

# CSCI 1515 Applied Cryptography

## This Lecture:

- Public Key Infrastructure (continued)
- Password-Based Authentication
- Two-Factor Authentication (2FA)

# One-Sided Secure Authentication

public Where does it come from?  
 $(VK_s, SK_s) \leftarrow \text{Gen}(1^\lambda)$

Client



$\Downarrow$   
 $g^a$

$\text{Vrfy}_{VK_s}(g^b, \sigma_s) \stackrel{?}{=} 1$

$\xrightarrow{g^a}$   
Diffie-Hellman Key Exchange

$\xleftarrow{g^b, \sigma_s \leftarrow \text{Sign}_{SK_s}(g^b)}$

Server



$\Downarrow$   
 $g^b$

Can we prevent Man-in-the-Middle Attack?

# Public Key Infrastructure (PKI)

public  
 $(VK, SK) \leftarrow \text{Gen}(1^\lambda)$

Certificate Authority  
(CA)

Alice



Certificate signing request (CSR)

"alice.com";  $VK_A$



"alice.com"

$$(VK_A, SK_A) \leftarrow \text{Gen}(1^\lambda)$$

"Certificate"

$$\text{cert}_A \leftarrow \text{Sign}_{SK}(\text{alice.com} \parallel VK_A)$$

$\text{cert}_A$



Standard: X.509 certificate

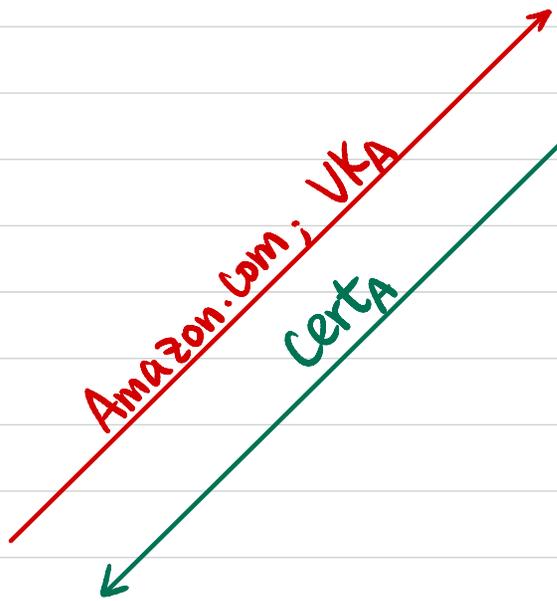
# Public Key Infrastructure (PKI)

Certificate Authority  
(CA)

public  
 $(VK, SK) \leftarrow \text{Gen}(1^\lambda)$

$\text{Cert}_A \leftarrow \text{Sign}_{SK}(\text{Amazon.com} \parallel VK_A)$

$\text{Cert}_G \leftarrow \text{Sign}_{SK}(\text{Google.com} \parallel VK_G)$



Amazon



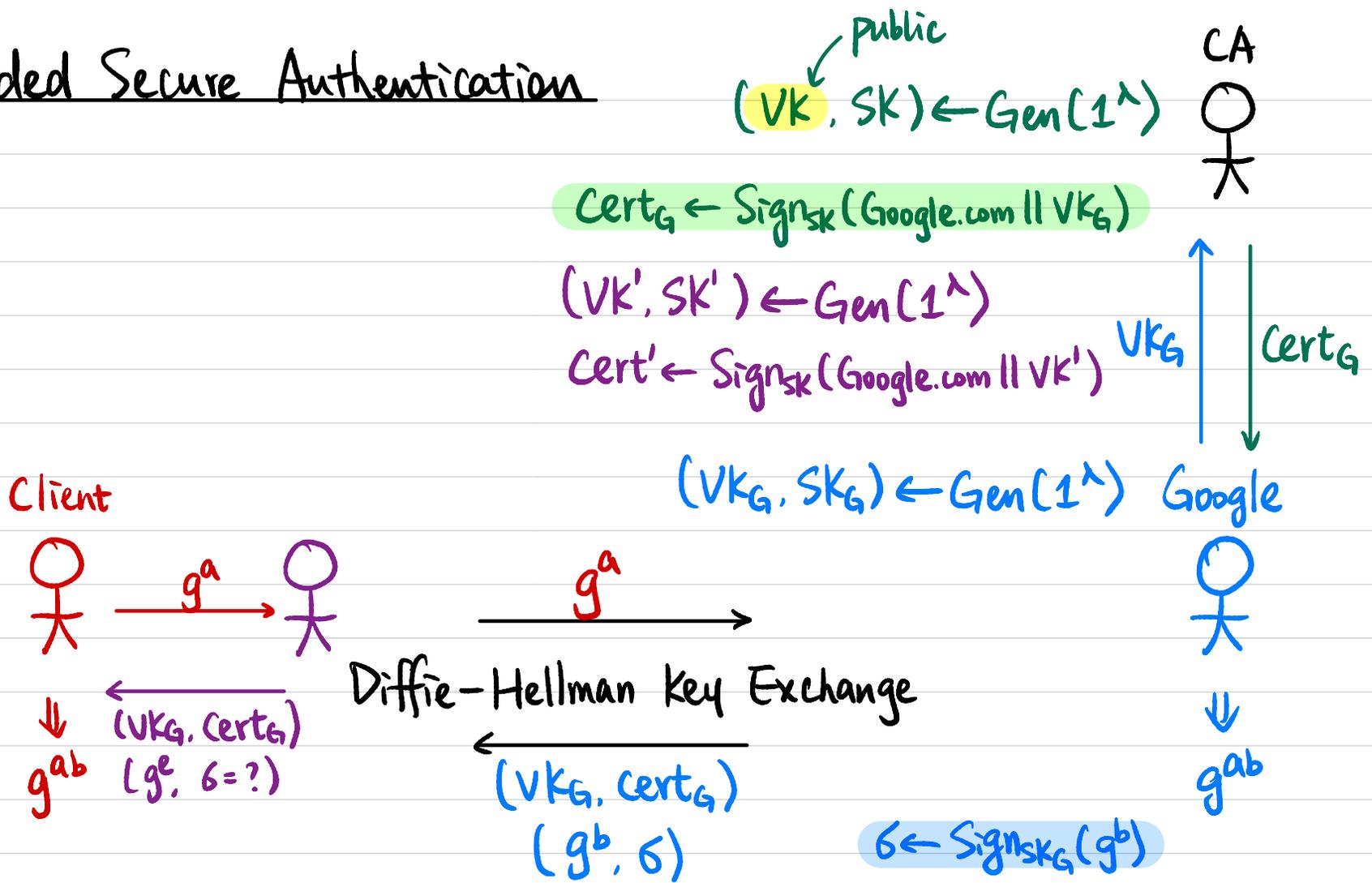
Google



$(VK_A, SK_A) \leftarrow \text{Gen}(1^\lambda)$

$(VK_G, SK_G) \leftarrow \text{Gen}(1^\lambda)$

# One-Sided Secure Authentication



$\text{Vrfy}_{VK}(\text{Google.com} \parallel VK_G, \text{Cert}_G) \stackrel{?}{=} 1$

$\text{Vrfy}_{VK_G}(g^b, \sigma) \stackrel{?}{=} 1$

**Subject Name** \_\_\_\_\_  
**Country** US  
**State/Province** CA  
**Locality** Menlo Park  
**Organization** Facebook, Inc.  
**Common Name** \*.facebook.com

**Issuer Name** \_\_\_\_\_  
**Country** US  
**Organization** DigiCert Inc  
**Organizational Unit** www.digicert.com  
**Common Name** DigiCert SHA2 High Assurance Server CA

**Serial Number** 0E CB 09 39 B2 B1 01 54 B8 95 70 C7 B2 2B 7A 47  
**Version** 3

**Signature Algorithm** SHA-256 with RSA Encryption  
( 1.2.840.113549.1.1.11 )

**Not Valid Before** Wednesday, August 27, 2014 at 5:00:00 PM Pacific Daylight Time

**Not Valid After** Friday, December 30, 2016 at 4:00:00 AM Pacific Standard Time

**Public Key Info** \_\_\_\_\_

**Algorithm** Elliptic Curve Public Key ( 1.2.840.10045.2.1 )

**Parameters** Elliptic Curve secp256r1 ( 1.2.840.10045.3.1.7 )

**Public Key** 65 bytes : 04 D8 D1 DD 35 BD E2 59 B6 FB 9B 1F 54  
15 8C DB BF 4E 58 BD 47 BE B8 10 FC 22 E9 D2 9E  
98 F8 49 2A 25 FB 94 46 E4 42 99 84 50 1C 5F 01  
FD 14 25 31 5C 4E D9 64 FD C5 0C B3 46 D2 A1 BC  
70 B4 87 8E

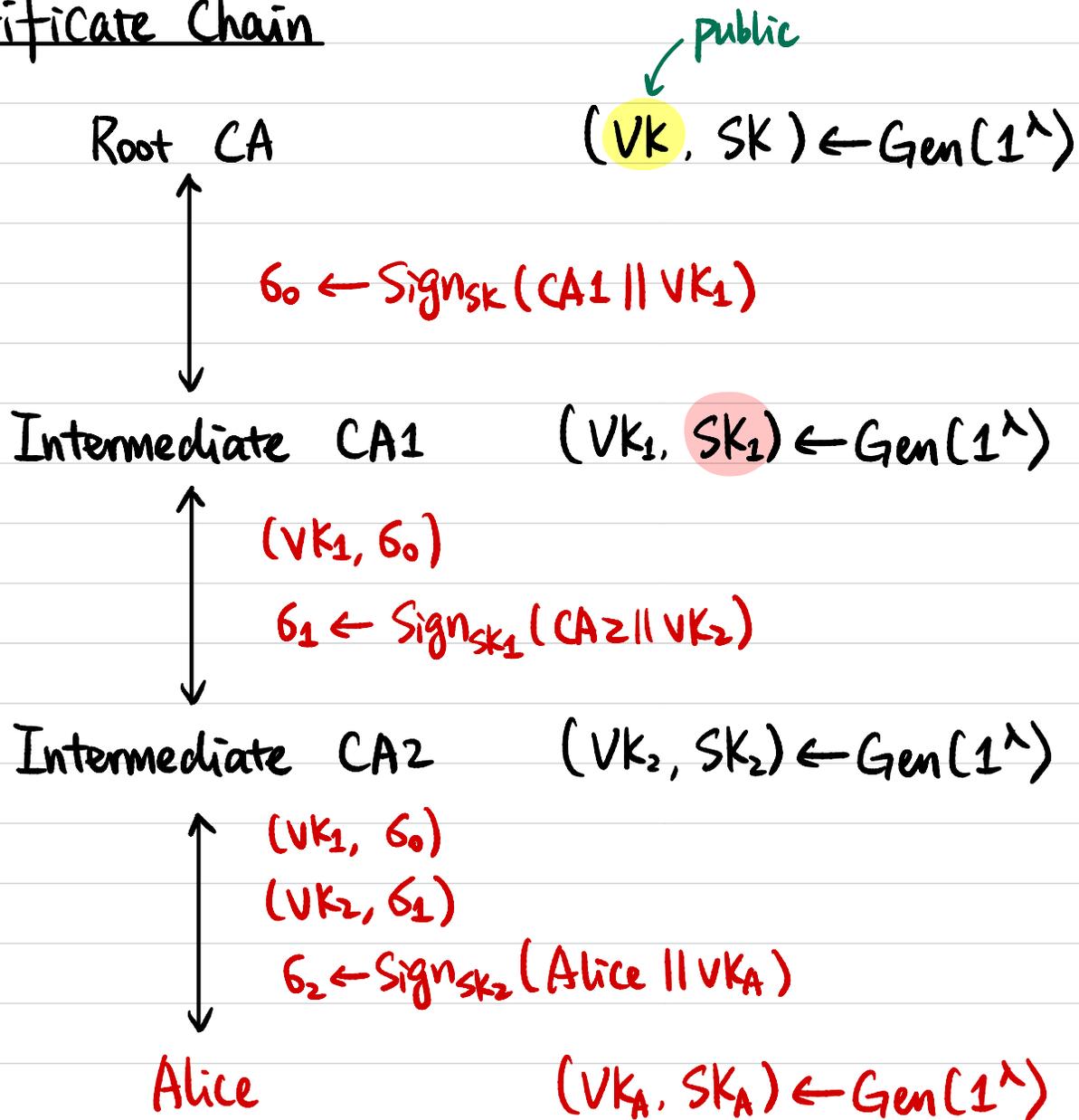
**Key Size** 256 bits

**Key Usage** Encrypt, Verify, Derive

**Signature** 256 bytes : AA 91 AE 52 01 8C 60 F6 02 B6 94 EB  
AF 6E EB DD 3C C8 E1 6F 17 AB B8 28 80 EC DC 54  
82 56 24 C1 16 08 E1 C2 C8 3E 3C 0F 53 18 40 7F  
DF 41 36 93 95 5F B1 D9 35 43 5E 94 60 F9 D6 A7...

**Figure 13.4:** An example X.509 certificate

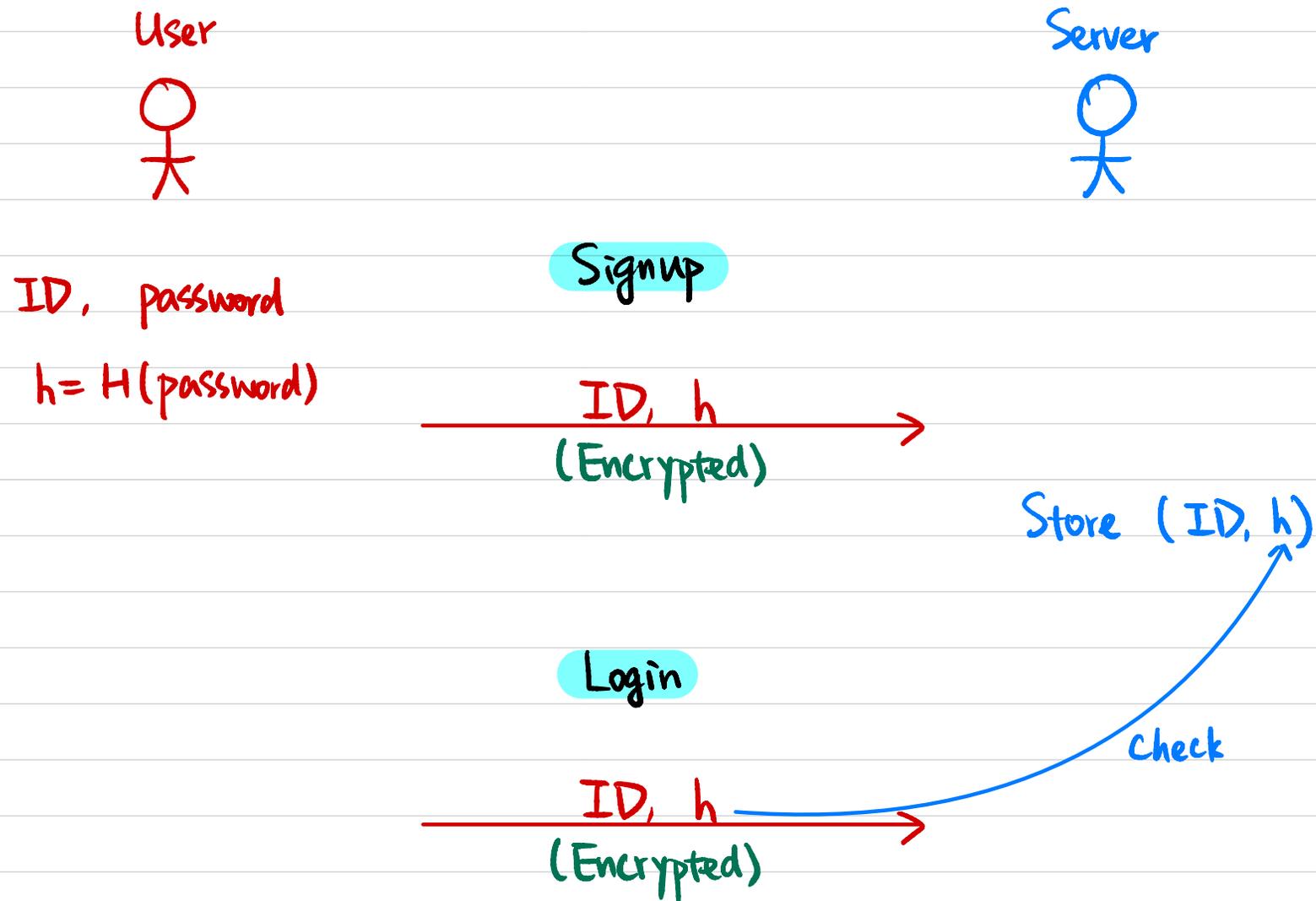
# Certificate Chain



## Certificate Revocation

- Short-lived certificates
- Certificate revocation lists (CRLs)

# Password-Based Authentication



Password Security ?

# Online Dictionary Attack

User



Server



ID, password

$h = H(\text{password})$

Signup

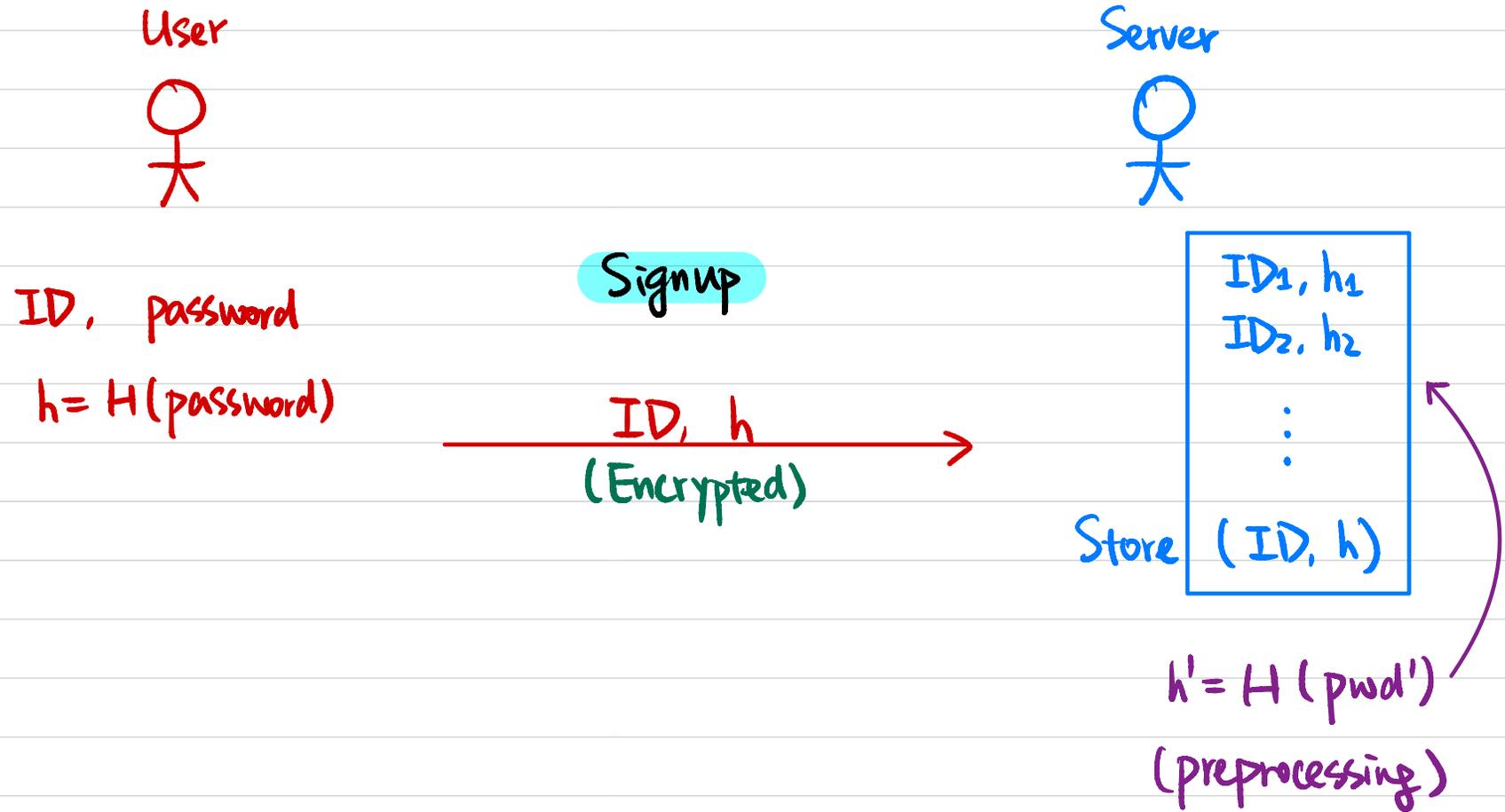
$\xrightarrow{\text{ID, } h \text{ (Encrypted)}}$

Store (ID, h)

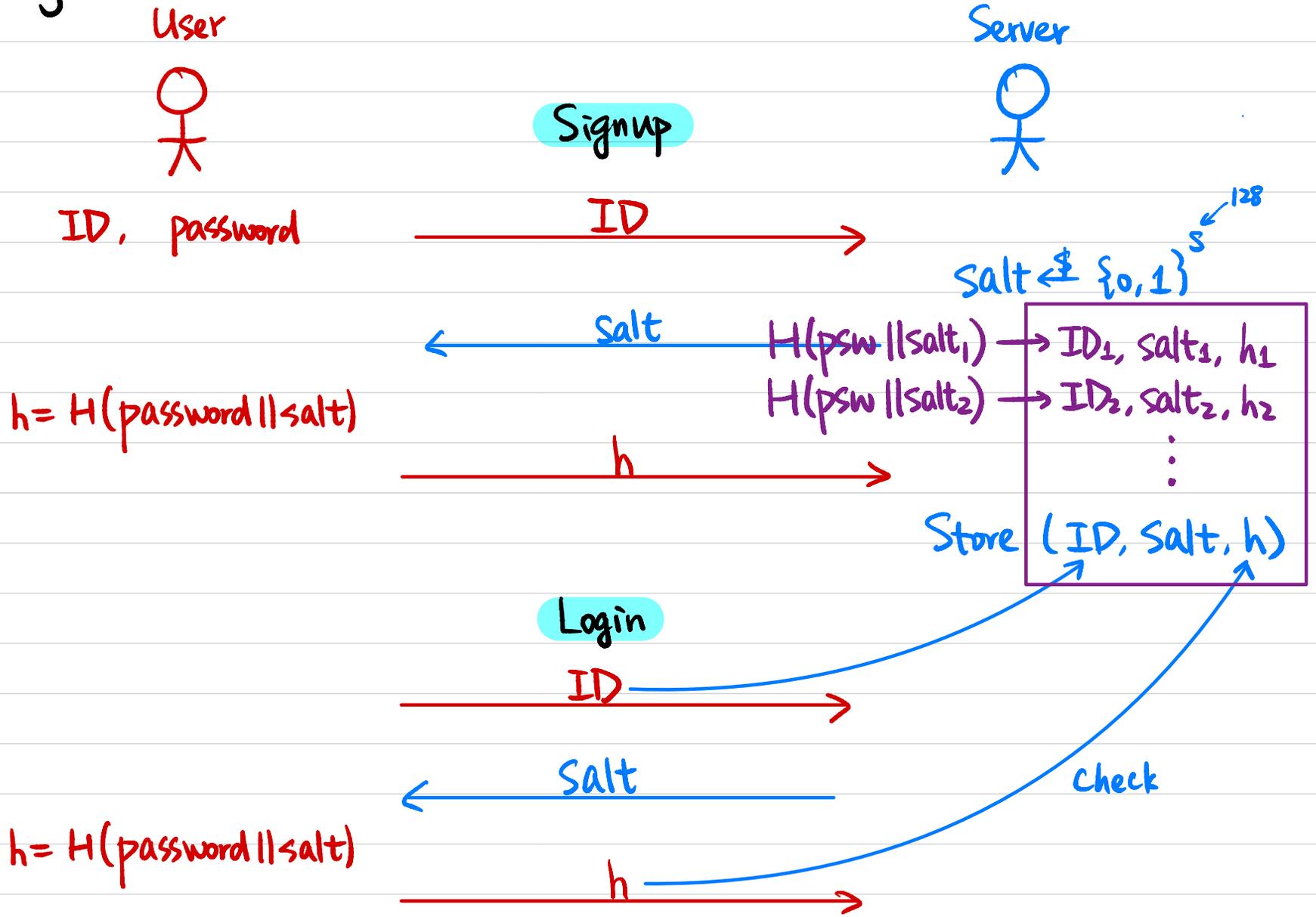
Login

$\xrightarrow{\text{ID, } h' = H(\text{pwd}') \text{ (Encrypted)}}$

# Offline Dictionary Attack

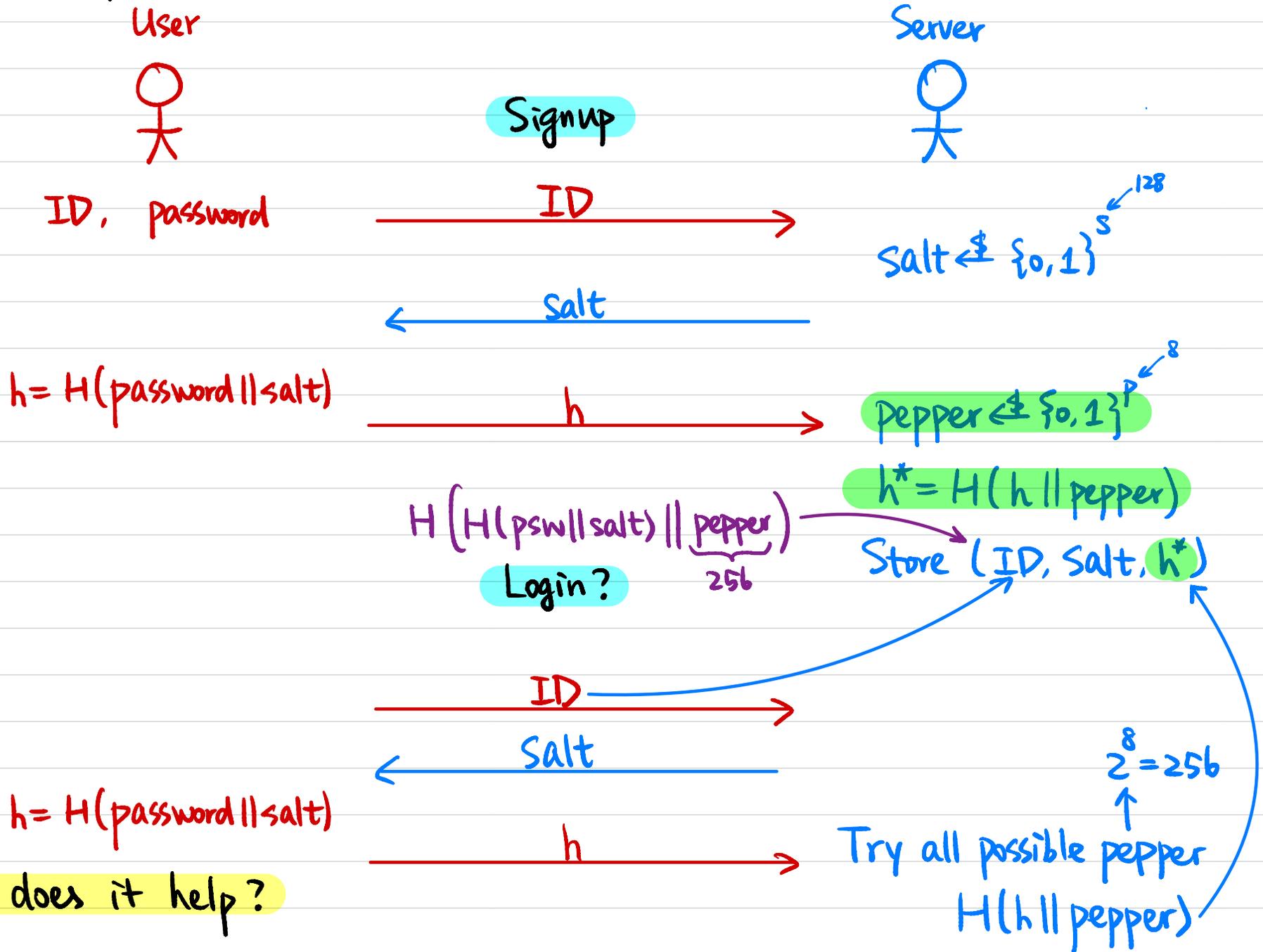


# Salting



Why does it help?

# Salt & Pepper



## Slow Hash Functions

- Computation-heavy hash function

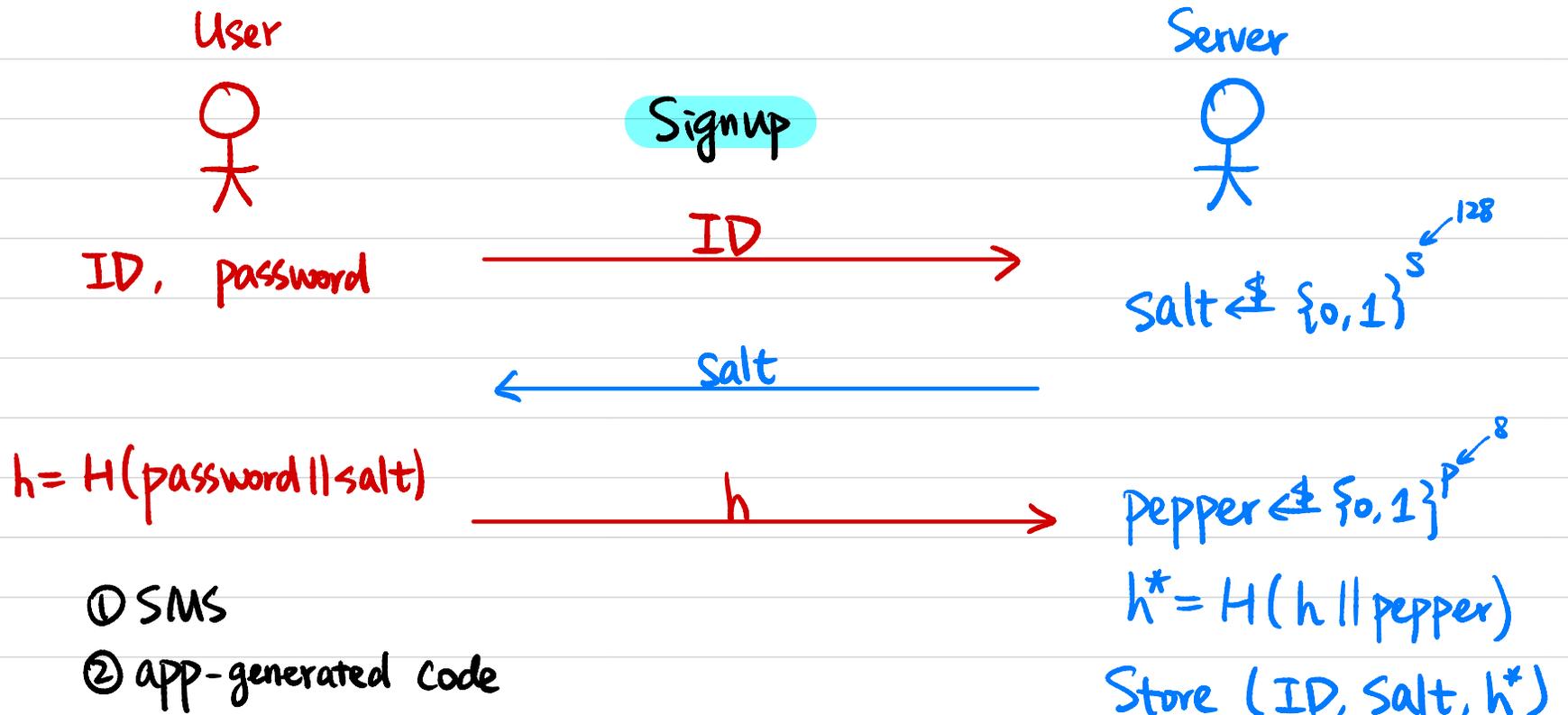
↳ Compose SHA256 in a certain way.

Application-Specific Integrated Circuit (ASIC) → blockchain mining

- Memory-hard hash functions

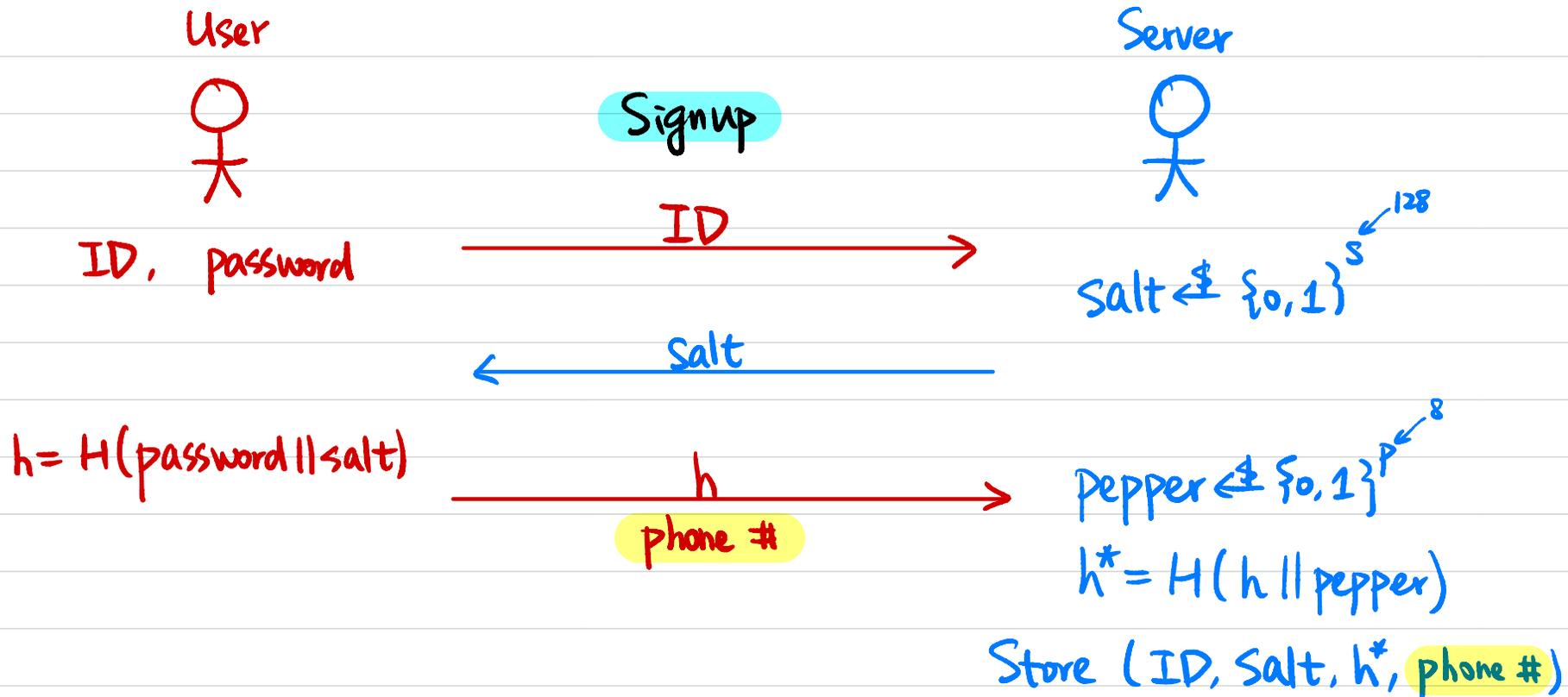
↳ Scrypt

# Two-Factor Authentication (2FA)



How would you design it?

# Two-Factor Authentication (2FA) ① SMS



# Two-Factor Authentication (2FA) ② app-generated code

