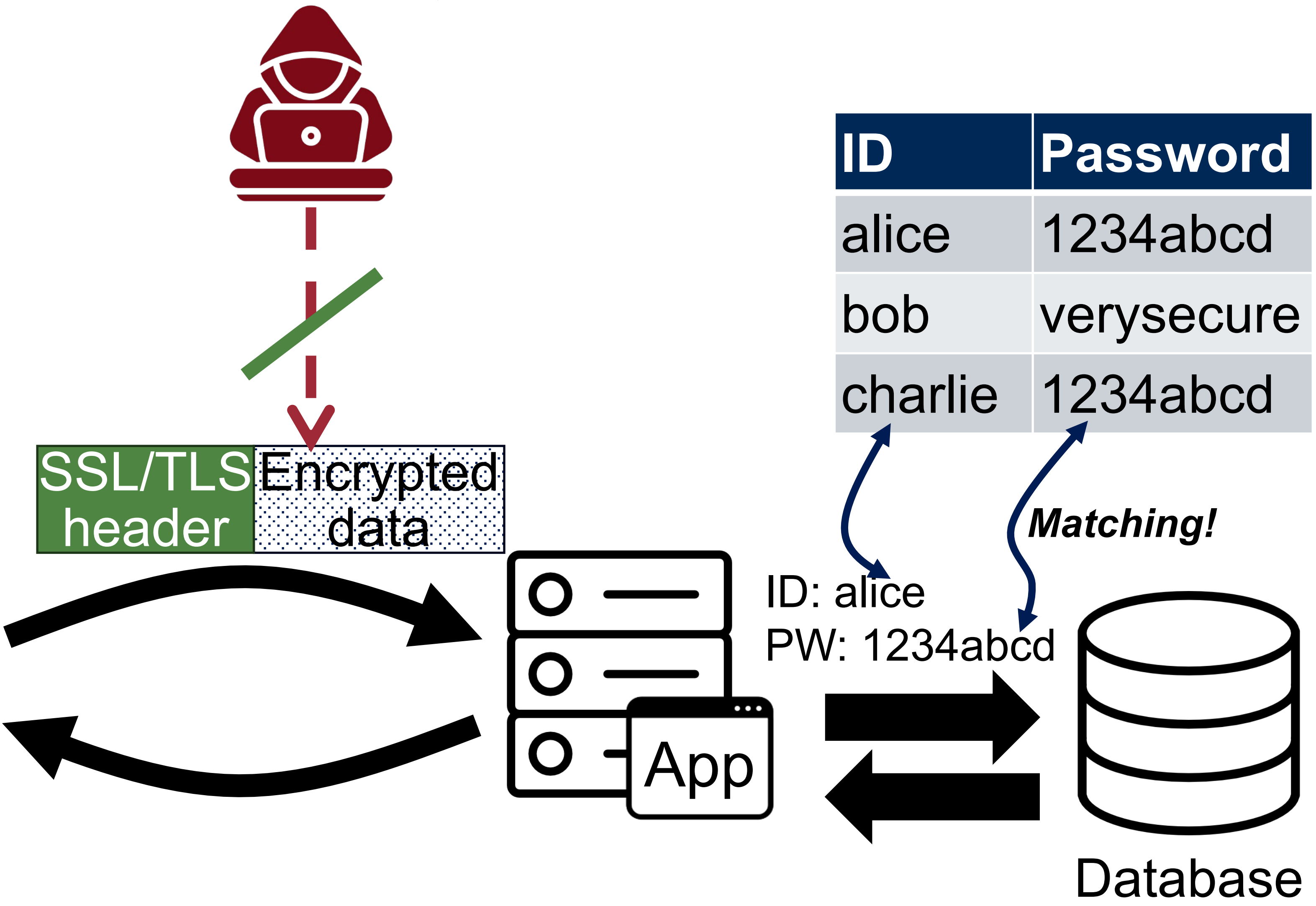
계정생성 | 아이디찾기 | 비밀번호 초기화

ID: alice

PW: 1234abcd

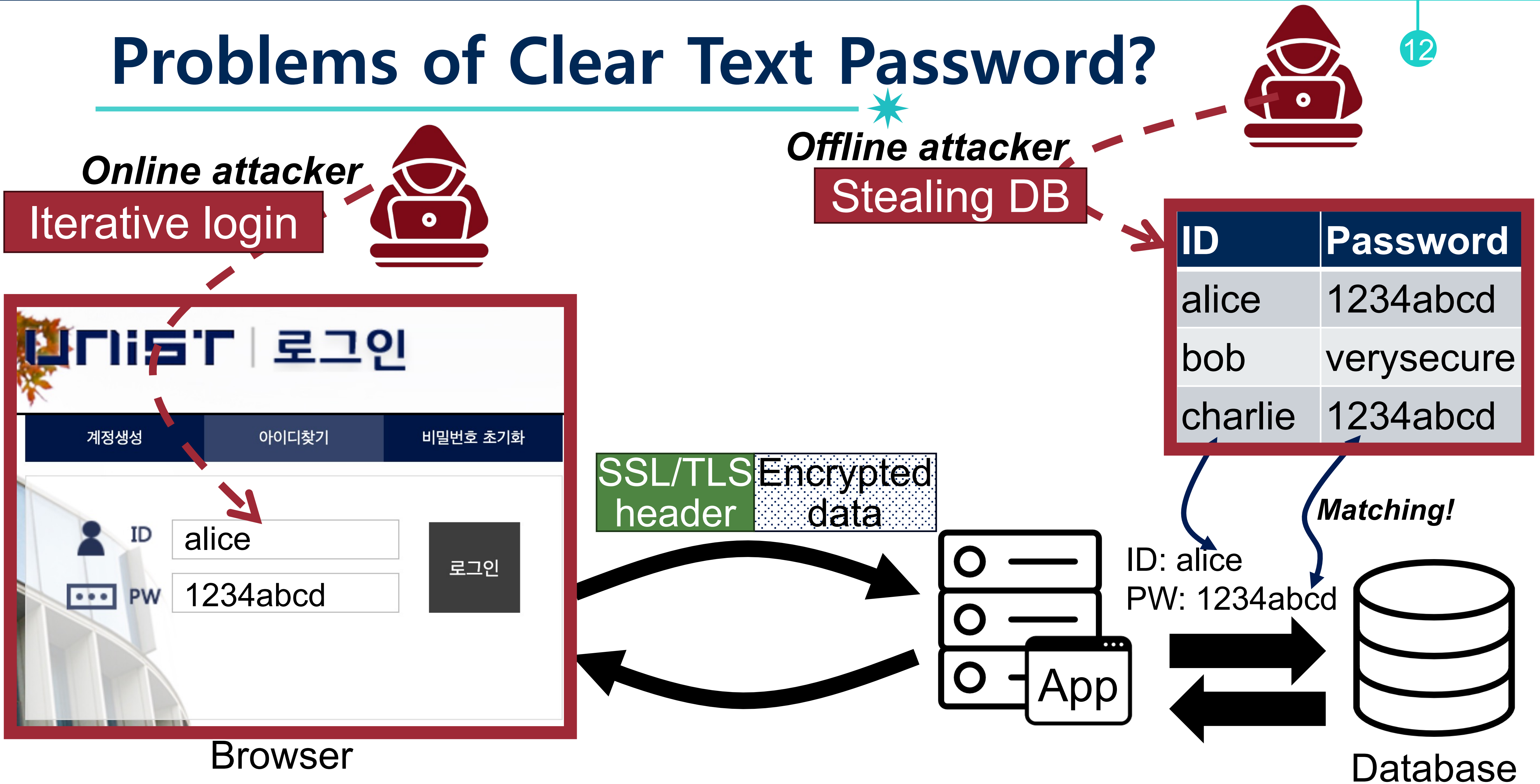
로그인

Browser



ID	Password
alice	1234abcd
bob	verysecure
charlie	1234abcd

Problems of Clear Text Password?



Online attacker

Iterative login

Offline attacker

Stealing DB

ID	Password
alice	1234abcd
bob	verysecure
charlie	1234abcd

UNIST | 로그인

계정생성 | 아이디찾기 | 비밀번호 초기화

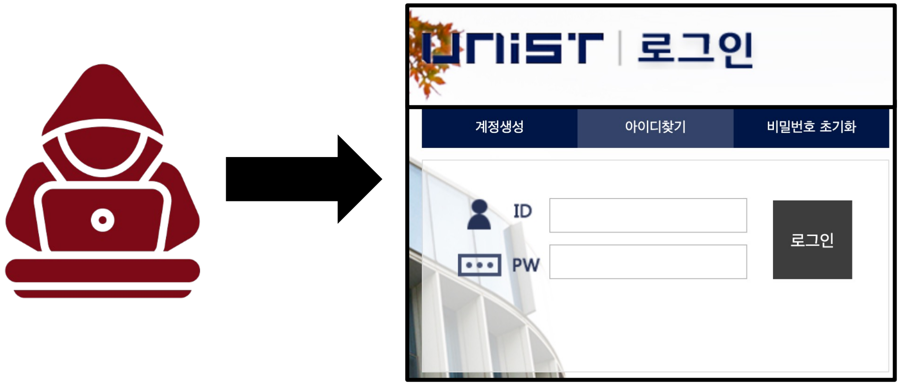
ID: alice

PW: 1234abcd

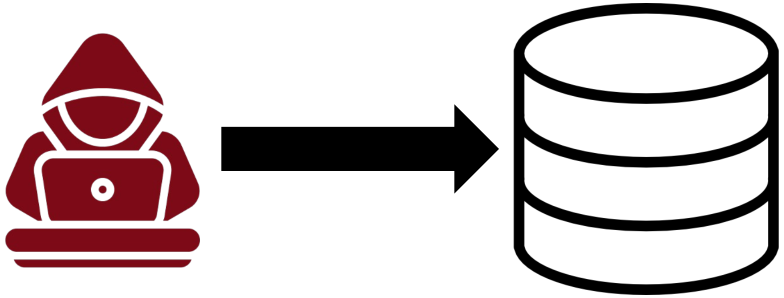
로그인

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?

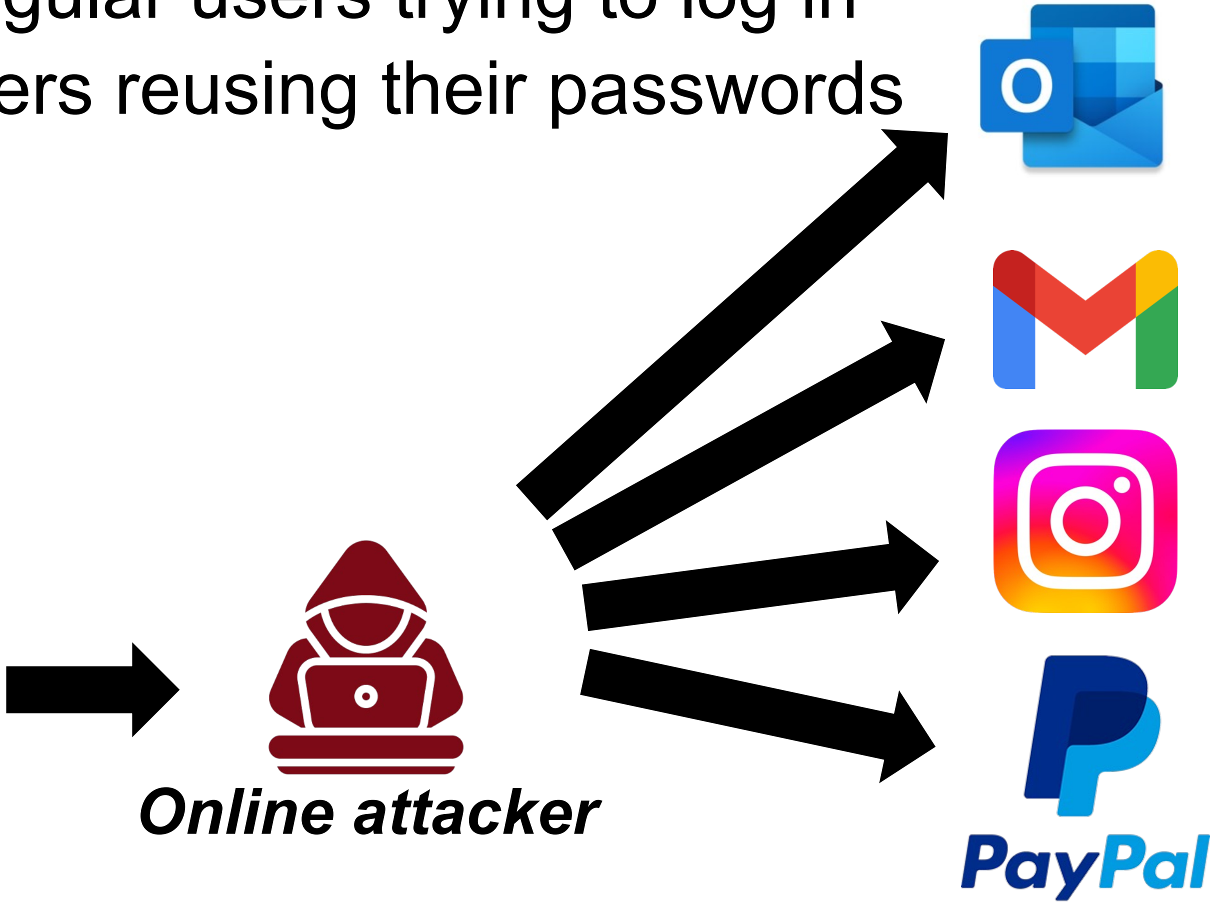
- Once a database of credentials is leaked, attackers can use them in multiple ways
  - Extract emails and usernames
  - Learn what are the most common passwords that most users use
  - Learn what are the passwords that specific users use

ID	Password
alice	1234abcd
bob	verysecure
charlie	1234abcd

Example: 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

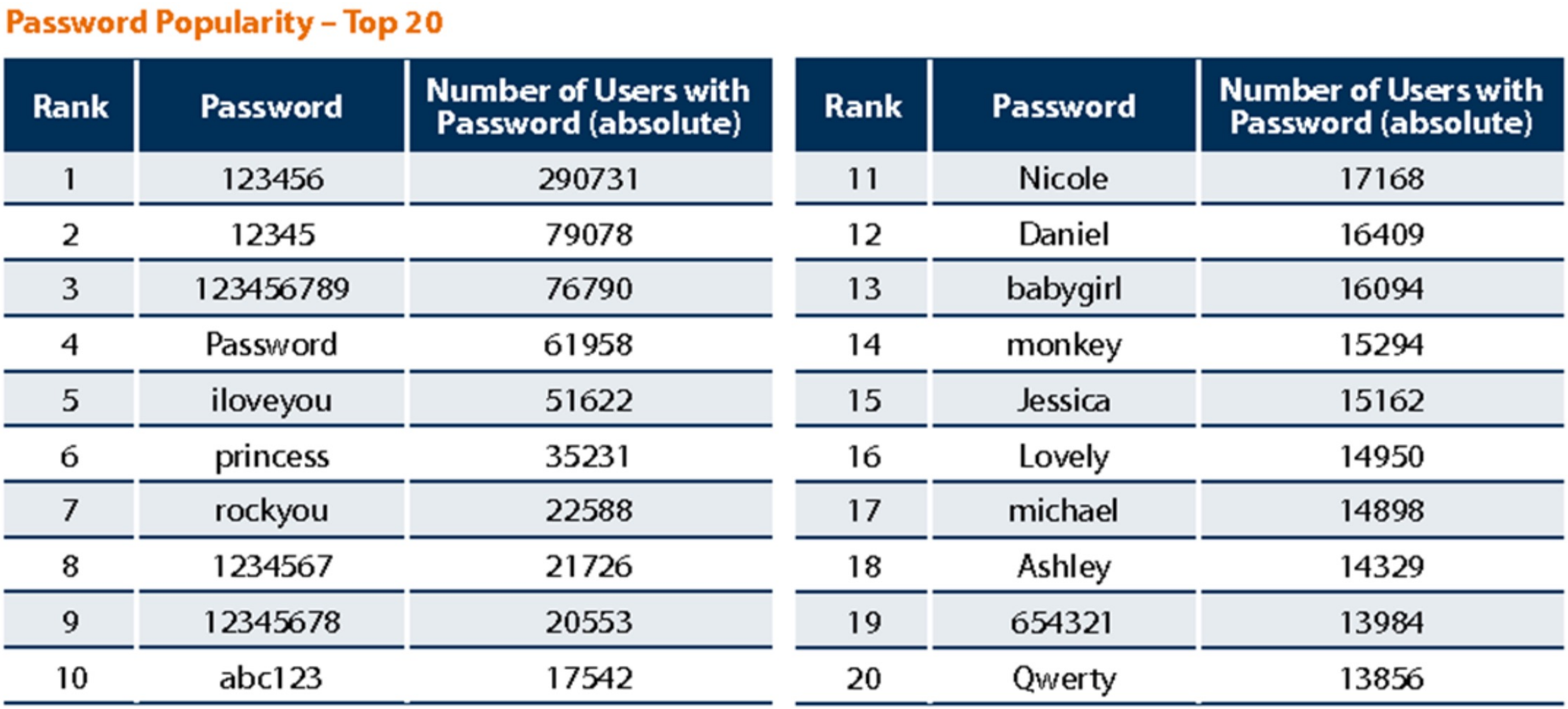
ID	Password
alice	1234abcd
bob	verysecure
charlie	1234abcd



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

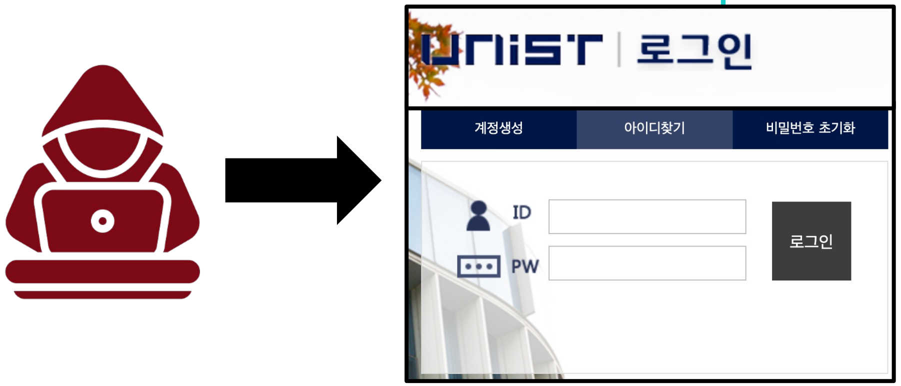
Passwords in RockYou Database



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

Defense for Online Attacker



UNIST | 로그인

계정생성 아이디찾기 비밀번호 초기화

ID

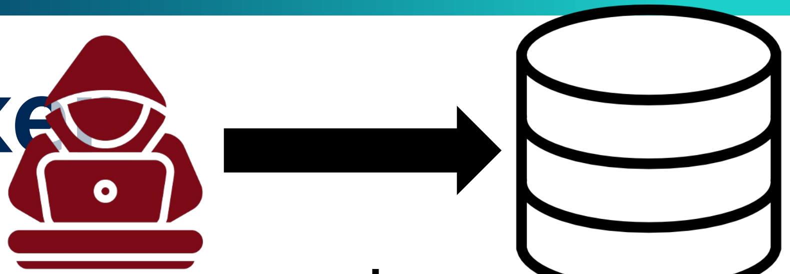
PW

로그인

- 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)

Defense for Offline Attacke



- 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?

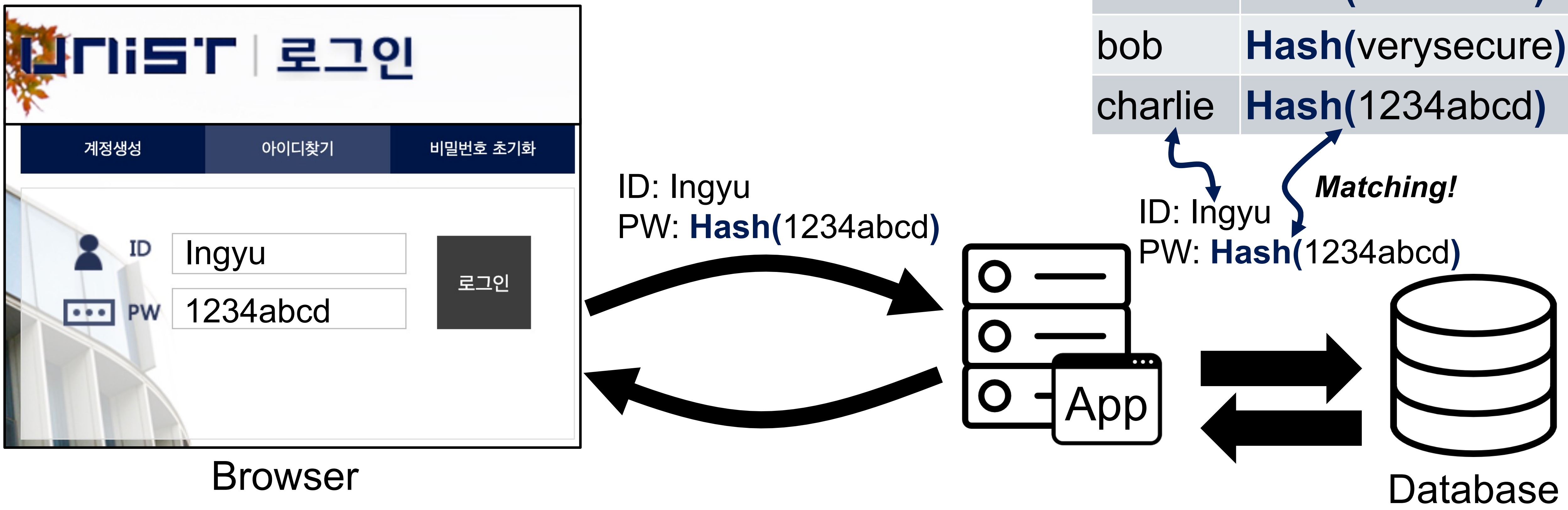
- How about encrypting each password with a secret key (e.g. only stored in the memory of the server) which is used to decrypt any single entry, on demand?

- 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
alice	**Hash(**1234abcd**)**
bob	**Hash(**verysecure**)**
charlie	**Hash(**1234abcd**)**



UNIST | 로그인

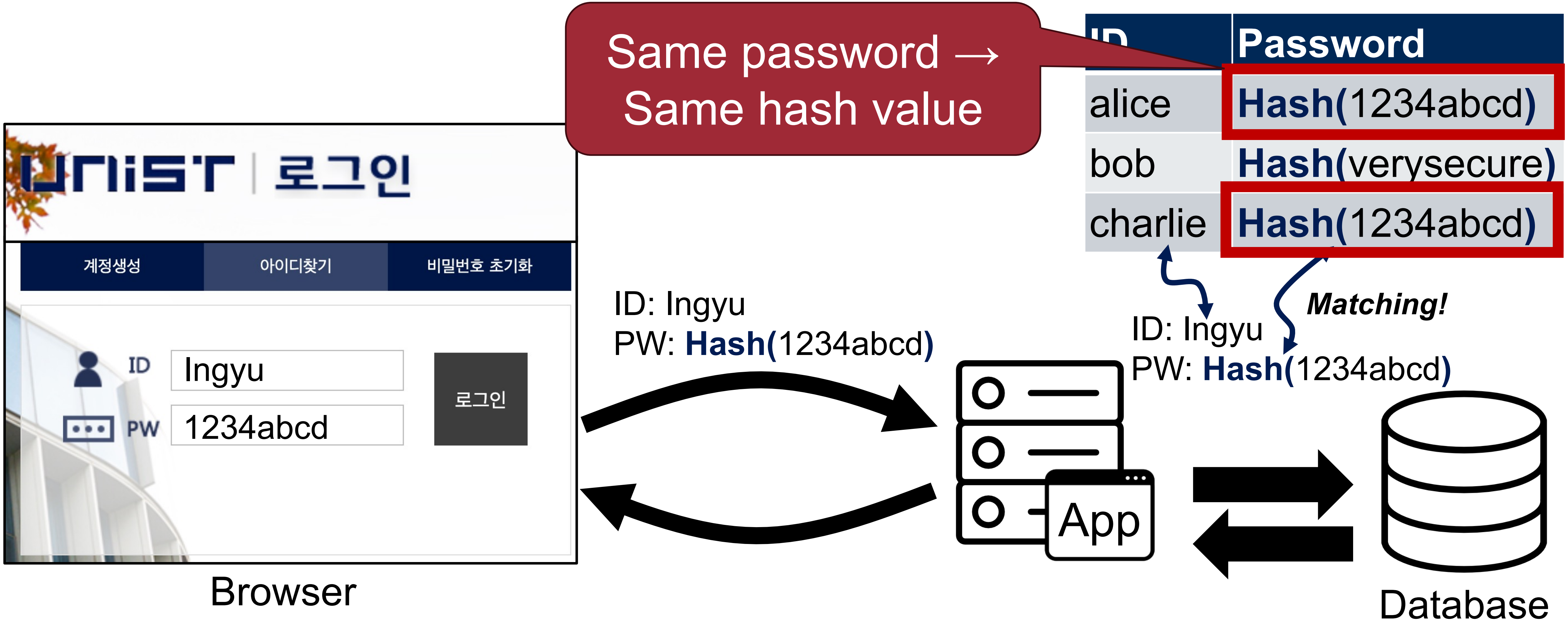
ID: Ingyu

PW: 1234abcd

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

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

Problems of Password Hashing?



Same password → Same hash value

ID	Password
alice	**Hash(**1234abcd**)**
bob	**Hash(**verysecure**)**
charlie	**Hash(**1234abcd**)**

UNIST | 로그인

계정생성 | 아이디찾기 | 비밀번호 초기화

ID: Ingyu

PW: 1234abcd

로그인

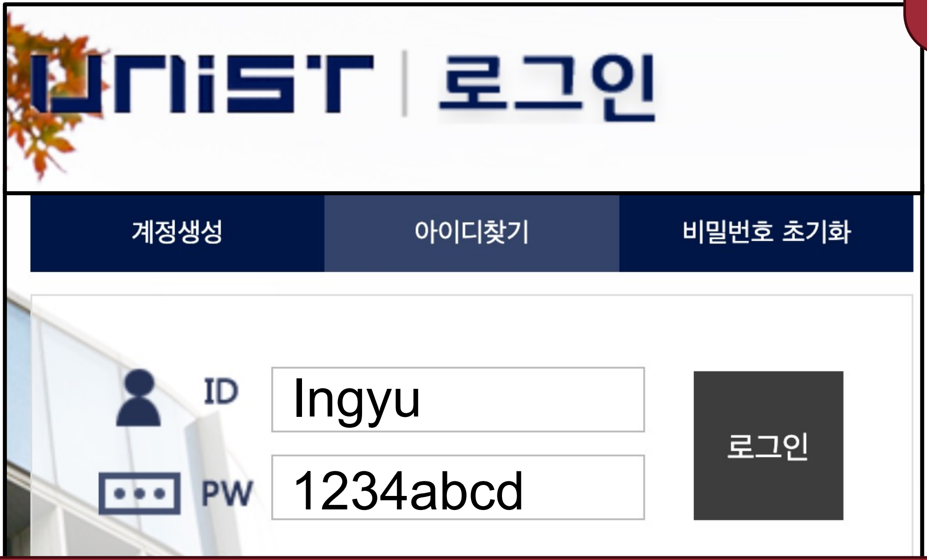
Browser

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

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

Matching!

Problems of Password Hashing?



UNIST | 로그인

ID Ingyu

PW 1234abcd

Same password → Same hash value

ID	Password
alice	**Hash(**1234abcd**)**
bob	**Hash(**verysecure**)**
charlie	**Hash(**1234abcd**)**

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

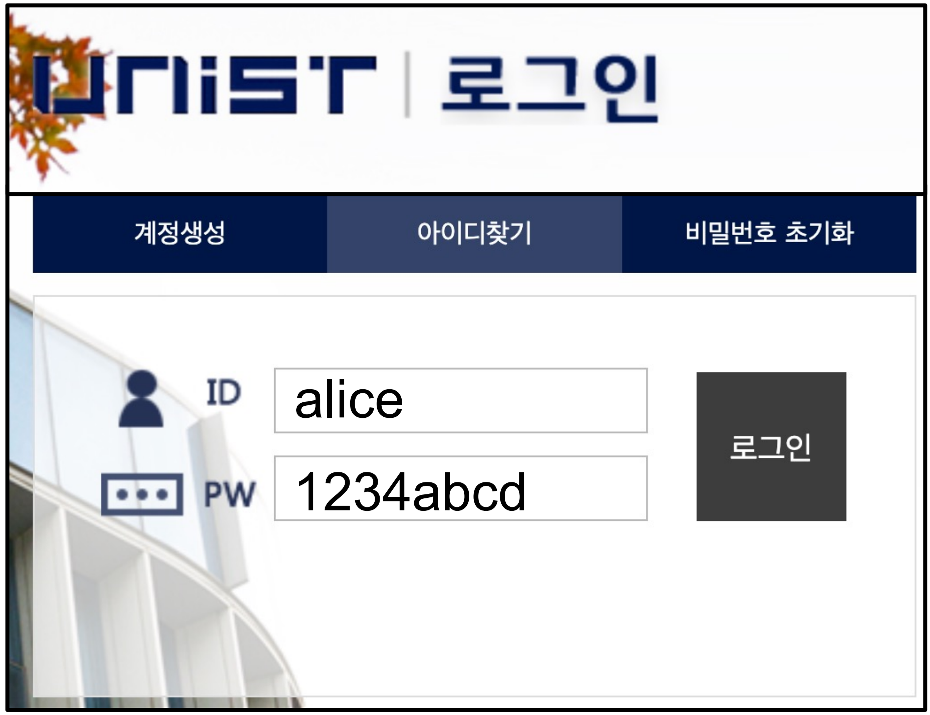
ID: Ingyu
PW: **Hash(**1234abcd**)**

Matching!

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

Recap: Salted Hash

ID	Salt	Password
alice	23	**Hash(**1234abcd, 23**)**
bob	51	**Hash(**verysecure, 51**)**
charlie	97	**Hash(**1234abcd, 97**)**



UNIST | 로그인

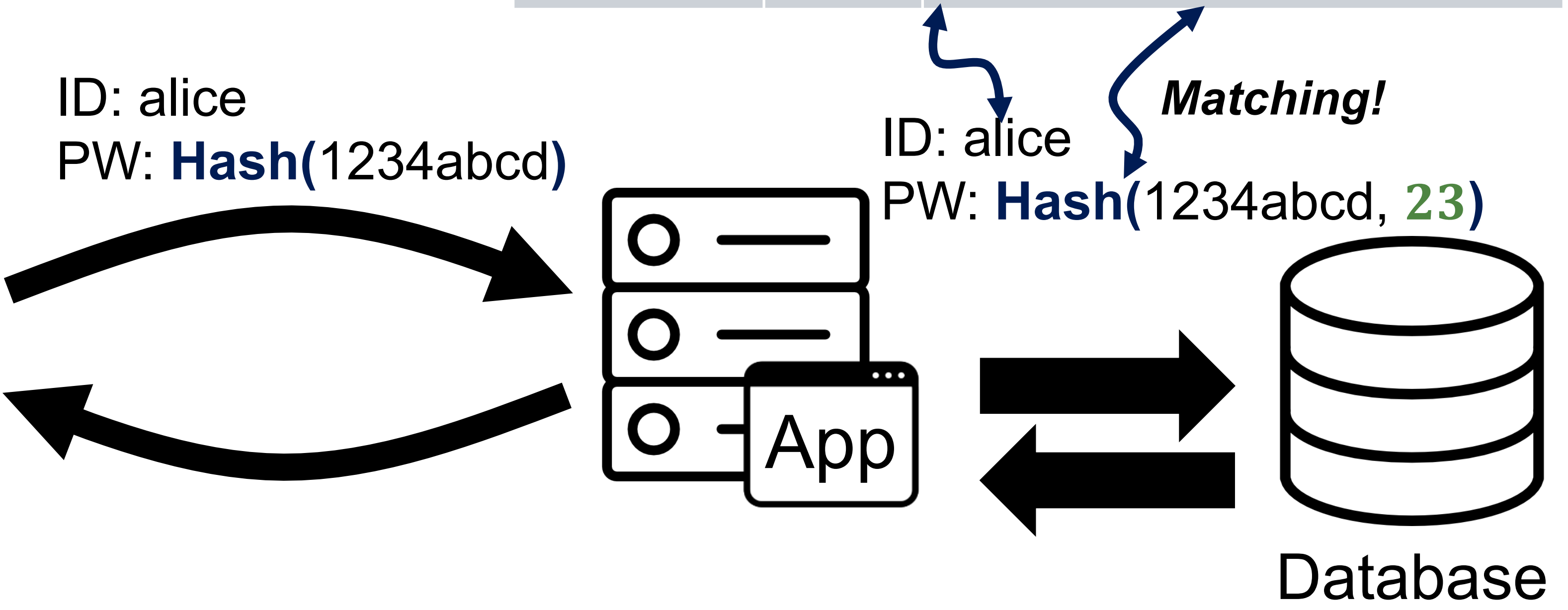
계정생성 | 아이디찾기 | 비밀번호 초기화

ID alice

PW 1234abcd

로그인

Browser



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

ID: alice
PW: **Hash(**1234abcd, 23**)**

Matching!

Recap: Salted Hash

Same password → Different hash value

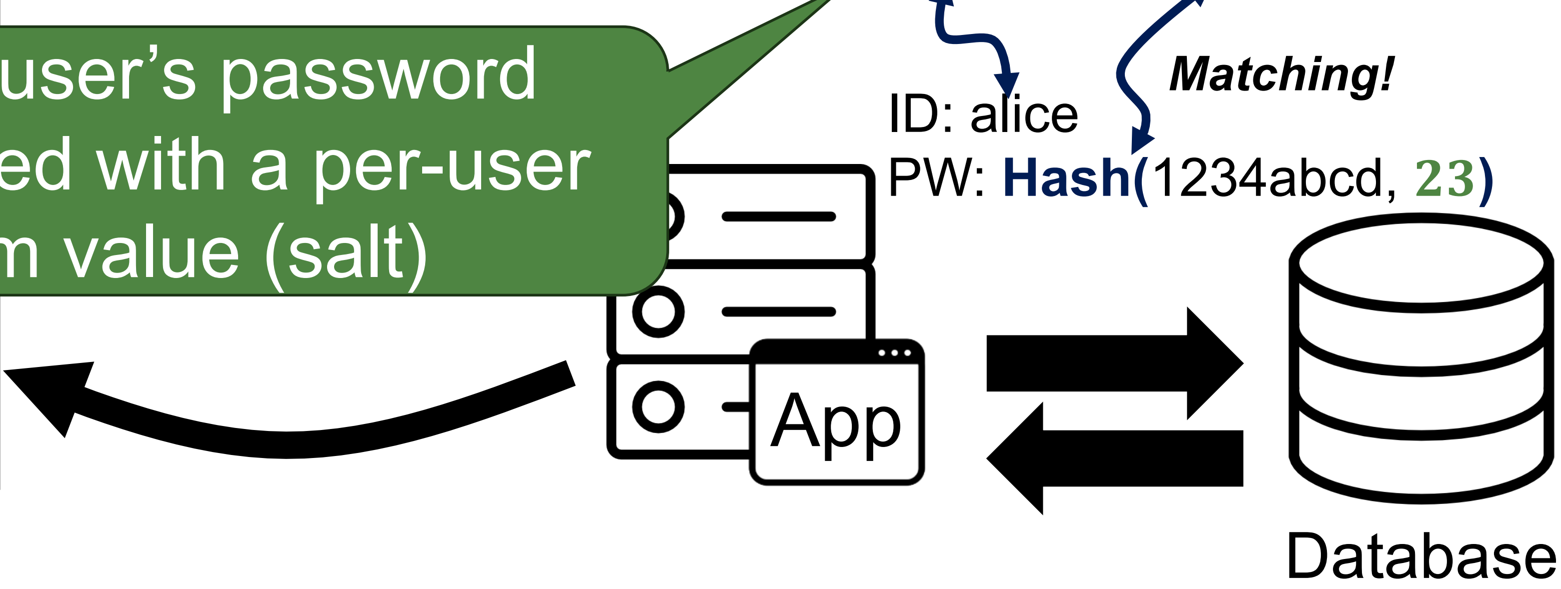
	Salt	Password
alice	23	Hash(1234abcd, 23)
bob	51	Hash(verysecure, 51)
charlie	97	Hash(1234abcd, 97)

ID alice

PW 1234abc

Browser

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



ID: alice
PW: Hash(1234abcd, 23)

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)

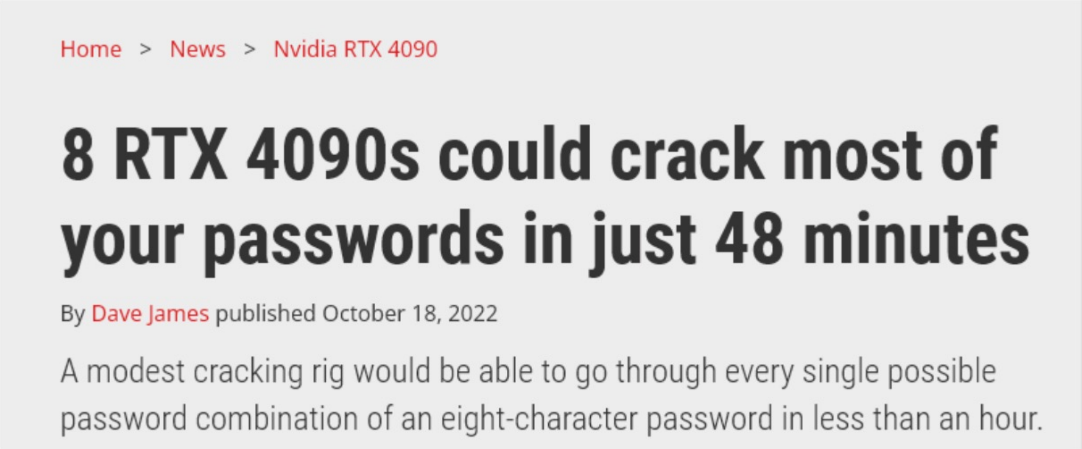
Password Guessing Techniques

- 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

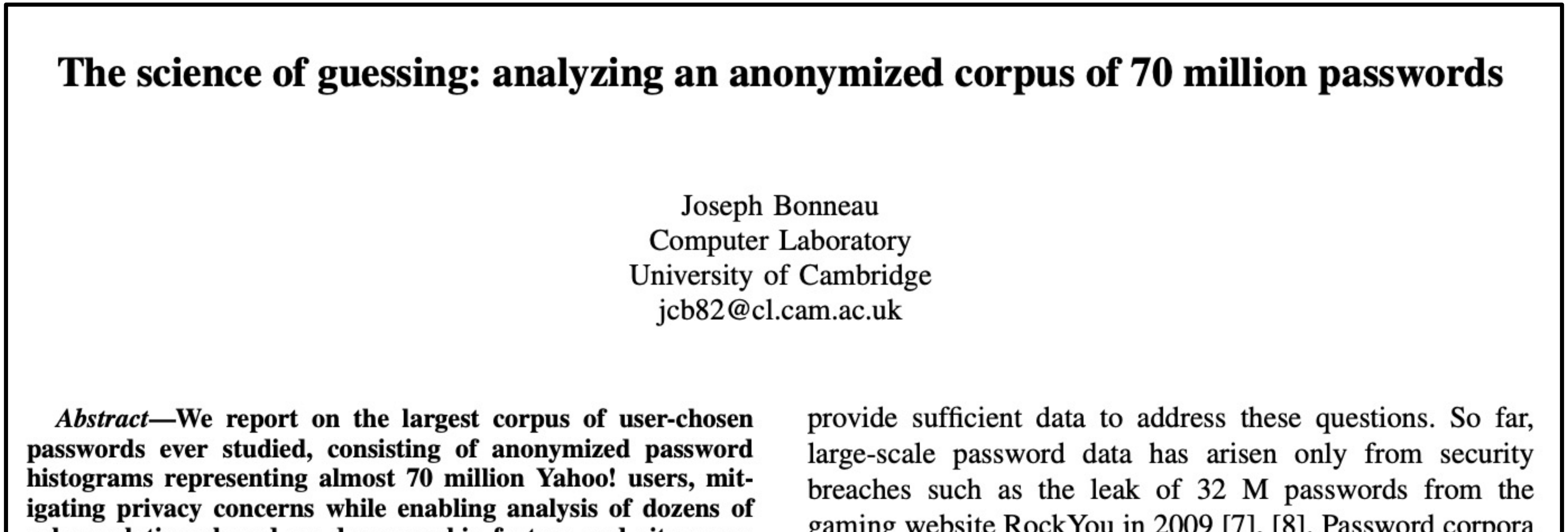
By Dave James published October 18, 2022

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

The Science of Guessing, *S&P'2012*

- Analysis of Yahoo! password data
- A measure of password distributions using Shannon Entropy
- Passwords provide roughly equivalent security of 10 bit random string guesses for large list of accounts



The science of guessing: analyzing an anonymized corpus of 70 million passwords

Joseph Bonneau
Computer Laboratory
University of Cambridge
jcb82@cl.cam.ac.uk

***Abstract*—We report on the largest corpus of user-chosen passwords ever studied, consisting of anonymized password histograms representing almost 70 million Yahoo! users, mitigating privacy concerns while enabling analysis of dozens of**

provide sufficient data to address these questions. So far, large-scale password data has arisen only from security breaches such as the leak of 32 M passwords from the gaming website RockYou in 2009 [7], [8]. Password corpora

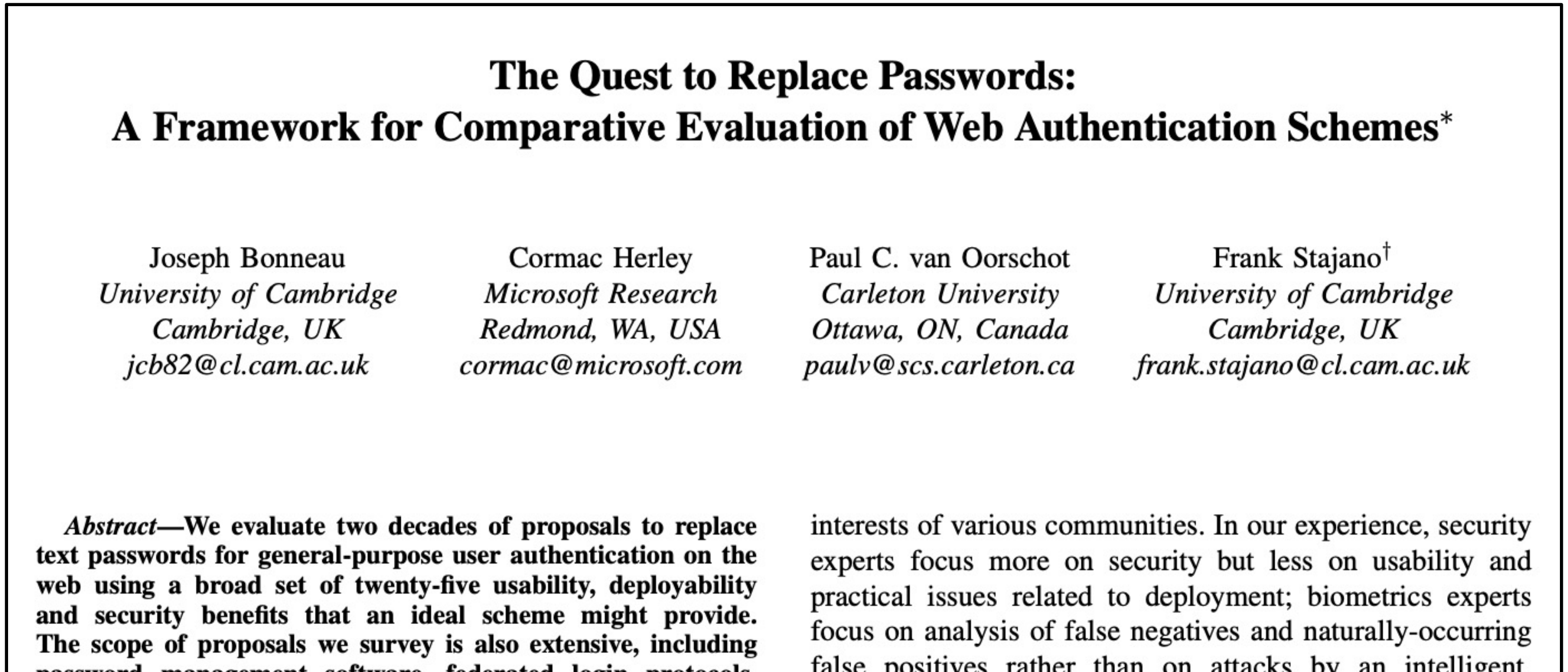
Alternatives for Password?

- There are two-decades of proposals to replace text passwords

Why are we still using passwords?

The Quest to Replace Passwords, *S&P'2012*

- The **security** is not a sole factor for adopting an authentication measure
  - Consider usability, deployability, and security
- **No known scheme** provides the full set of benefits that legacy passwords already provide



The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes*

Joseph Bonneau
University of Cambridge
Cambridge, UK
jcb82@cl.cam.ac.uk

Cormac Herley
Microsoft Research
Redmond, WA, USA
cormac@microsoft.com

Paul C. van Oorschot
Carleton University
Ottawa, ON, Canada
paulv@scs.carleton.ca

Frank Stajano[†]
University of Cambridge
Cambridge, UK
frank.stajano@cl.cam.ac.uk

***Abstract*—We evaluate two decades of proposals to replace text passwords for general-purpose user authentication on the web using a broad set of twenty-five usability, deployability and security benefits that an ideal scheme might provide. The scope of proposals we survey is also extensive, including password management software, federated login protocols**

interests of various communities. In our experience, security experts focus more on security but less on usability and practical issues related to deployment; biometrics experts focus on analysis of false negatives and naturally-occurring false positives rather than on attacks by an intelligent

Usability

- **[U1] Memorywise-Effortless**: No need to remember any secret
- **[U2] Scalable-for-Users**: Having many accounts brings no burden to users
- **[U3] Nothing-to-carry**
- **[U4] Physically-Effortless**
- **[U5] Easy-to-learn**
- **[U6] Efficient-to-use**: The time the user must spend for each authentication is acceptably short
- **[U7] Infrequent-errors**
- **[U8] Easy-Recovery-from-Loss**

Deployability

- **[D1] Accessible**: Users who can use passwords are not prevented from using the scheme from disabilities
- **[D2] Negligible-Cost-per-User**: The total cost per user of the scheme is negligible
- **[D3] Server-compatible**
- **[D4] Browser-compatible**

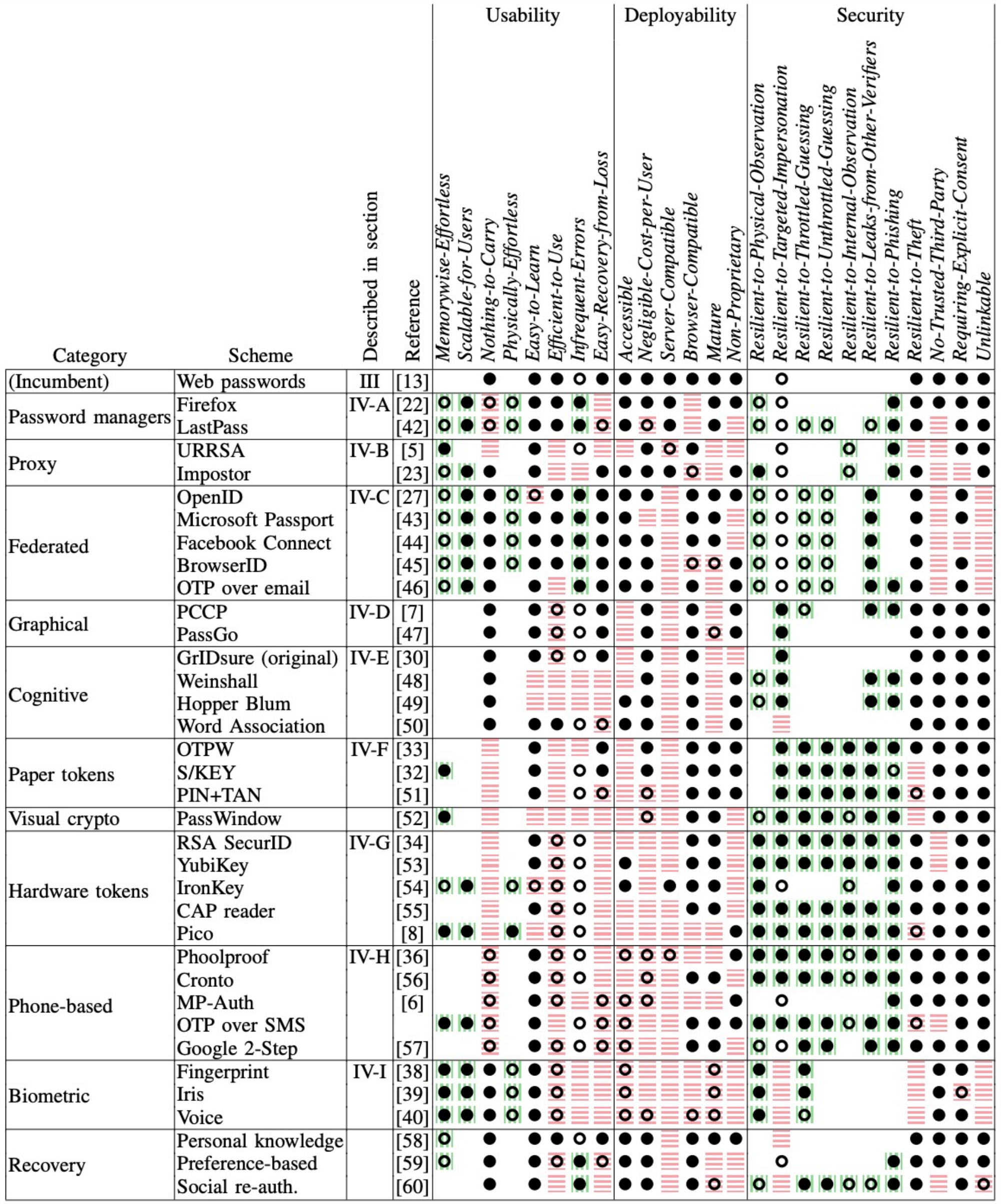
Security

- **[S1] Resilient-to-Physical-Observation**
- **[S2] Resilient-to-Targeted-Impersonation**: Impossible for an acquaintance to impersonate a user by using personal details
- **[S3] Resilient-to-Throtted-Guessing**: The attacker with a limited number of guesses should not guess the significant fraction of users
- **[S4] Resilient-to-Unthrotted-Guessing**: Offline attacker with enough computing power should not compromise large # of users
- **[S5] Resilient-to-Internal-Observation**: The attacker who intercepts user input cannot compromise the account
- **[S6] Resilient-to-Leaks-from-Other-Verifiers**: Nothing for a verifier to leak

Security

- **[S7] Resilient-to-Phishing**: Cannot use harvested credentials later to impersonate a victim (It don’t include MITM attack)
- **[S8] Resilient-to-Theft**
- **[S9] No-Trusted-Third-Party**: Don’t rely on a trusted-third party
- **[S10] Requiring-Explicit-Consent**
- **[S11] Unlinkable**: Colluding verifiers cannot determine from the authenticator alone

Let’s Evaluate Various Methods



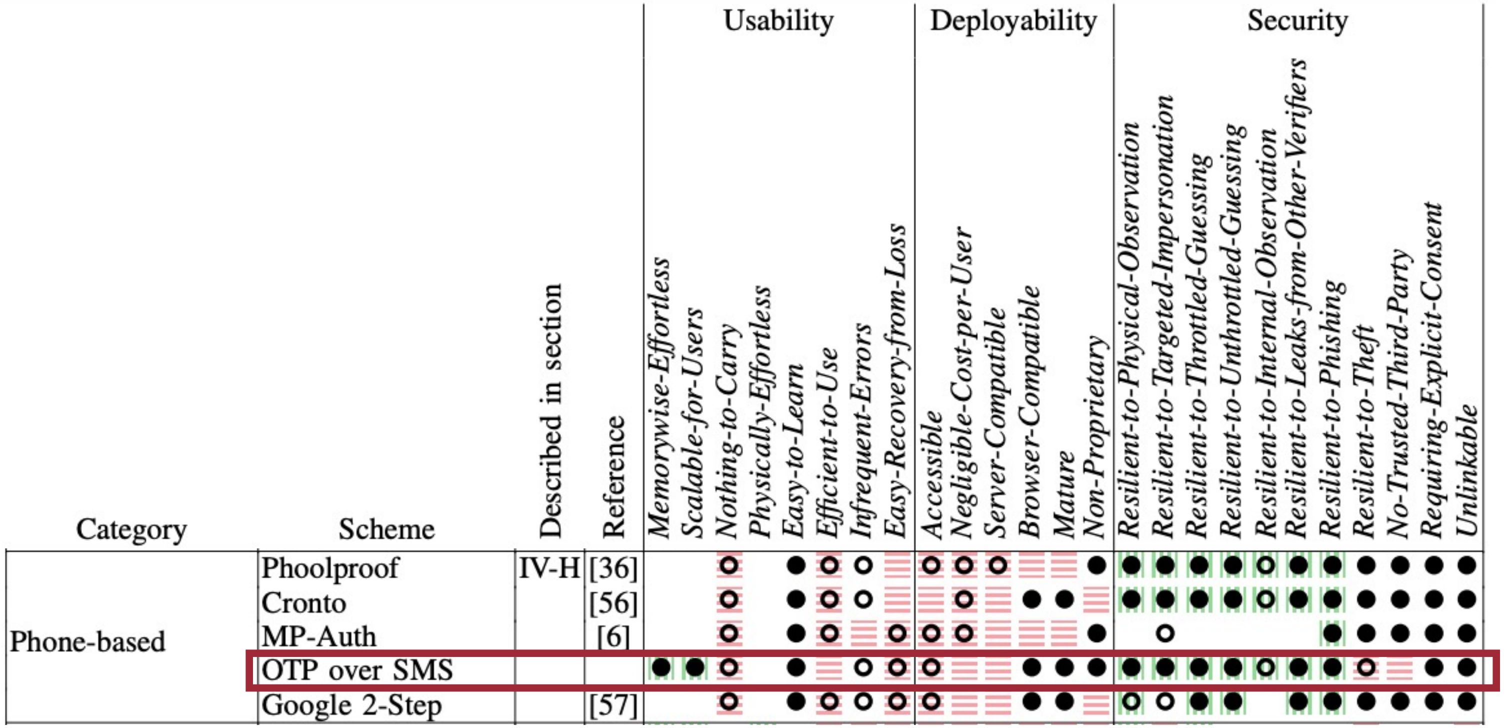
Category	Scheme	Described in section	Reference	Usability								Deployability						Security										
				Memorywise-Effortless	*Scalable-for-Users*	*Nothing-to-Carry*	*Physically-Effortless*	*Easy-to-Learn*	*Efficient-to-Use*	*Infrequent-Errors*	*Easy-Recovery-from-Loss*	*Accessible*	*Negligible-Cost-per-User*	*Server-Compatible*	*Browser-Compatible*	*Mature*	*Non-Proprietary*	*Resilient-to-Physical-Observation*	*Resilient-to-Targeted-Impersonation*	*Resilient-to-Throttled-Guessing*	*Resilient-to-Unthrottled-Guessing*	*Resilient-to-Internal-Observation*	*Resilient-to-Leaks-from-Other-Verifiers*	*Resilient-to-Phishing*	*Resilient-to-Theft*	*No-Trusted-Third-Party*	*Requiring-Explicit-Consent*	*Unlinkable*
(Incumbent)	Web passwords	III	[13]			●		●	●	○	●	●	●	●	●	●	●		○						●	●	●	●
Password managers	Firefox	IV-A	[22]	🟩○	🟩●	🟥○	🟩○	●	●	🟩●	🟥	●	●	●	🟥	●	●	🟩○	○					🟩●	●	●	●	●
	LastPass		[42]	🟩○	🟩●	🟥○	🟩○	●	●	🟩●	🟥○	●	🟥○	●	🟥	●	🟥	🟩○	○	🟩○	🟩○		🟩○	🟩●	●	🟥	●	●
Proxy	URRSA	IV-B	[5]	🟩●		🟥		●	🟥	○	🟥	🟥	●	🟥○	●	🟥	🟥		○			🟩○		🟩●	🟥	🟥	●	●
	Impostor		[23]	🟩○	🟩●	●		●	🟥	🟥	●	●	●	●	🟥○	🟥	●	🟩●	○			🟩○		🟩●	●	🟥	🟥	●
Federated	OpenID	IV-C	[27]	🟩○	🟩●	●	🟩○	🟥○	●	🟩●	●	●	●	🟥	●	●	●	🟩○	○	🟩○	🟩○		🟩●		●	🟥	●	🟥
	Microsoft Passport		[43]	🟩○	🟩●	●	🟩○	●	●	🟩●	●	●	🟥	🟥	●	●	🟥	🟩○	○	🟩○	🟩○		🟩●		●	🟥	●	🟥
	Facebook Connect		[44]	🟩○	🟩●	●	🟩○	●	●	🟩●	●	●	●	🟥	●	●	🟥	🟩○	○	🟩○	🟩○		🟩●		●	🟥	🟥	🟥
	BrowserID		[45]	🟩○	🟩●	●	🟩○	●	●	🟩●	●	●	●	🟥	🟥○	🟥○	●	🟩○	○	🟩○	🟩○		🟩●		●	🟥	●	🟥
	OTP over email		[46]	🟩○	🟩●	●		●	🟥	🟩●	●	●	●	🟥	●	🟥	●	🟩○	○	🟩○	🟩○		🟩●	🟩●	●	🟥	●	🟥
Graphical	PCCP	IV-D	[7]			●		●	🟥○	○	●	🟥	●	🟥	●	🟥	●		🟩●	🟩○			🟩●	🟩●	●	●	●	●
	PassGo		[47]			●		●	🟥○	○	●	🟥	●	🟥	●	🟥○	●		🟩●						●	●	●	●
Cognitive	GrIDsure (original)	IV-E	[30]			●		●	🟥○	○	●	🟥	●	🟥	●	🟥	🟥		🟩●						●	●	●	●
	Weinshall		[48]			●		🟥	🟥	🟥	🟥	🟥	●	🟥	●	🟥	●	🟩○	🟩●				🟩●	🟩●	●	●	●	●
	Hopper Blum		[49]			●		🟥	🟥	🟥	🟥	●	●	🟥	●	🟥	●	🟩○	🟩●				🟩●	🟩●	●	●	●	●
	Word Association		[50]			●		●	●	○	🟥○	●	●	🟥	●	🟥	●		🟥						●	●	●	●
Paper tokens	OTPW	IV-F	[33]			🟥		●	🟥	🟥	●	🟥	●	🟥	●	●	●		🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	●	●	●	●
	S/KEY		[32]	🟩●		🟥		●	🟥	○	●	🟥	●	🟥	●	●	●		🟩●	🟩●	🟩●	🟩●	🟩●	🟩○	🟥	●	●	●
	PIN+TAN		[51]			🟥		●	🟥	○	🟥○	🟥	🟥○	🟥	●	●	●		🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	🟥○	●	●	●
Visual crypto	PassWindow		[52]	🟩●		🟥		🟥	🟥	🟥	🟥	🟥	🟥○	🟥	●	●	🟥	🟩○	🟩●	🟩●	🟩●	🟩○	🟩●	🟩●	🟥	●	●	●
Hardware tokens	RSA SecurID	IV-G	[34]			🟥		●	🟥○	○	🟥	🟥	🟥	🟥	●	●	🟥	🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	●	🟥	●	●
	YubiKey		[53]			🟥		●	🟥○	○	🟥	●	🟥	🟥	●	●	🟥	🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	●	🟥	●	●
	IronKey		[54]	🟩○	🟩●	🟥	🟩○	🟥○	🟥○	○	🟥	●	🟥	●	●	●	🟥	🟩●	○			🟩○		🟩●	●	●	●	●
	CAP reader		[55]			🟥		●	🟥○	○	🟥	🟥	🟥	🟥	●	●	🟥	🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	●	●	●	●
	Pico		[8]	🟩●	🟩●	🟥	🟩●	🟥	🟥○	○	🟥	🟥	🟥	🟥	🟥	🟥	●	🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	🟩●	🟥○	●	●	●
Phone-based	Phoolproof	IV-H	[36]			🟥○		●	🟥○	○	🟥	🟥○	🟥○	🟥○	🟥	🟥	●	🟩●	🟩●	🟩●	🟩●	🟩○	🟩●	🟩●	●	●	●	●
	Cronto		[56]			🟥○		●	🟥○	○	🟥	🟥	🟥○	🟥	●	●	🟥	🟩●	🟩●	🟩●	🟩●	🟩○	🟩●	🟩●	●	●	●	●
	MP-Auth		[6]			🟥○		●	🟥○	🟥	🟥○	🟥○	🟥○	🟥	🟥	🟥	●		○					🟩●	●	●	●	●
	OTP over SMS			🟩●	🟩●	🟥○		●	🟥	○	🟥○	🟥○	🟥	🟥	●	●	●	🟩●	🟩●	🟩●	🟩●	🟩○	🟩●	🟩●	🟥○	🟥	●	●
	Google 2-Step		[57]			🟥○		●	🟥○	○	🟥○	🟥○	🟥	🟥	●	●	🟥	🟩○	○	🟩●	🟩●		🟩●	🟩●	●	●	●	●
Biometric	Fingerprint	IV-I	[38]	🟩●	🟩●	●	🟩○	●	🟥○	🟥	🟥	🟥○	🟥	🟥	🟥	🟥○	🟥	🟩●	🟥	🟩●					🟥	●	●	🟥
	Iris		[39]	🟩●	🟩●	●	🟩○	●	🟥○	🟥	🟥	🟥○	🟥	🟥	🟥	🟥○	🟥	🟩●	🟥	🟩●					🟥	●	🟥○	🟥
	Voice		[40]	🟩●	🟩●	●	🟩○	●	🟥○	🟥	🟥	🟥○	🟥○	🟥	🟥○	🟥○	🟥	🟩●	🟥	🟩○					🟥	●	●	🟥
Recovery	Personal knowledge		[58]	🟩○		●		●	●	○	●	●	●	🟥	●	●	●		🟥						●	●	●	●
	Preference-based		[59]	🟩○		●		●	🟥○	🟩●	🟥○	●	●	🟥	●	🟥	🟥		○					🟩●	●	●	●	●
	Social re-auth.		[60]			●		●	🟥	🟩●	🟥	●	●	🟥	●	🟥○	🟥	🟩○	🟥	🟩●	🟩●	🟩○	🟩○	🟩●	●	🟥	●	🟥○

- **Green**: better than passwords
- **Red**: worse than passwords
- **No background**: no benefit

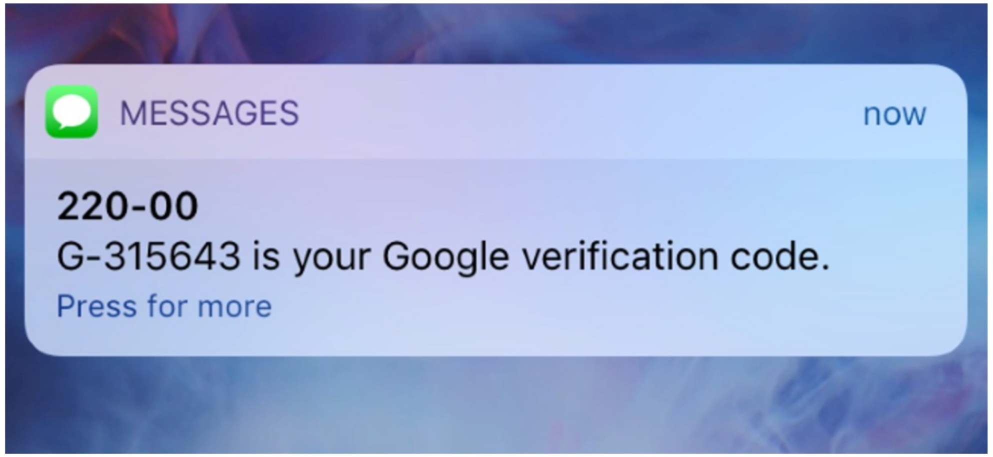
- **Solid Circle (●)**: offers the benefit
- **Empty Circle (○)**: almost offers the benefit
- **No Circle**: no benefit

No known scheme provides the full set of benefits that passwords already provide

How About OTP over SMS?



Category	Scheme	Described in section	Reference	Usability								Deployability						Security										
				Memorywise-Effortless	*Scalable-for-Users*	*Nothing-to-Carry*	*Physically-Effortless*	*Easy-to-Learn*	*Efficient-to-Use*	*Infrequent-Errors*	*Easy-Recovery-from-Loss*	*Accessible*	*Negligible-Cost-per-User*	*Server-Compatible*	*Browser-Compatible*	*Mature*	*Non-Proprietary*	*Resilient-to-Physical-Observation*	*Resilient-to-Targeted-Impersonation*	*Resilient-to-Throttled-Guessing*	*Resilient-to-Unthrottled-Guessing*	*Resilient-to-Internal-Observation*	*Resilient-to-Leaks-from-Other-Verifiers*	*Resilient-to-Phishing*	*Resilient-to-Theft*	*No-Trusted-Third-Party*	*Requiring-Explicit-Consent*	*Unlinkable*
Phone-based	Phoolproof	IV-H	[36]			○		●	○	○		○	○	○			●	●	●	●	●	○	●	●	●	●	●	●
	Cronto		[56]			○		●	○	○			○		●	●		●	●	●	●	○	●	●	●	●	●	●
	MP-Auth		[6]			○		●	○		○	○	○				●		○					●	●	●	●	●
	OTP over SMS			●	●	○		●		○	○	○			●	●	●	●	●	●	●	○	●	●	○		●	●
	Google 2-Step		[57]			○		●	○	○	○	○			●	●		○	○	●	●		●	●	●	●	●	●



MESSAGES now

220-00

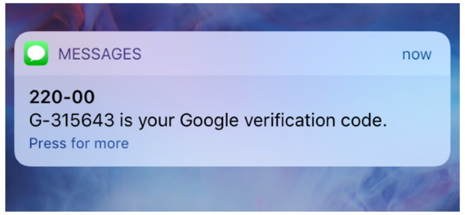
G-315643 is your Google verification code.

Press for more

- Phone-based methods provide better security
- Its usability and deployablity is worse than password
  - [D1] Accessible
  - [D2] Negligible-Cost-per-User
  - [U3] Nothing-to-Carry
  - [U6] Efficient-to-User
  - [U8] Easy-Recovery-from-Loss

Multi-factor Authentication (MFA)

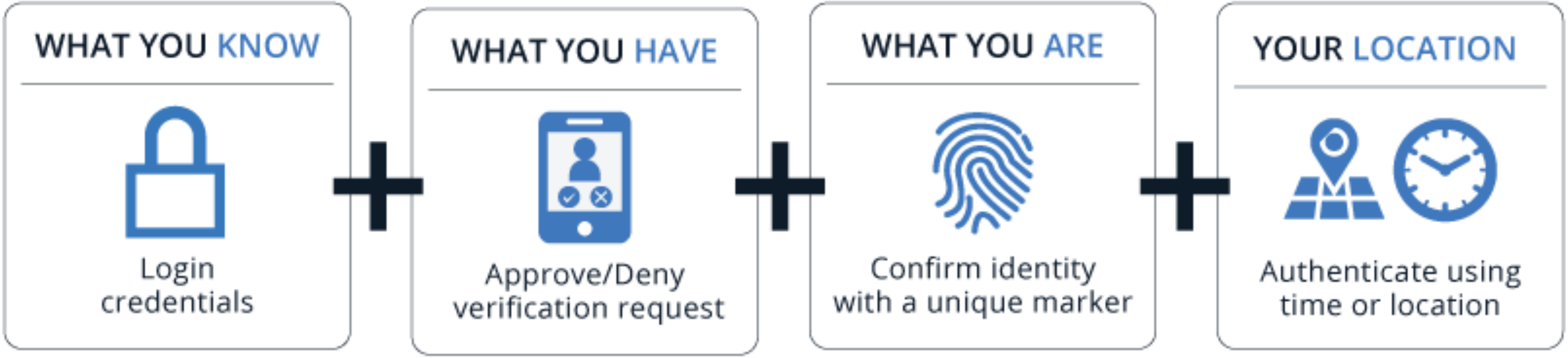
- A combination of criteria that need to be met
  - To strengthen the overall security of a system



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

+

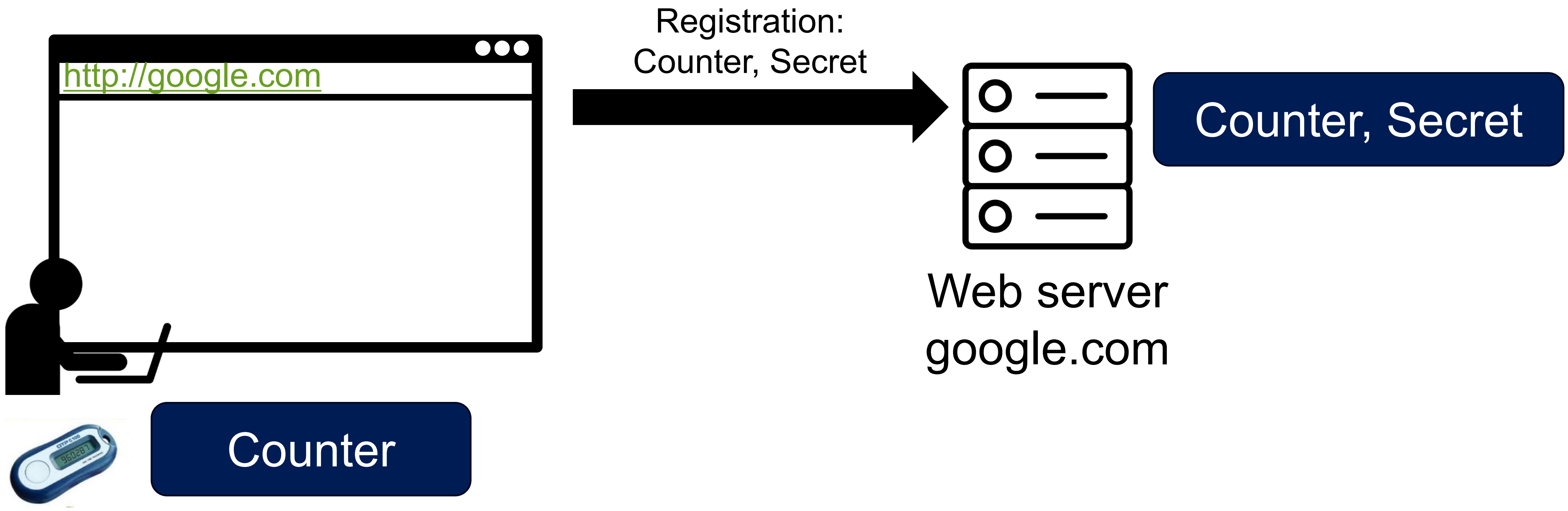
WHAT YOU ARE
Confirm identity with a unique marker

+

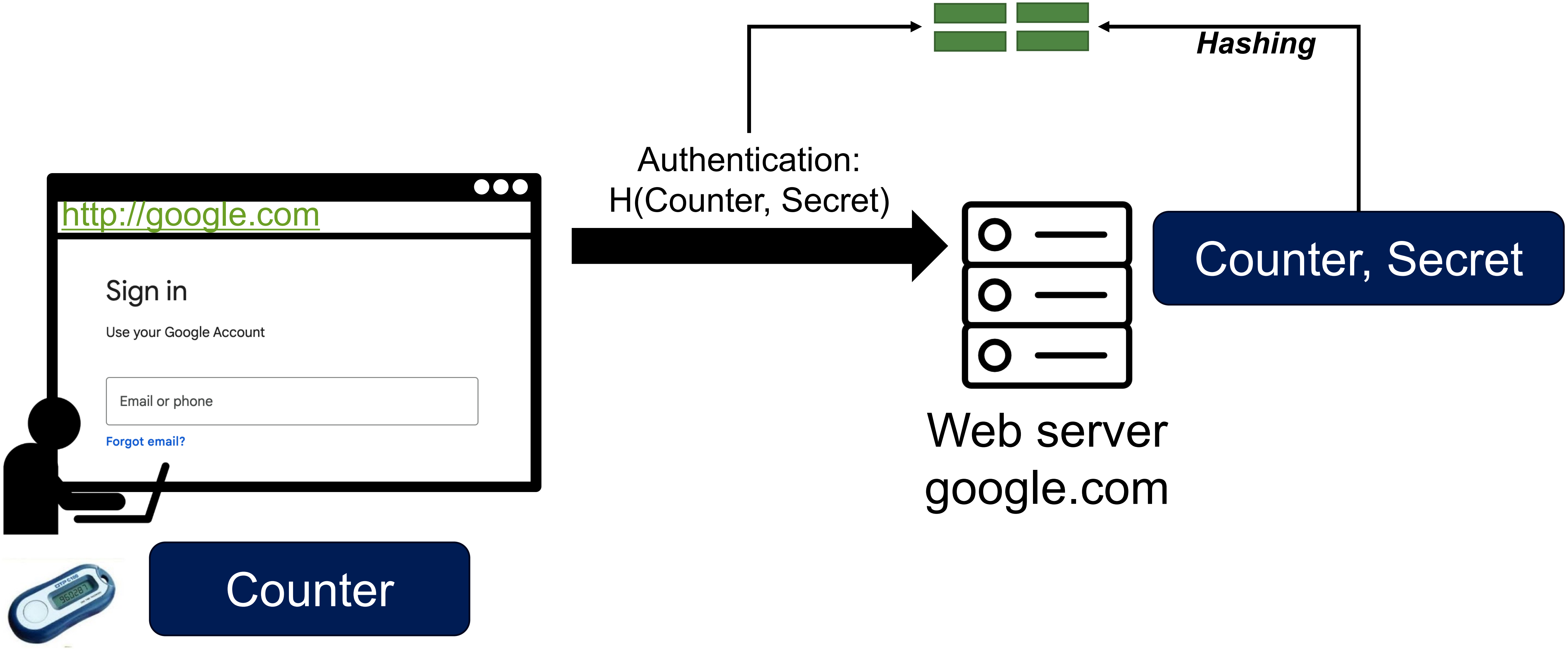
YOUR LOCATION
Authenticate using time or location

Example: HMAC-based One-Time Password

- HMAC-based One-Time Password (HOTP) = HMAC(Secret, Counter)
  - E.g., Google Authenticator



Authentication (1st Try)



Hashing

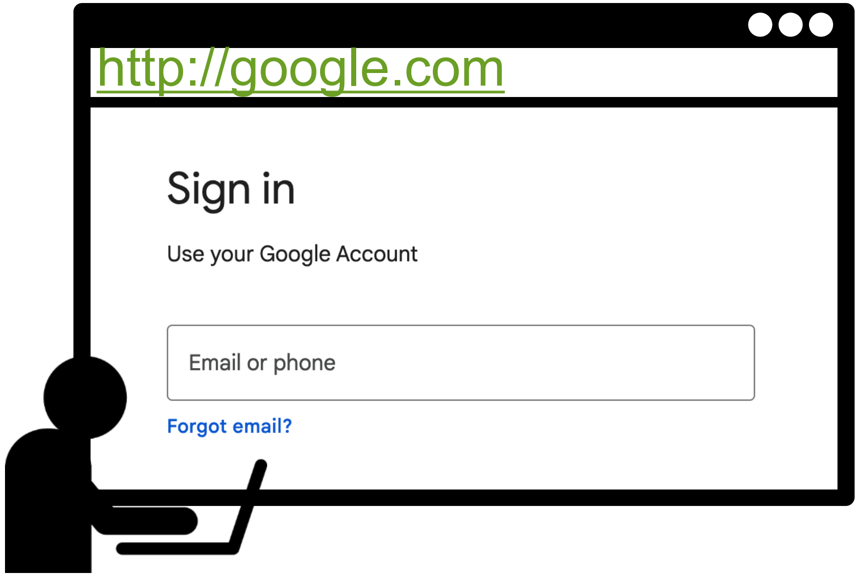
Authentication:
H(Counter, Secret)

Counter, Secret

Web server
google.com

Counter

Authentication (After the 1st Try)



http://google.com

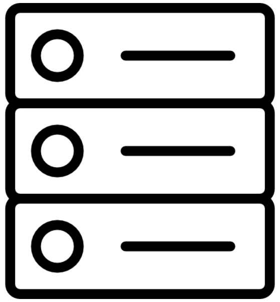
Sign in

Use your Google Account

Email or phone

Forgot email?

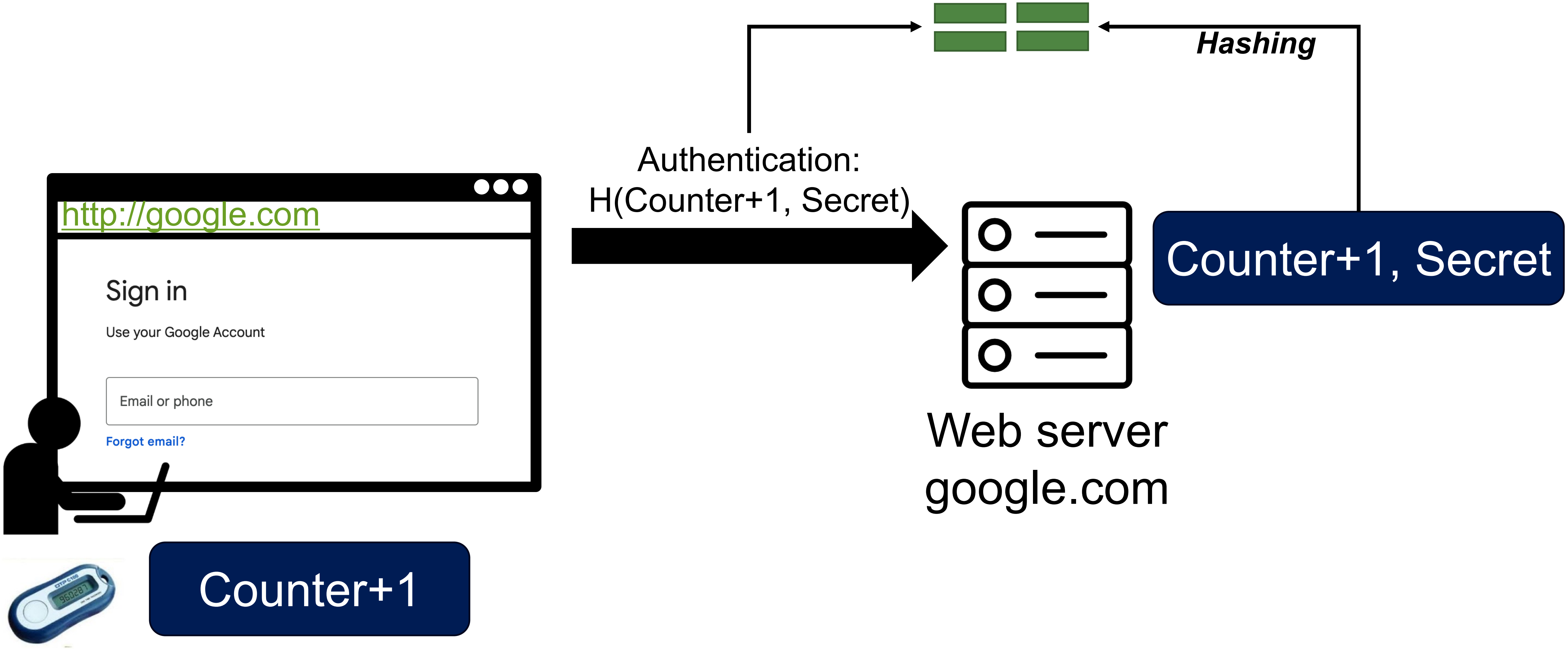
Counter+1



Counter+1, Secret

Web server
google.com

Authentication (2nd Try)



Hashing

Authentication:
H(Counter+1, Secret)

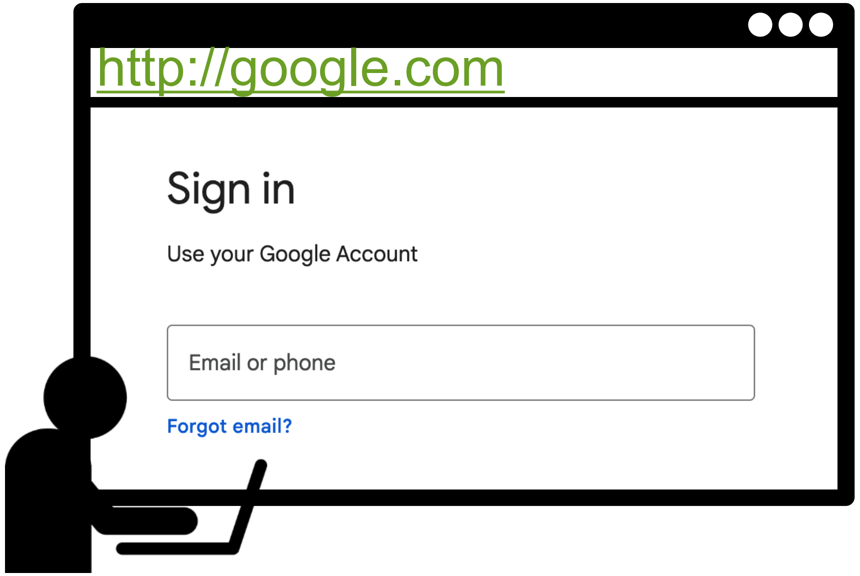
Counter+1, Secret

Web server
google.com

Counter+1

Out-of-Sync Between Client and Server

- What if your child starts pushing your OTP button?



http://google.com

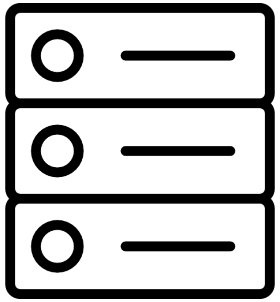
Sign in

Use your Google Account

Email or phone

Forgot email?

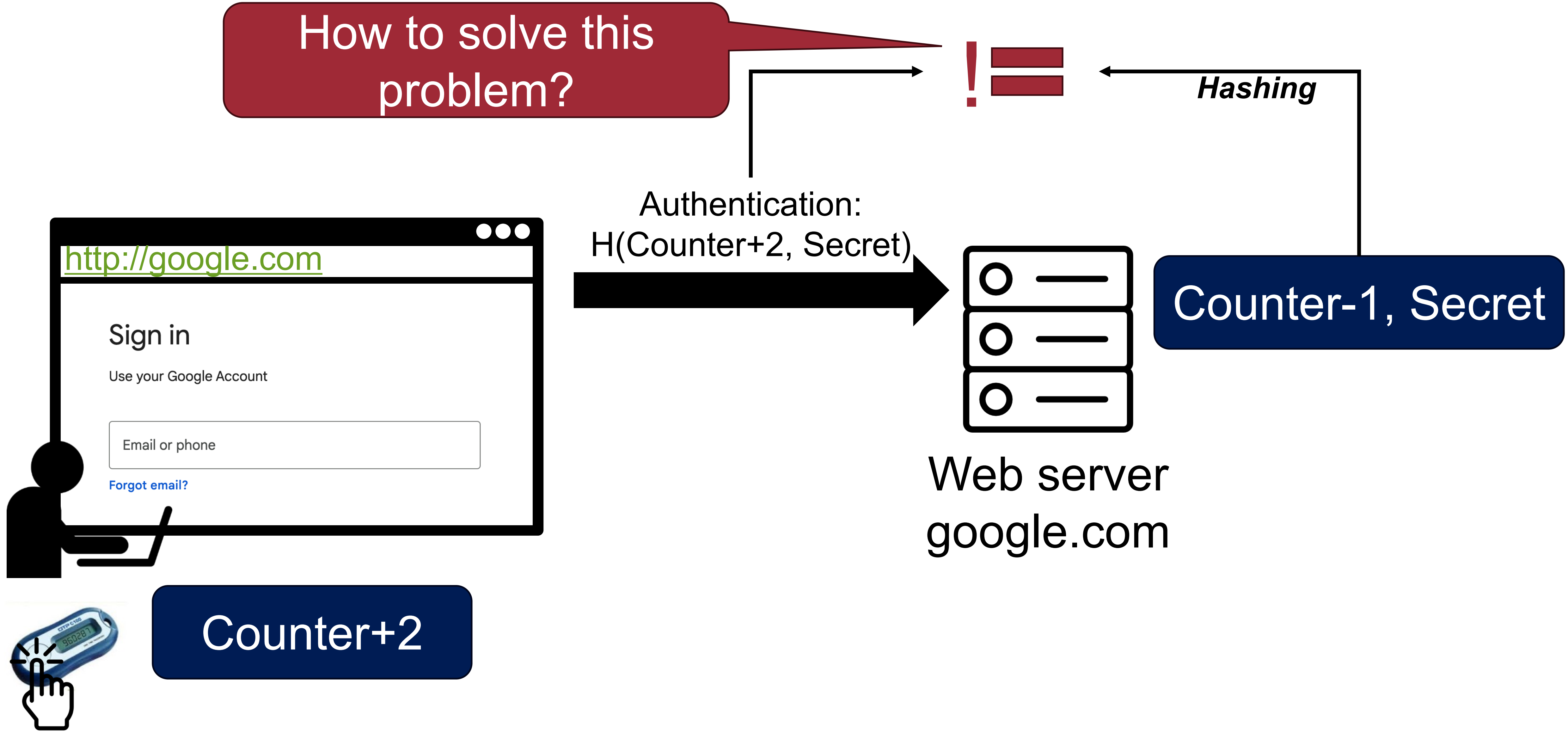
Counter+2



Counter-1, Secret

Web server
google.com

Out-of-Sync Between Client and Server



How to solve this problem?

!=

Hashing

Authentication:
H(Counter+2, Secret)

http://google.com

Sign in

Use your Google Account

Email or phone

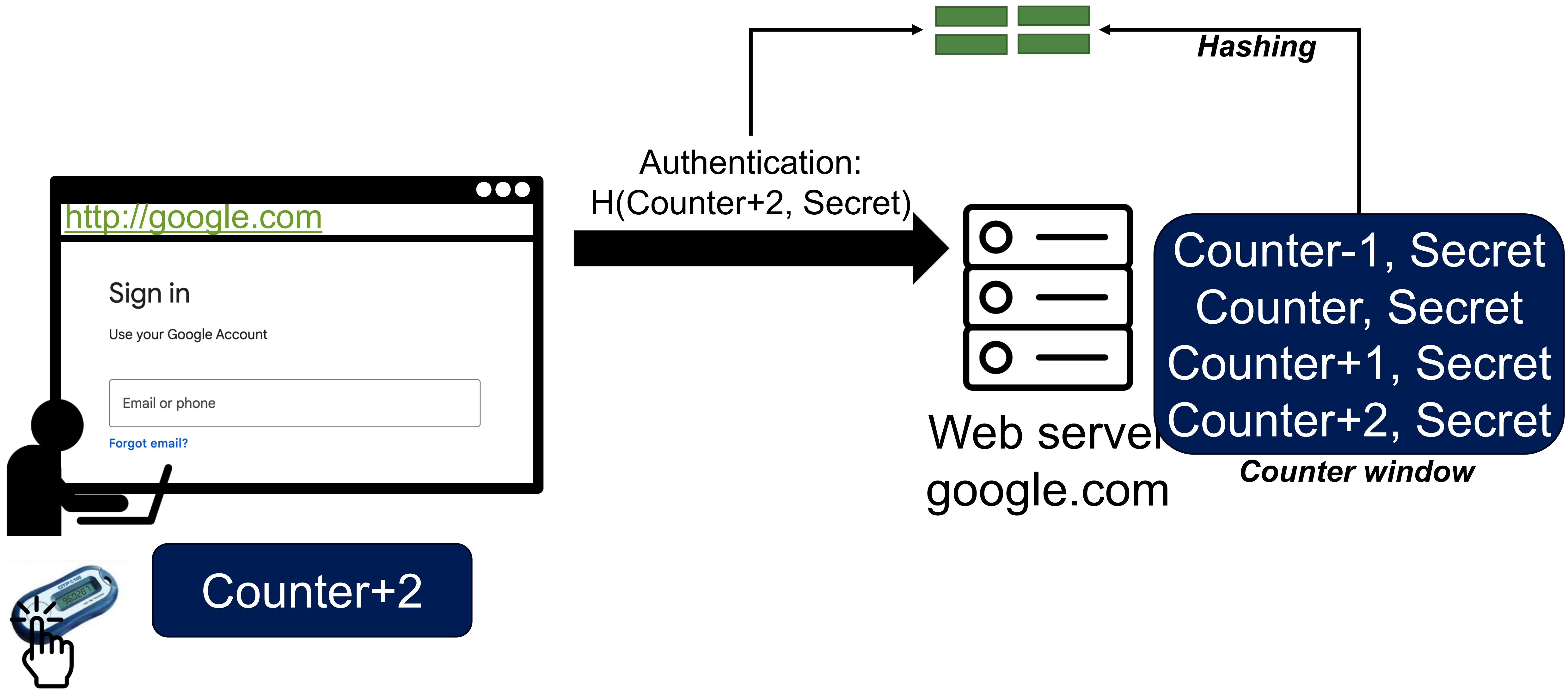
Forgot email?

Web server
google.com

Counter-1, Secret

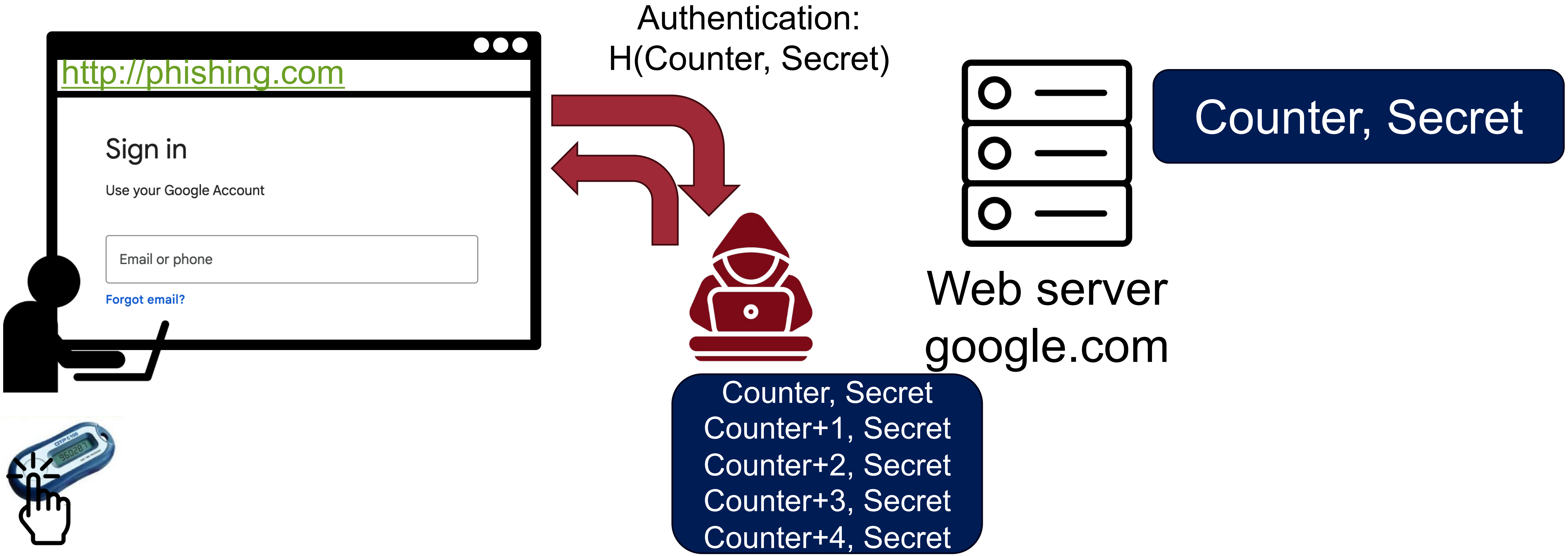
Counter+2

Out-of-Sync Between Client and Server



Is HOTP Secure against Phishing?

Is HOTP Secure against Phishing?



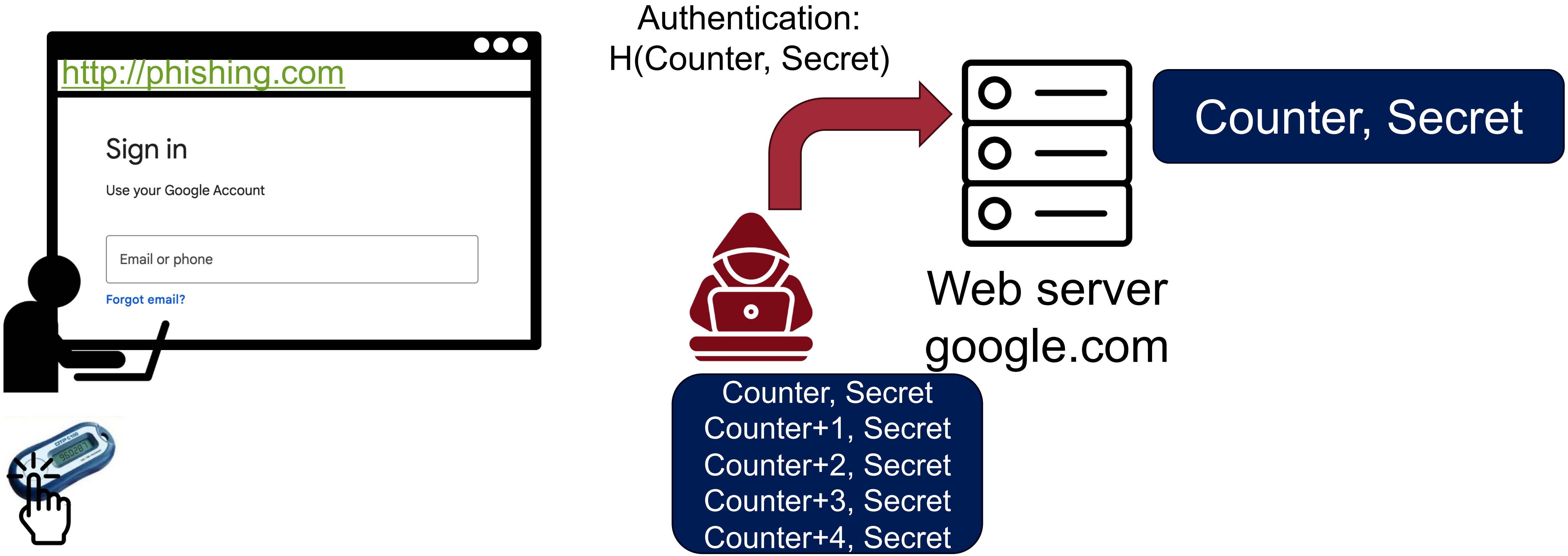
Authentication:
H(Counter, Secret)

Counter, Secret

Web server
google.com

Counter, Secret
Counter+1, Secret
Counter+2, Secret
Counter+3, Secret
Counter+4, Secret

Is HOTP Secure against Phishing?



http://phishing.com

Sign in

Use your Google Account

Email or phone

Forgot email?

Authentication:
H(Counter, Secret)

Counter, Secret

Web server
google.com

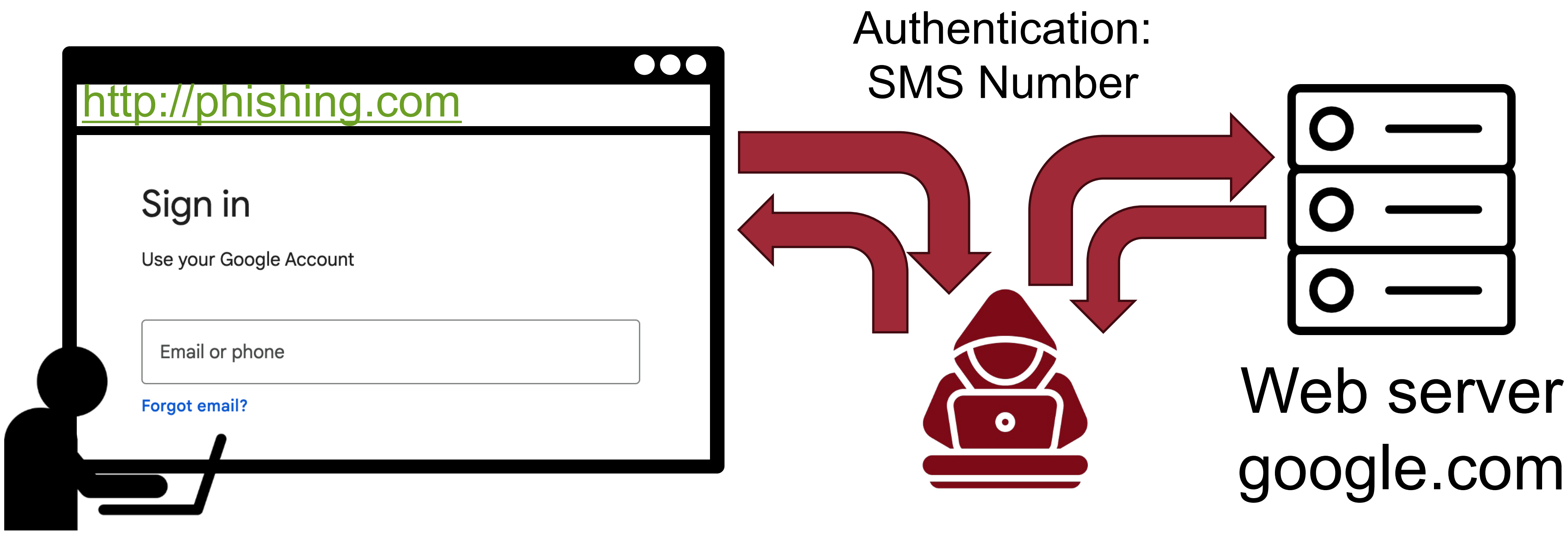
Counter, Secret
Counter+1, Secret
Counter+2, Secret
Counter+3, Secret
Counter+4, Secret

How about SMS-based OTP?

- Does this phishing works for SMS (Time)-based OTP?
  - No! but how about real-time phishing?

How about SMS-based OTP?

- Does this phishing works for SMS (Time)-based OTP?
  - No! but how about real-time phishing?



http://phishing.com

Sign in

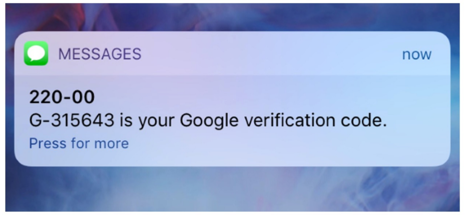
Use your Google Account

Email or phone

Forgot email?

Authentication:
SMS Number

Web server
google.com



MESSAGES now

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

Summary

- Password is an insecure authentication method for large audiences
- So far, no authentication method provides the full set of benefits that legacy passwords already provide
- 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

Question?