Lec 16: User Authentication (2)

CSED415: Computer Security

Spring 2025

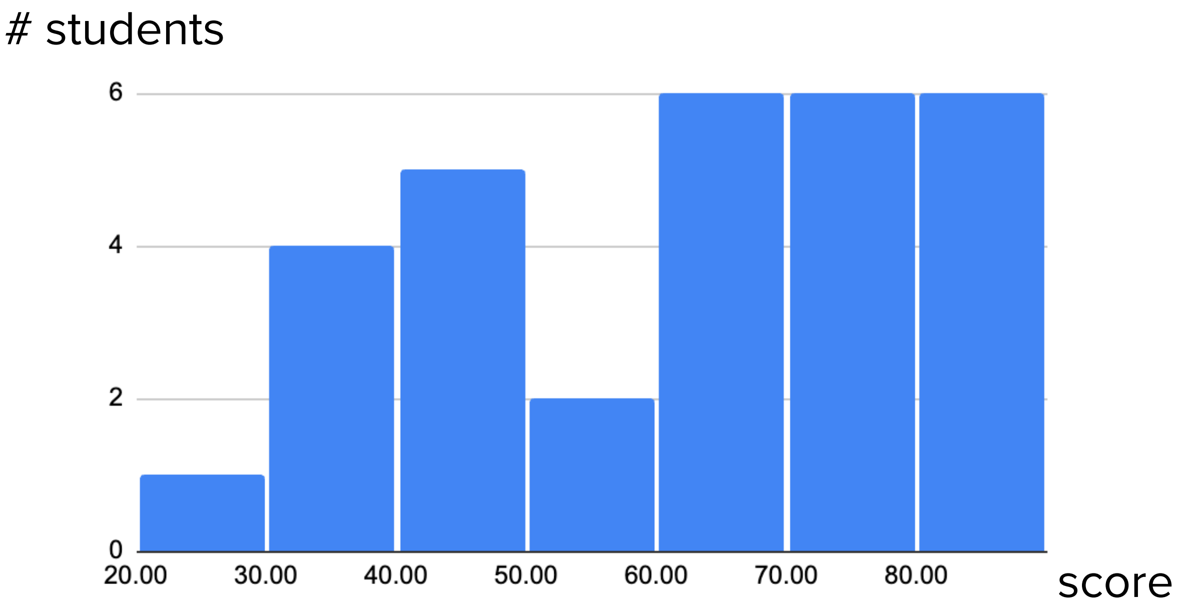
Seulbae Kim

POSTECH

POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Midterm exam

- Statistics
  - Max: 87
  - Average: 62
    - (vs 46 in 2024)



- To dispute:
  - Please meet me after today's class

Administrivia

- Lab 04 has been released!
  - About password-based authentication and entropy
  - Due on **April 25**

Recap

- Password-based authentication
  - Most widely used authentication method
  - Very easy to use and deployable
- Passwords are valuable, but considered weak due to
  - Human factors
  - Inevitable brute-force attacks
  - Incorrect policy

Means of authentication

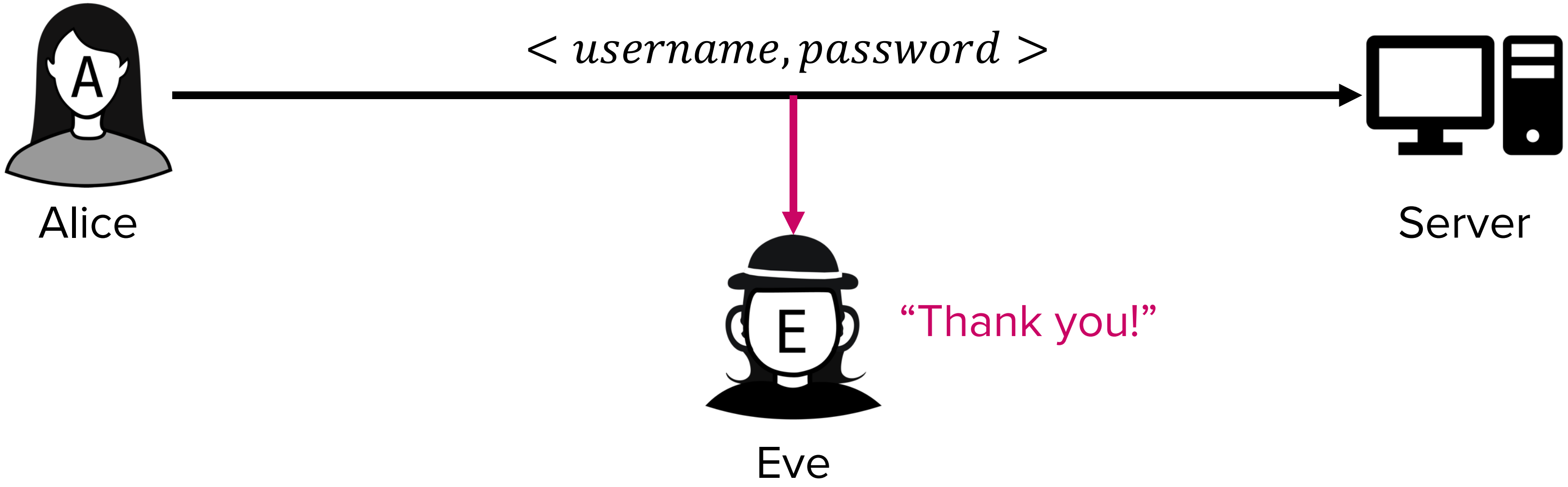
- Password-based ✅
- Challenge-response
- Biometric
- Zero-knowledge
- Multi-factor

Today's topic!

Challenge-Response Authentication

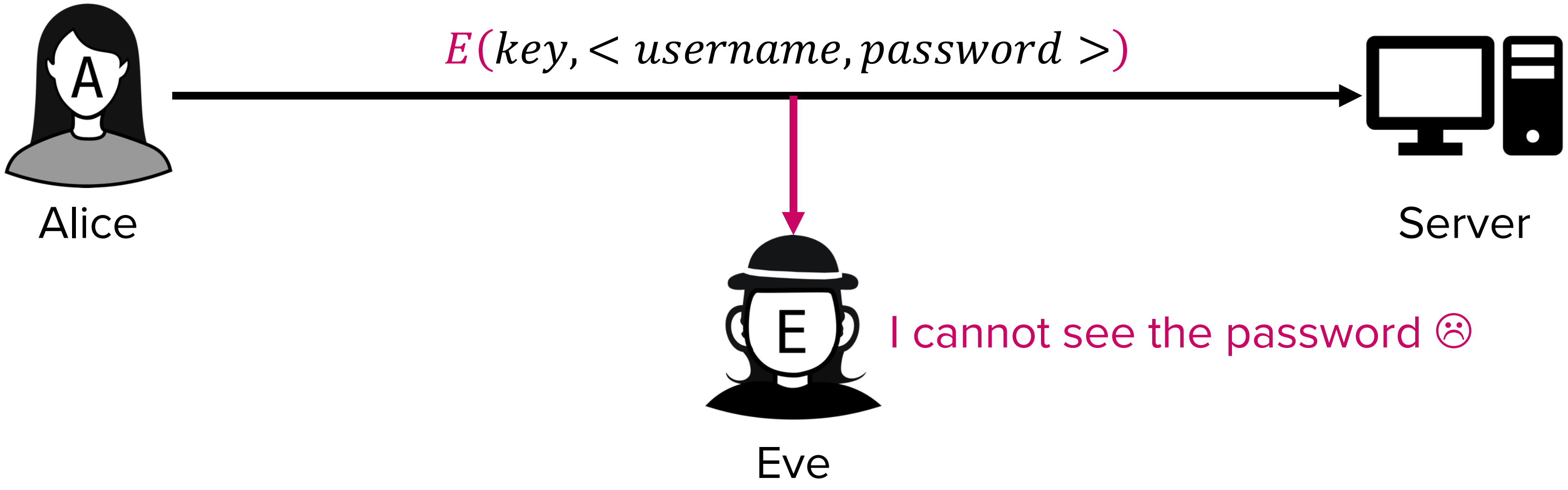
Transmitting a password

- How should a user transmit a password to a system?
  - Worst idea: Send the password in the clear (as plaintext)



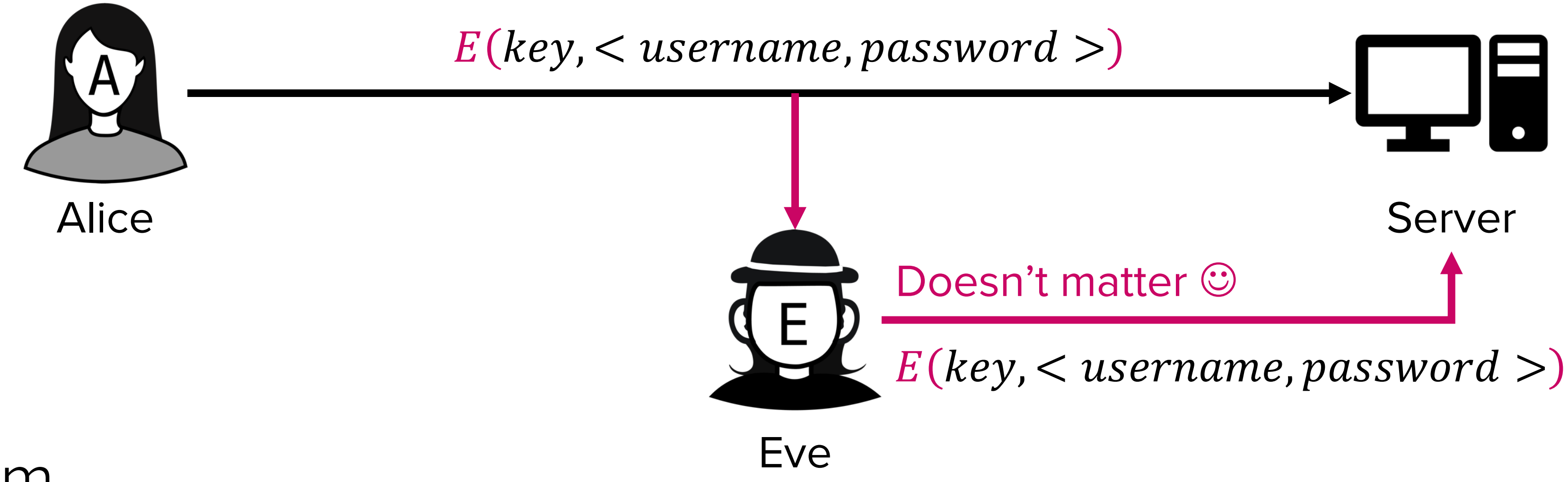
Transmitting a password

- How should a user transmit a password to a system?
  - Slightly better idea: Send the encrypted password



Transmitting a password

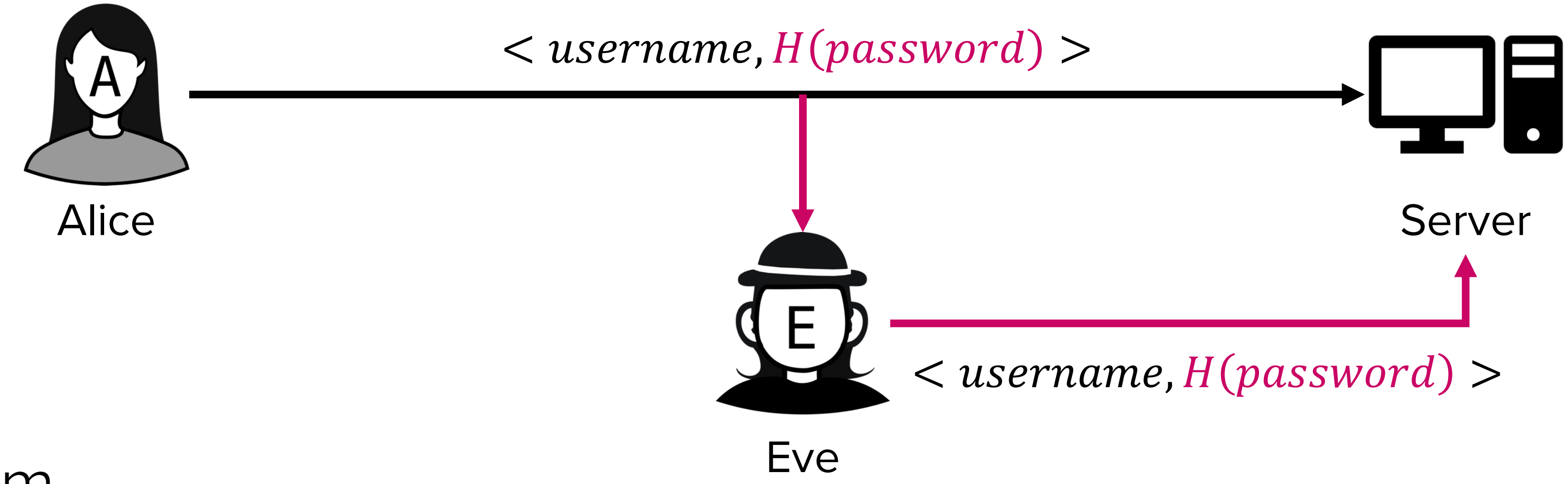
- How should a user transmit a password to a system?
  - Slightly better idea: Send the encrypted password


$E(key, < username, password >)$
A
Alice
Server
E
Eve
Doesn't matter ☺
$E(key, < username, password >)$

- Problem
  - An MitM attacker can record and replay the identification

Transmitting a password

- How should a user transmit a password to a system?
  - Another idea: Send the hashed password



$< username, H(password) >$

Alice

Server

Eve

$< username, H(password) >$

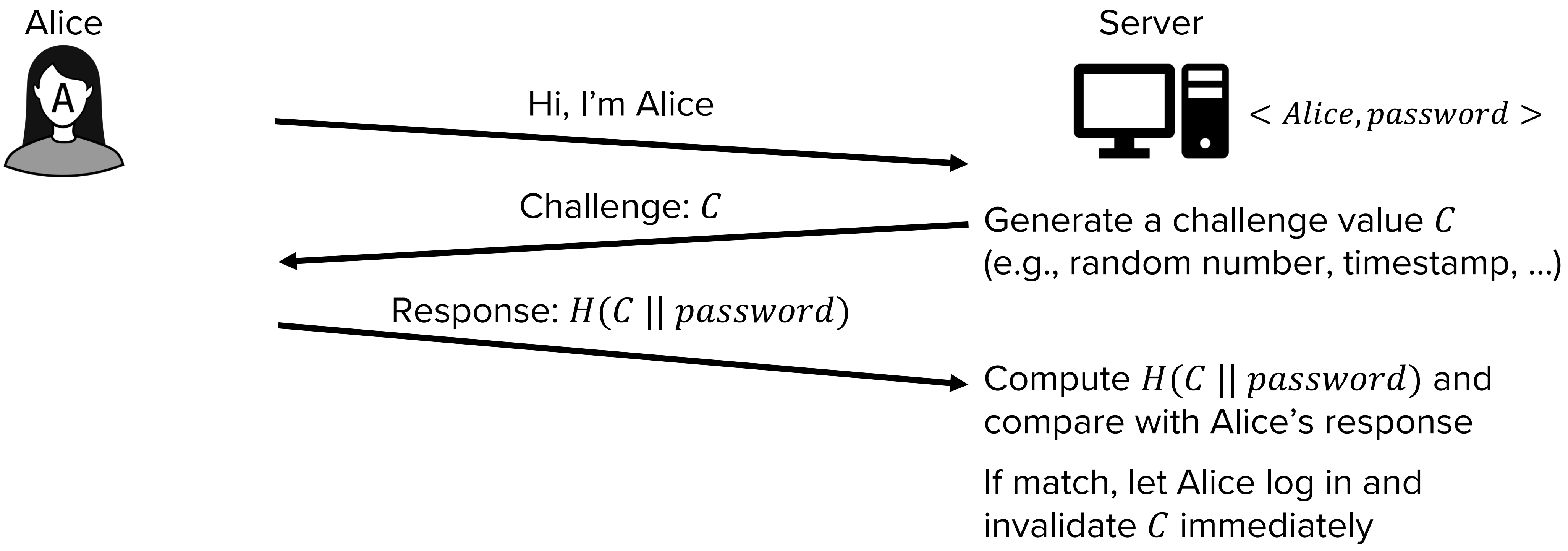
  - Problem
    - Hashing does not improve security, since the hash can also be replayed

Transmitting a password

- How should a user transmit a password to a system?
  - Encryption and hashing do not automatically add security
  - A better idea: Challenge-response protocol

Challenge-response authentication

- Idea



Alice

Server

$< Alice, password >$

Hi, I'm Alice

Challenge: C

Generate a challenge value C
(e.g., random number, timestamp, …)

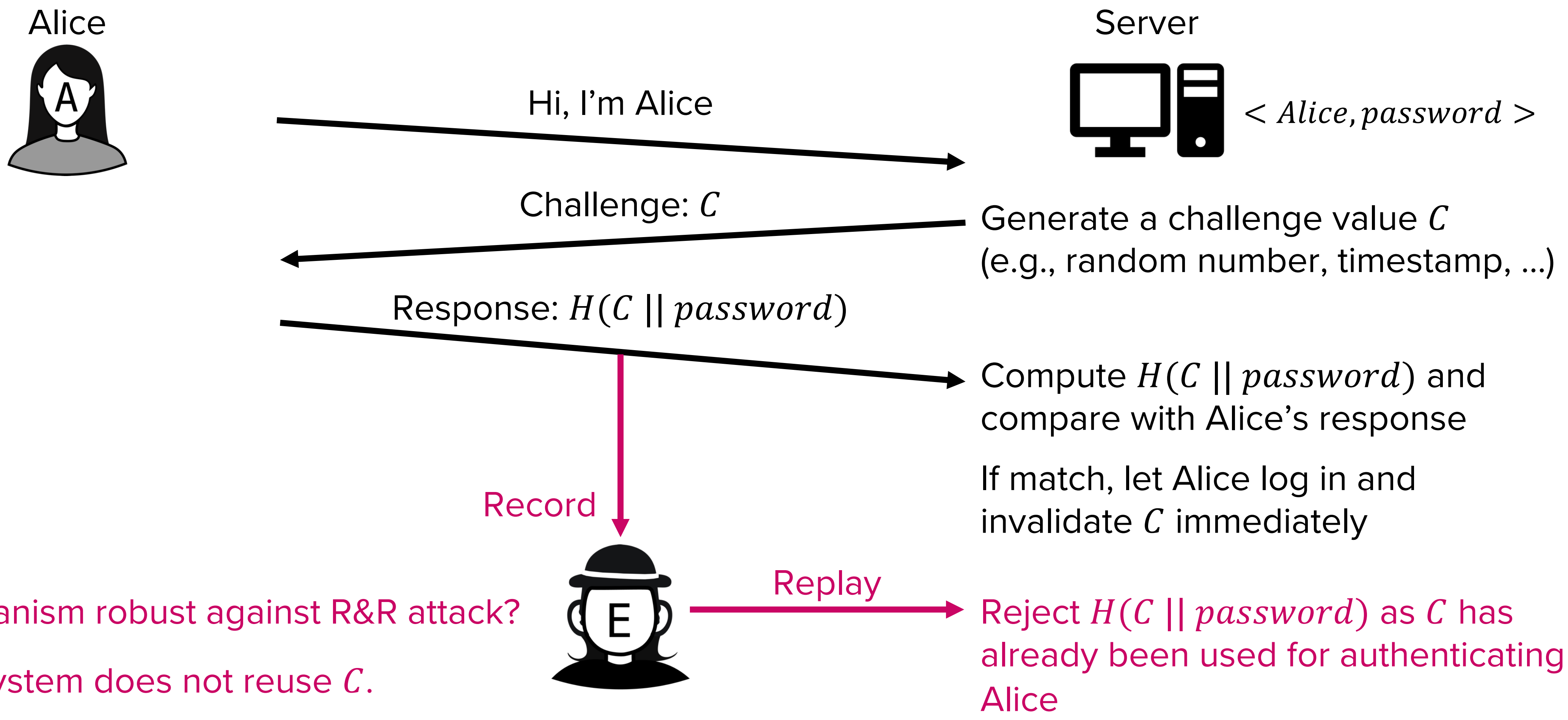
Response: $H(C \mid\mid password)$

Compute $H(C \mid\mid password)$ and compare with Alice's response

If match, let Alice log in and invalidate C immediately

Challenge-response authentication

- Idea



Alice

Server

$< Alice, password >$

Hi, I’m Alice

Challenge: C

Generate a challenge value C (e.g., random number, timestamp, …)

Response: $H(C \mid\mid password)$

Compute $H(C \mid\mid password)$ and compare with Alice’s response

If match, let Alice log in and invalidate C immediately

Record

Replay

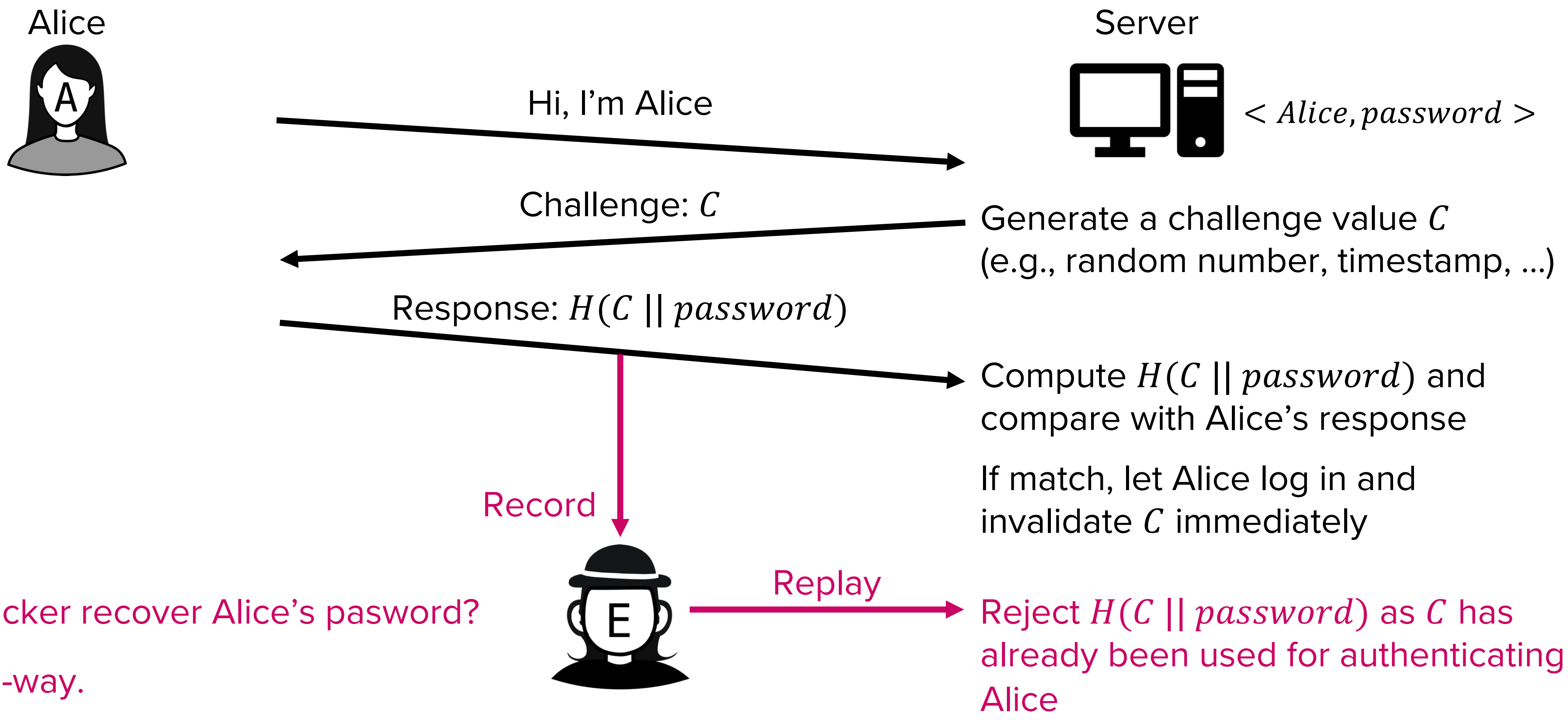
Reject $H(C \mid\mid password)$ as C has already been used for authenticating Alice

Q) Is the mechanism robust against R&R attack?

A) Yes! If the system does not reuse C.

Challenge-response authentication

- Idea



Alice

Server

$< Alice, password >$

Hi, I'm Alice

Challenge: C

Generate a challenge value C (e.g., random number, timestamp, …)

Response: $H(C \,||\, password)$

Compute $H(C \,||\, password)$ and compare with Alice's response

If match, let Alice log in and invalidate C immediately

Record

Replay

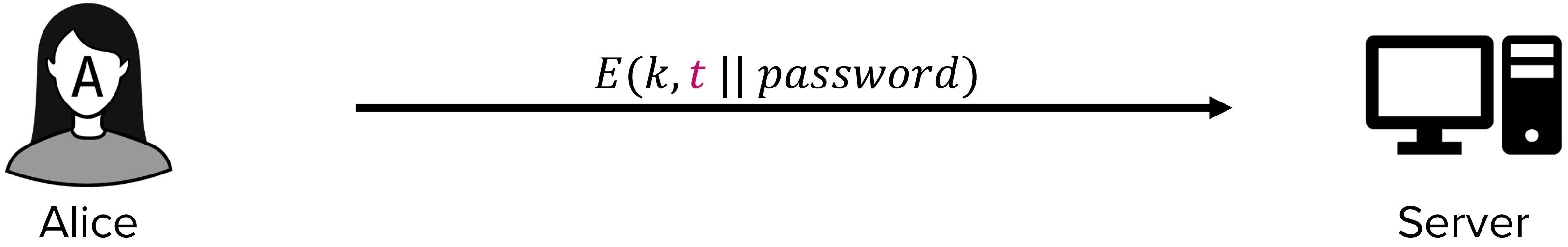
Reject $H(C \,||\, password)$ as C has already been used for authenticating Alice

Q) Can the attacker recover Alice's pasword?

A) No! H is one-way.

Challenge-response in practice

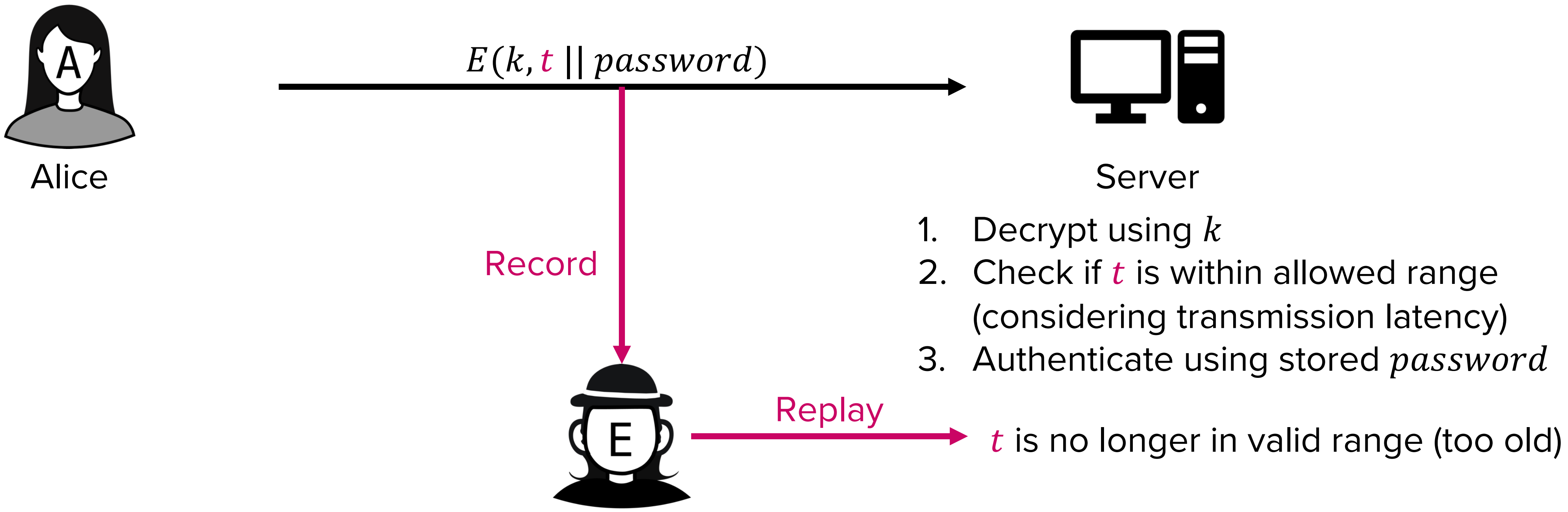
- Symmetric key-based implementation (1)
    - Using shared key k and timestamp t (current time)



1. Decrypt using k
2. Check if t is within allowed range (considering transmission latency)
3. Authenticate using stored $password$

Challenge-response in practice

- Symmetric key-based implementation (1)
  - Using shared key k and timestamp t (current time)



$E(k, t \,||\, password)$

Alice

Server

1. Decrypt using k
2. Check if t is within allowed range (considering transmission latency)
3. Authenticate using stored $password$

Record

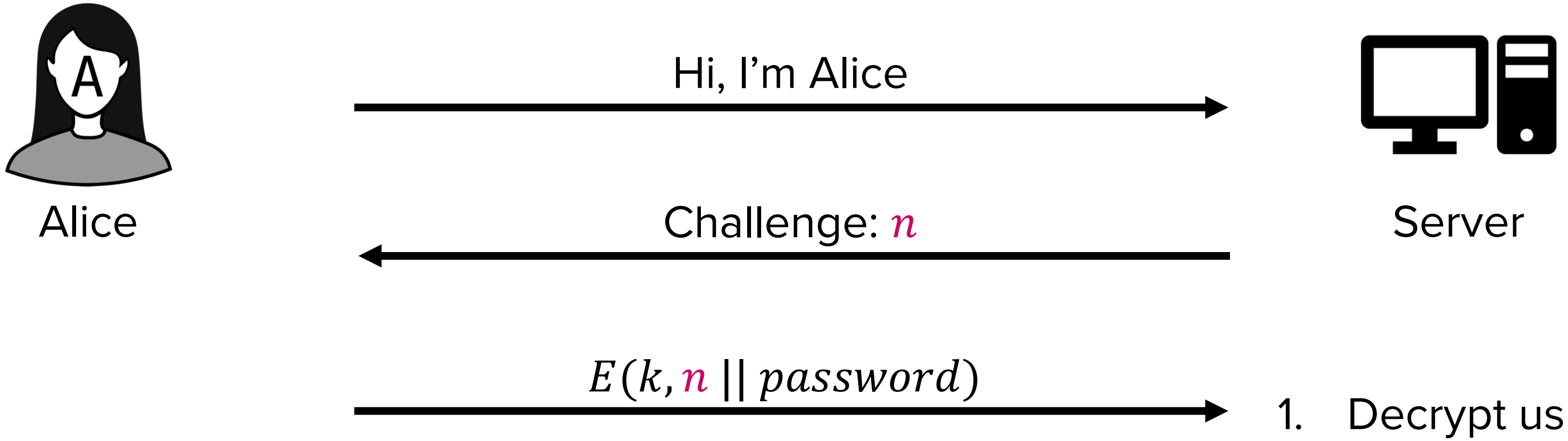
Replay

t is no longer in valid range (too old)

Cannot recover $password$ without knowing the shared key k

Challenge-response in practice

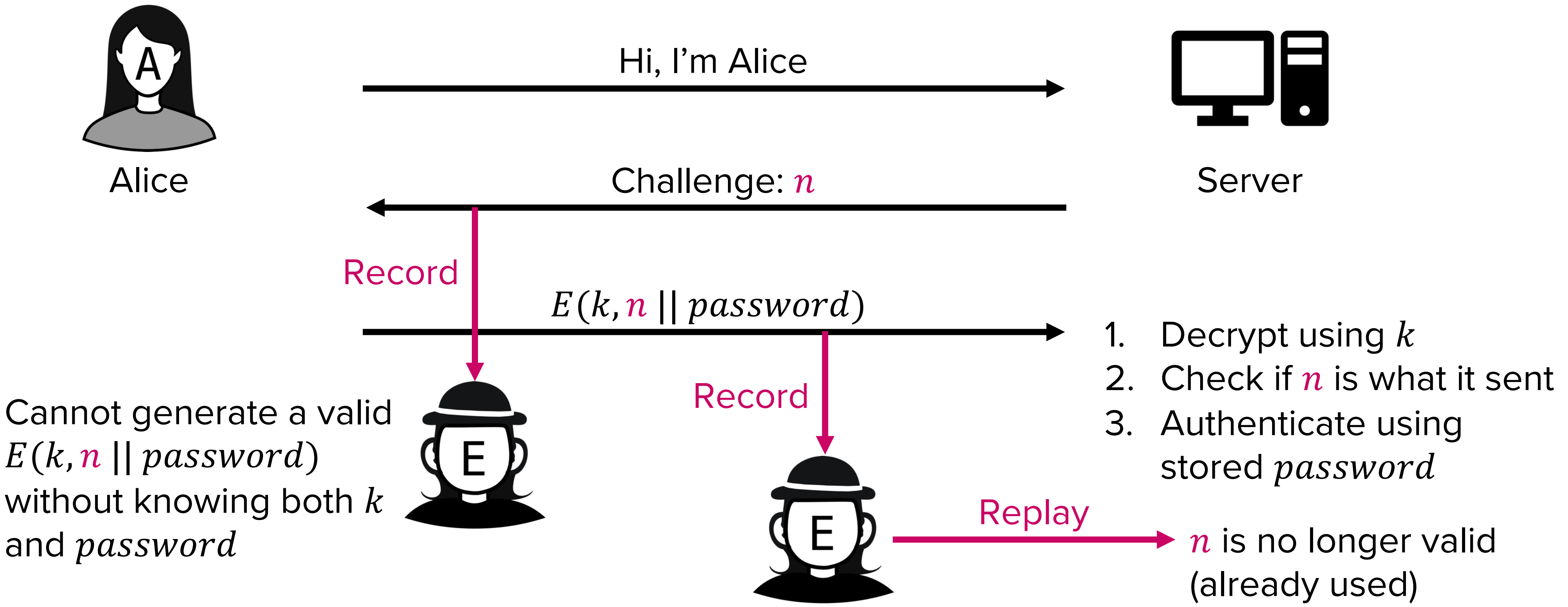
- Symmetric key-based implementation (2)
  - Using shared key k and a nonce n (random number)



1. Decrypt using k
2. Check if n is what it sent
3. Authenticate using stored $password$

Challenge-response in practice

- Symmetric key-based implementation (2)
  - Using shared key k and a nonce n (random number)



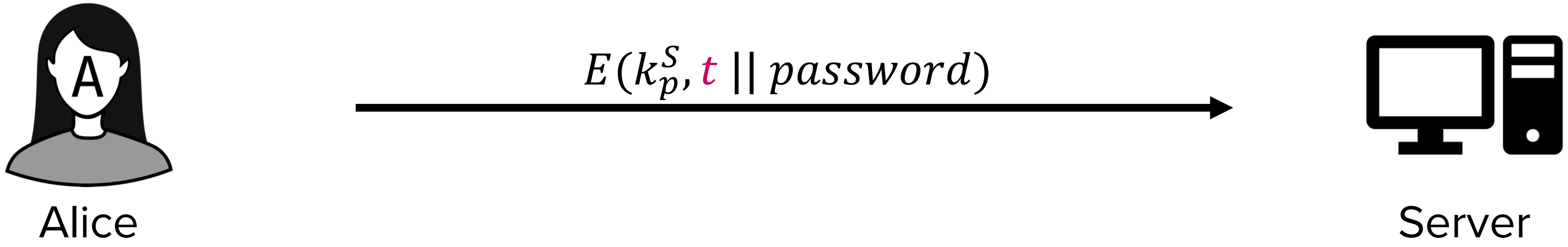
1. Decrypt using k
2. Check if n is what it sent
3. Authenticate using stored $password$

Cannot generate a valid $E(k, n \,||\, password)$ without knowing both k and $password$

n is no longer valid (already used)

Challenge-response in practice

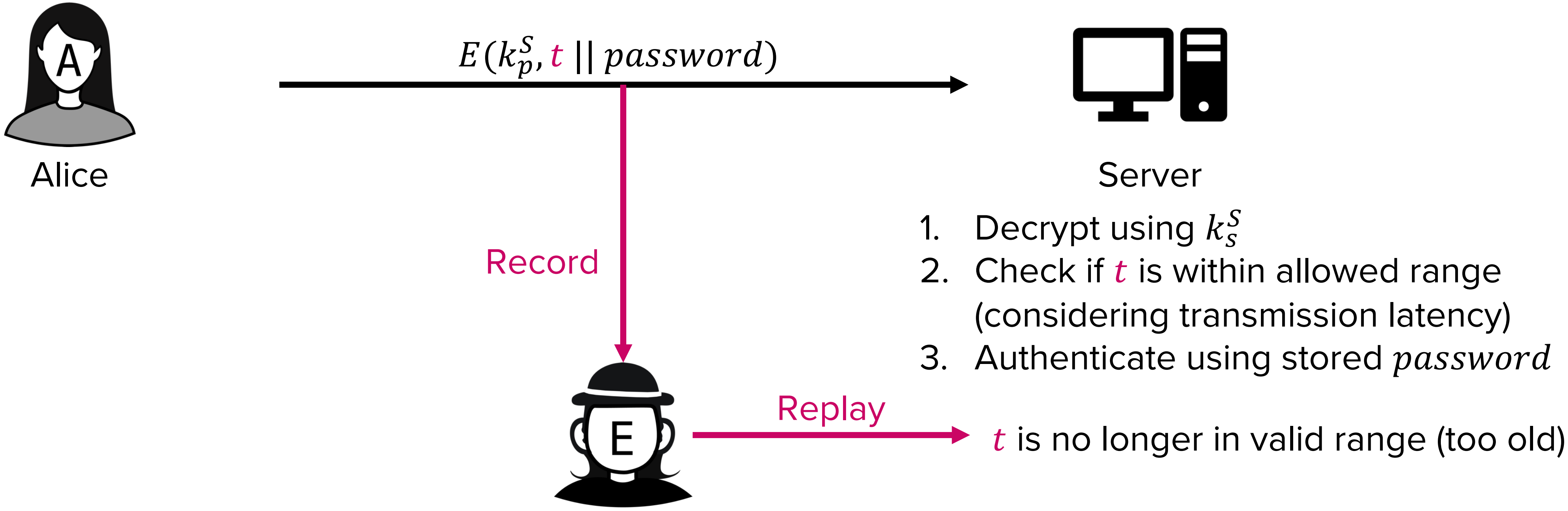
- Asymmetric key-based implementation (1)
  - Using public key k_p^S, secret key k_s^S, and timestamp t (current time)



1. Decrypt using k_s^S
2. Check if t is within allowed range (considering transmission latency)
3. Authenticate using stored $password$

Challenge-response in practice

- Asymmetric key-based implementation (1)
  - Using public key k_p^S, secret key k_s^S, and timestamp t (current time)



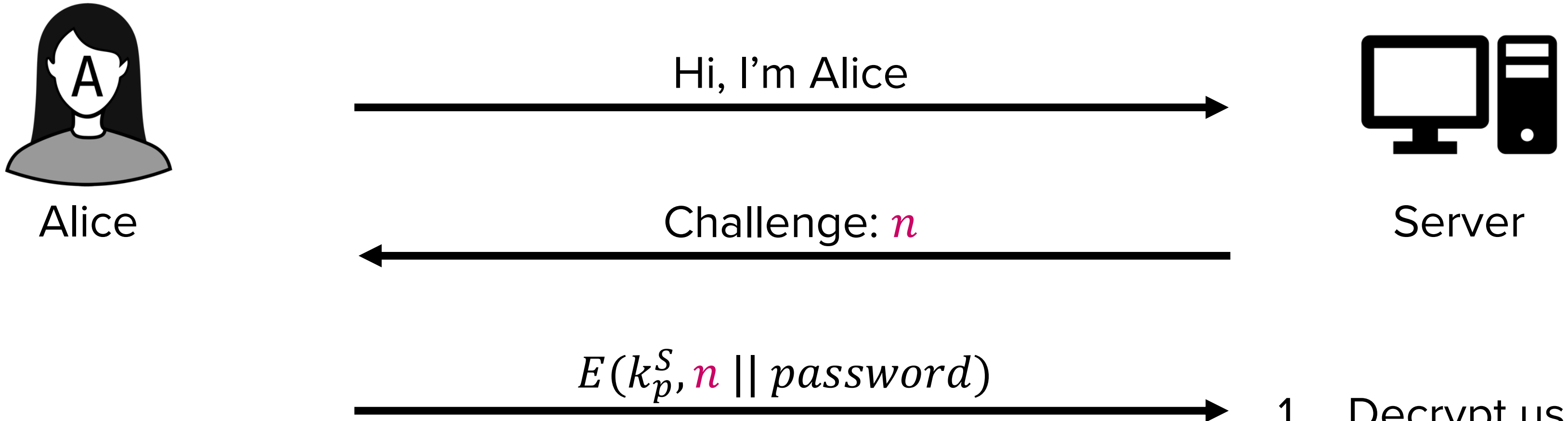
1. Decrypt using k_s^S
2. Check if t is within allowed range (considering transmission latency)
3. Authenticate using stored $password$

t is no longer in valid range (too old)

Cannot recover $password$ without knowing server's secret key k_s^S

Challenge-response in practice

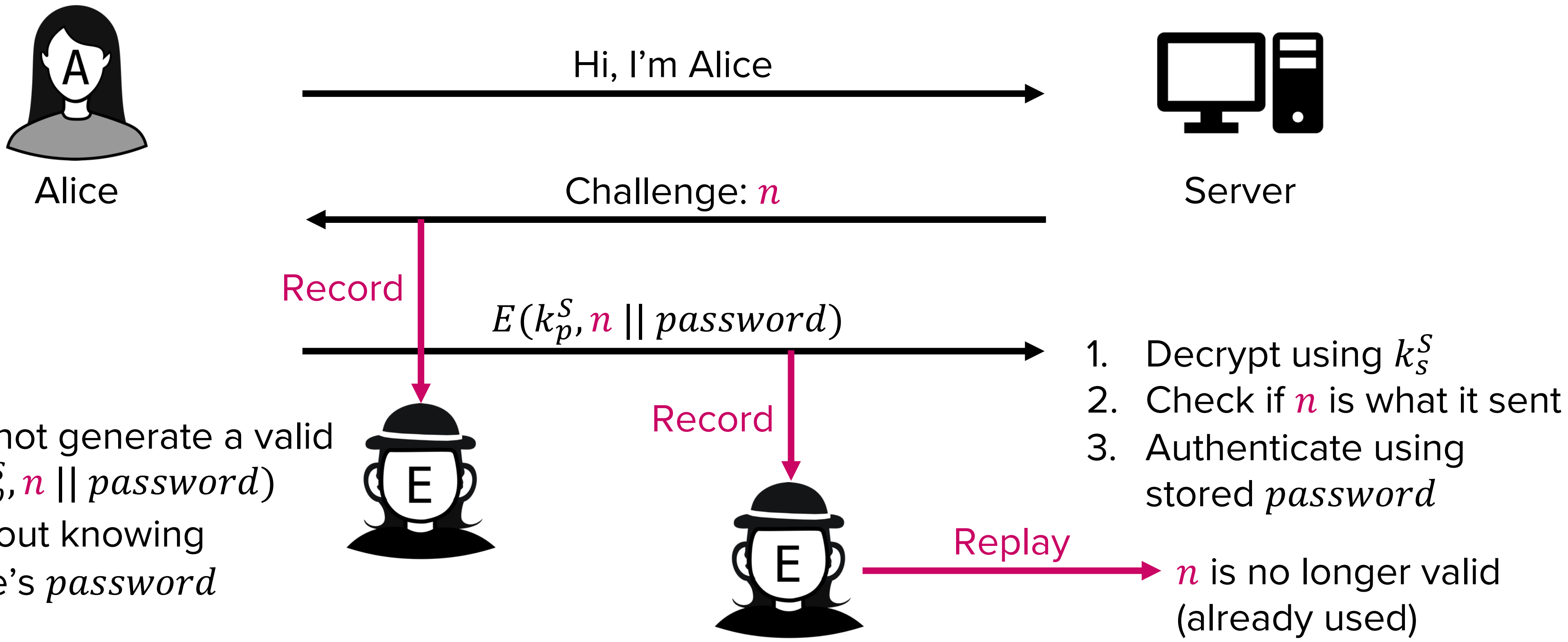
- Asymmetric key-based implementation (2)
  - Using public key k_p^S, secret key k_s^S, and a nonce n



1. Decrypt using k_s^S
2. Check if n is what it sent
3. Authenticate using stored $password$

Challenge-response in practice

- Asymmetric key-based implementation (2)
  - Using public key k_p^S, secret key k_s^S, and a nonce n



1. Decrypt using k_s^S
2. Check if n is what it sent
3. Authenticate using stored $password$

Cannot generate a valid $E(k_p^S, n \,||\, password)$ without knowing Alice's $password$

n is no longer valid (already used)

Biometric Authentication

Biometric authentication

- Use “something you are” for authentication
  - Authenticate users based on their unique physical characteristics
  - Characteristics include
    - Facial characteristics (e.g., Apple’s Face ID)
    - Fingerprints (e.g., Apple’s Touch ID)
    - Retina (Pattern of retinal blood vessels)
    - Iris
    - Voice

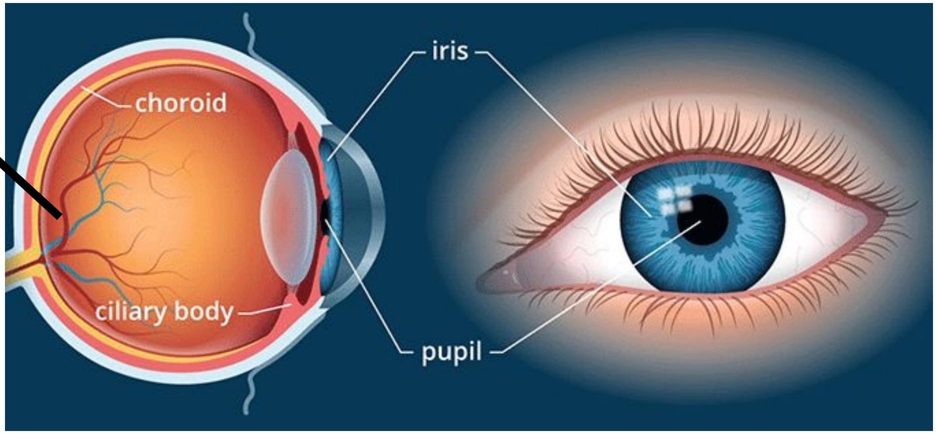


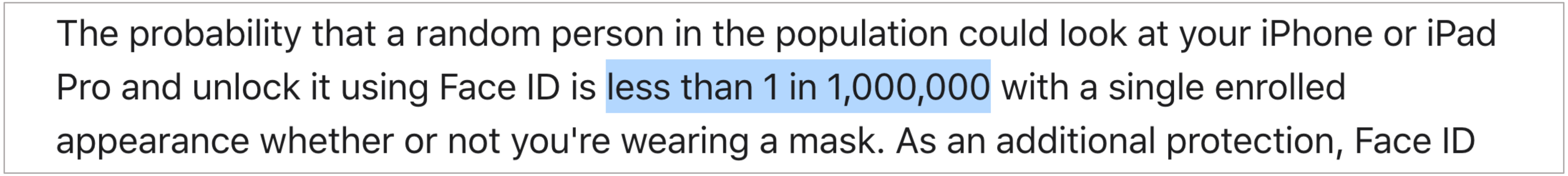
Image from All About Vision

Biometric authentication

- Advantages of using something you are for authentication
  - No need to remember anything (== can never forget the secret)
  - No need to carry anything (== can never lose the secret)
- Problems
  - Once compromised, cannot easily be changed
  - Not as accurate as digital methods (e.g., password matching)
  - Authentication is costly
  - Biometric information is considered more sensitive than a password
    - Your personal data needs to be stored on the service

Biometric authentication

- Problems
  - **Accuracy**: Not as accurate as digital methods, such as password matching
    - https://www.youtube.com/watch?v=e8-yupM-6Oc



The probability that a random person in the population could look at your iPhone or iPad Pro and unlock it using Face ID is less than 1 in 1,000,000 with a single enrolled appearance whether or not you're wearing a mask. As an additional protection, Face ID

https://support.apple.com/en-us/102381

Biometric authentication

- Problems
  - **Recovery**: If stolen or compromised, it is very hard to change biometric information
  - **Cost**: Authentication is slow and costly
    - Need a dedicated hardware (e.g., retina scanner, LiDAR, etc.)
  - **Privacy**: Your biometric data needs to be stored on the service
    - Biometric information is considered more sensitive than a password

Zero-knowledge Authentication

Your identity matters

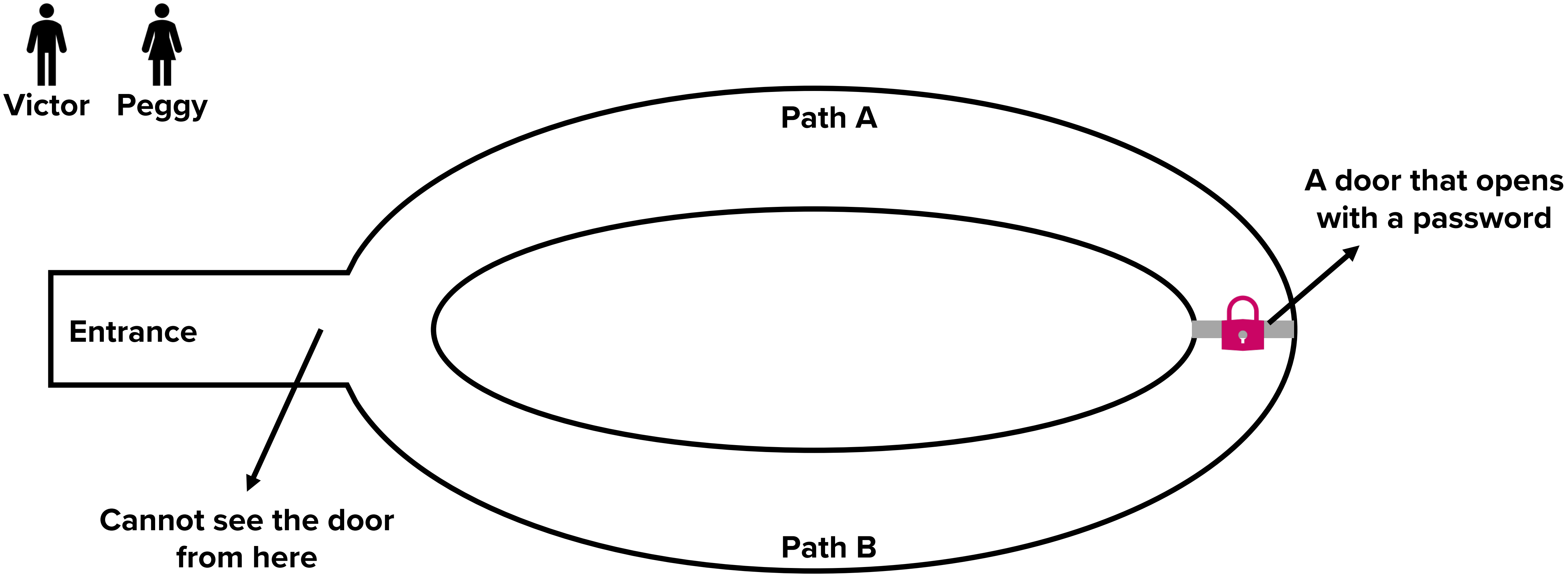
- Problem of existing authentication methods
  - Your identity must be revealed during authentication
    - What you know (password / challenge-response)
    - What you have (token)
    - What you are (biometric information)

Zero-knowledge proofs (ZKP)

- Problem setting
  - Peggy is a prover and Victor is a verifier
  - Peggy wants to prove to Victor that **she knows the secret**
  - However, she does not want to reveal any other information to Victor
    - Including the secret itself

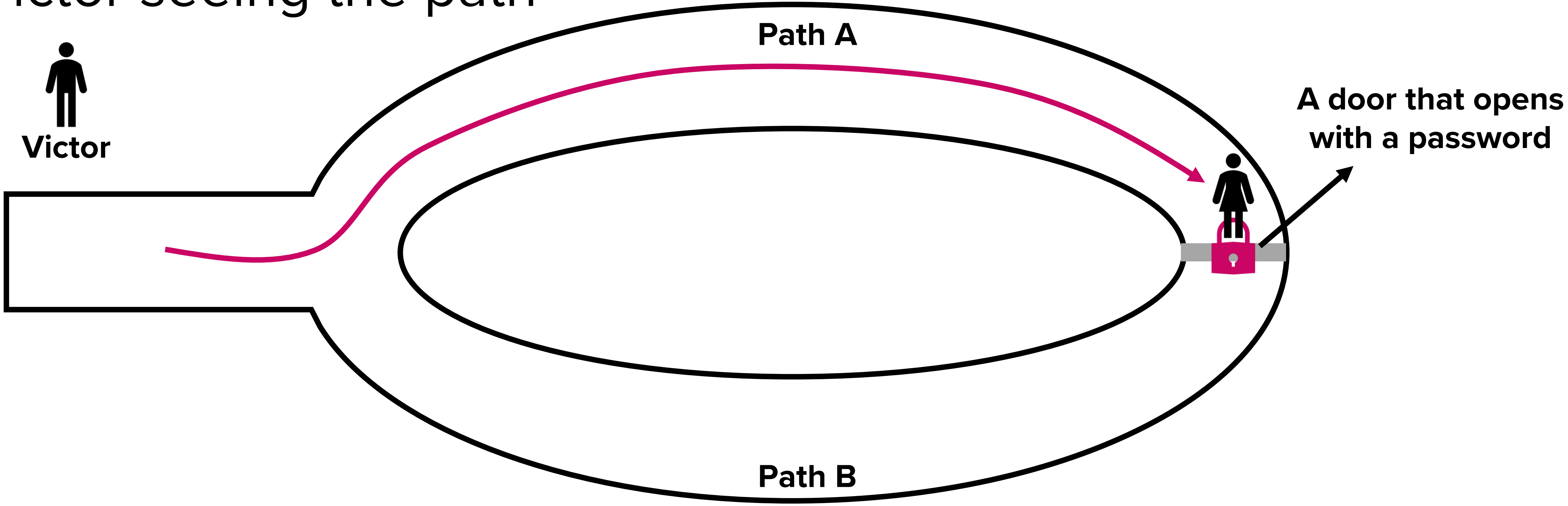
→ Can Peggy authenticate without revealing her identity?

The Ali Baba cave example


Victor
Peggy
Path A
A door that opens with a password
Entrance
Cannot see the door from here
Path B

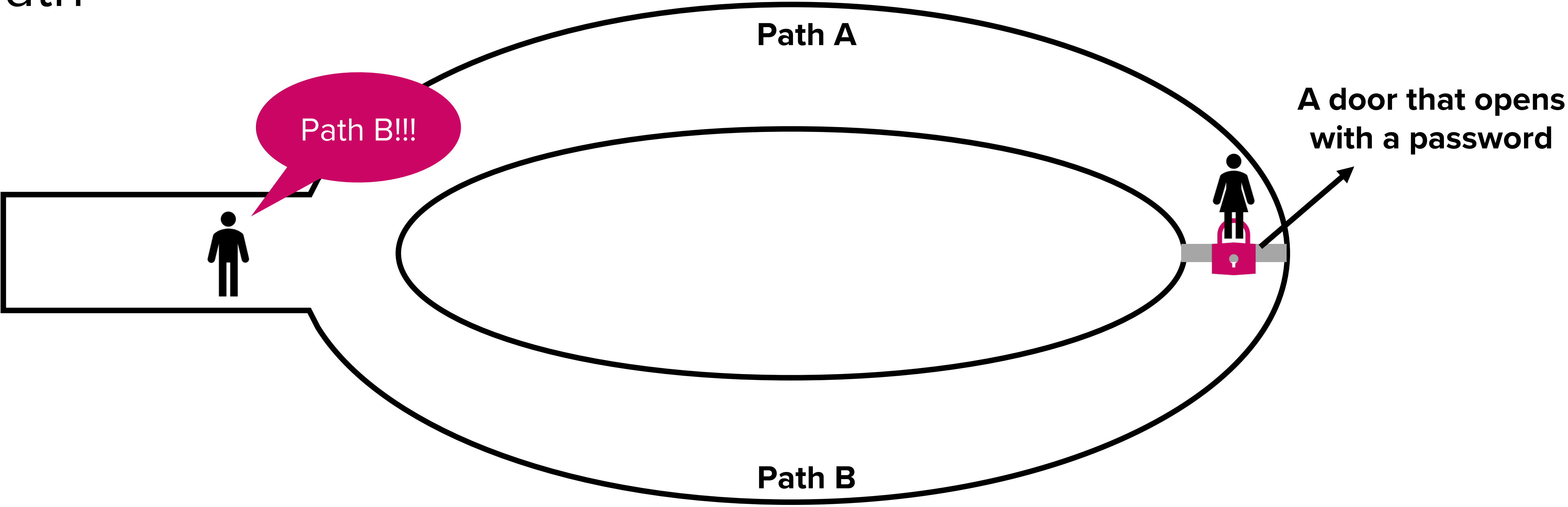
The Ali Baba cave example

1. Peggy enters the cave and randomly selects a path w/o Victor seeing the path



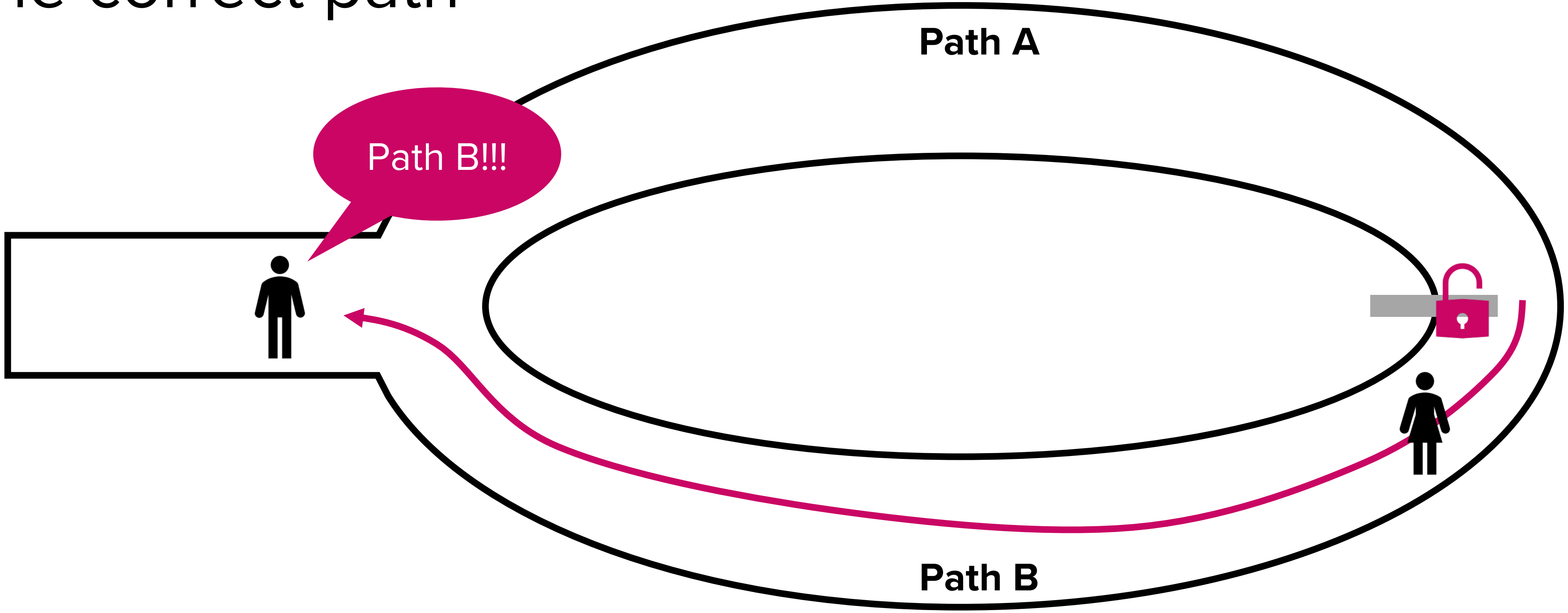
The Ali Baba cave example

2. Victor enters and shouts the name of the randomly selected path



The Ali Baba cave example

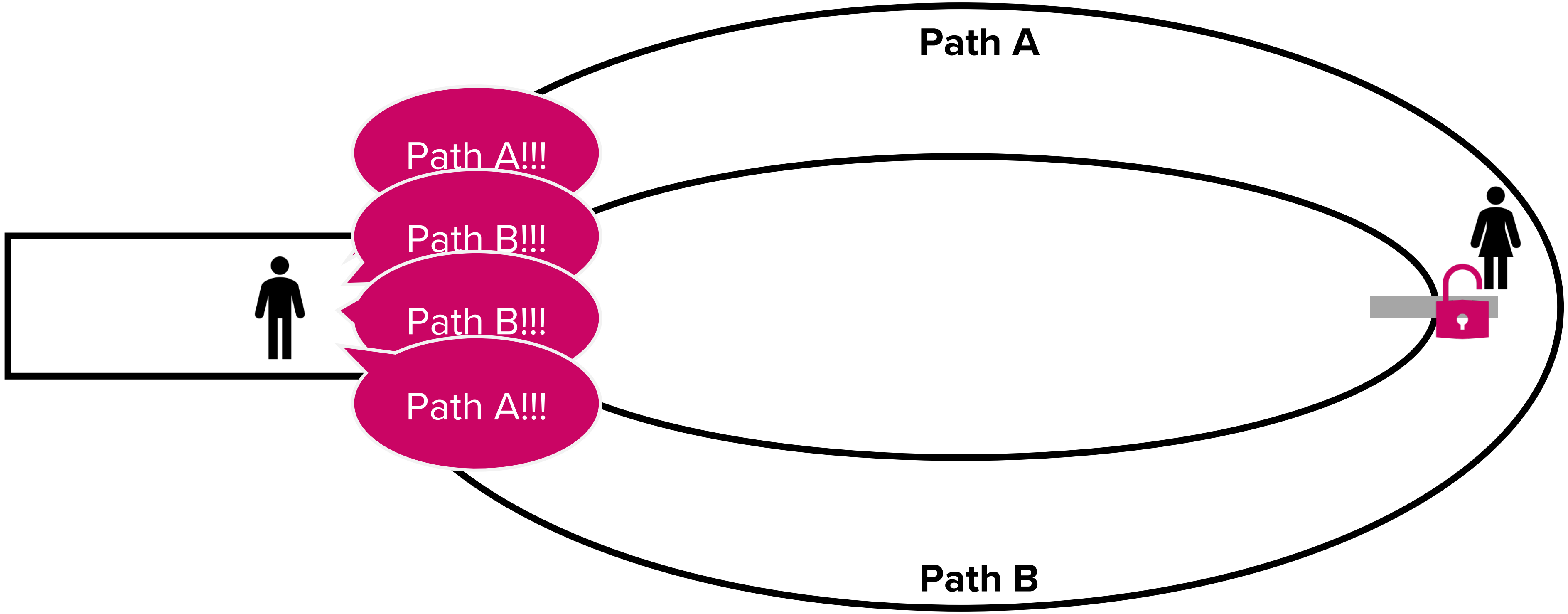
3. If Peggy knows the password, she can return to Victor using the correct path



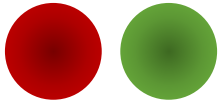
* If Peggy doesn't know the password, she still has a 50% chance to succeed

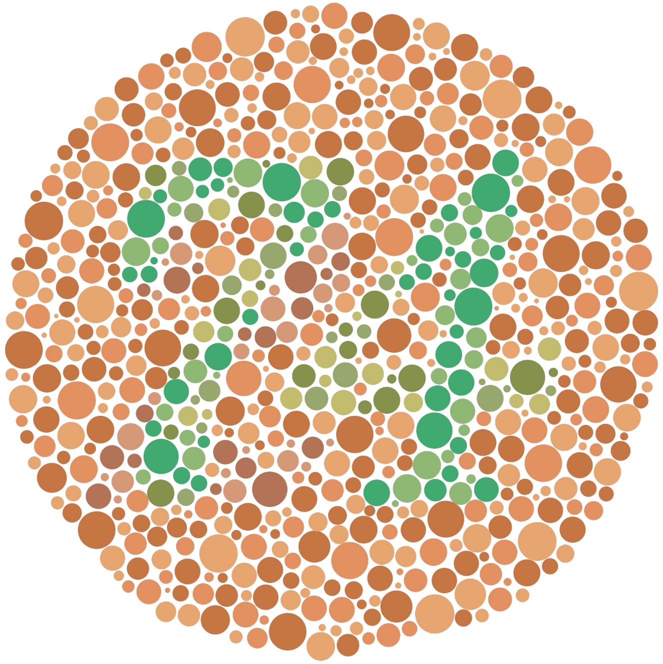
The Ali Baba cave example

4. Repeat multiple times until Victor is confident



Color-blind Victor example

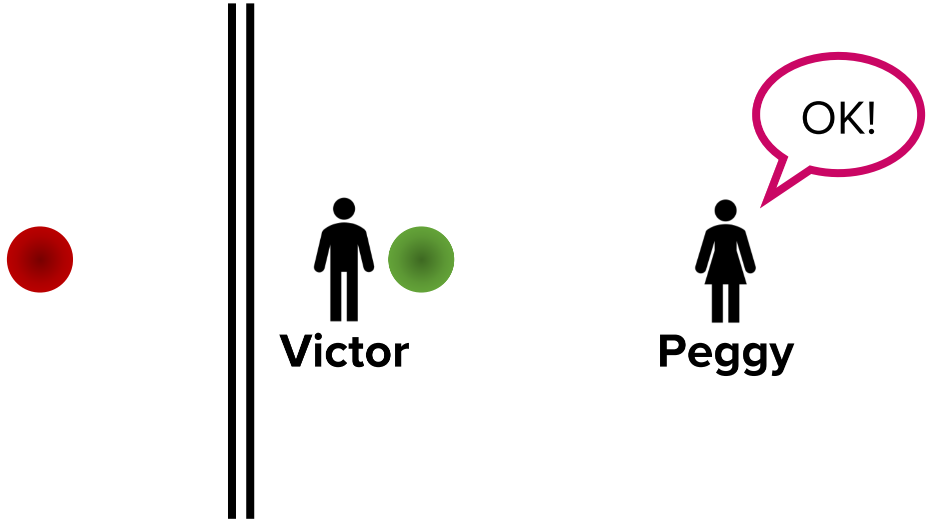
- Victor has a “red-green color blindness”
  - He cannot tell red from green
- Setting
  - Prepare two balls 
    - One red ball, one green ball
    - All properties (weight, size, …) are identical except for the color
  - Peggy should prove to Victor that the two balls have different colors



Ishihara Plate #9

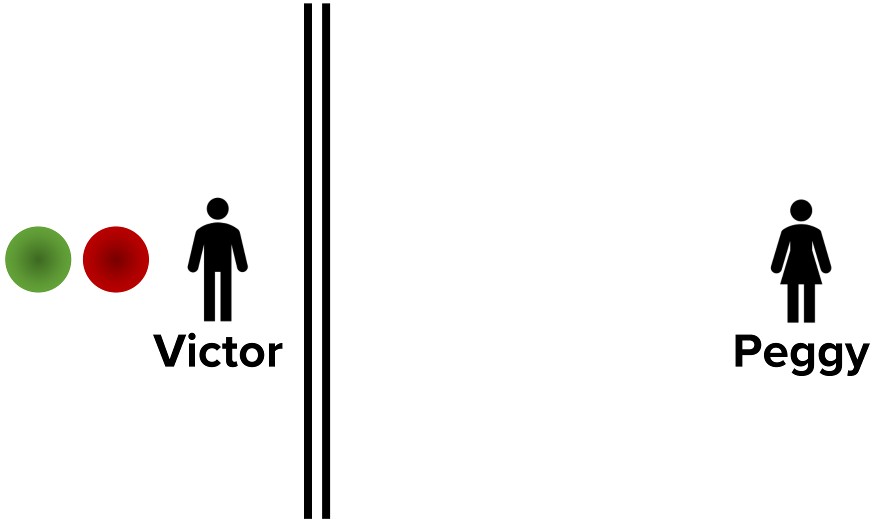
Color-blind Victor example

1. Victor randomly selects a ball and shows it to Peggy



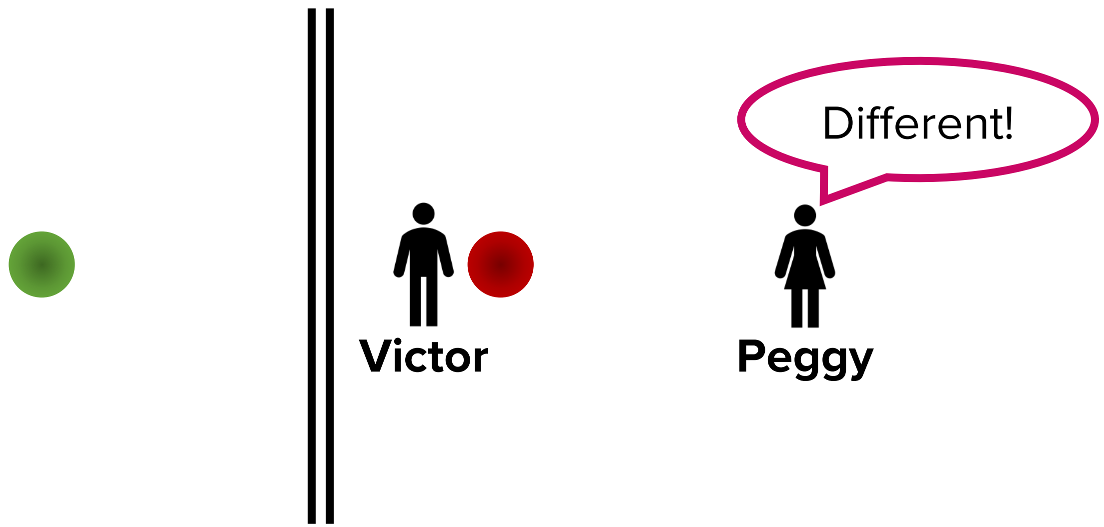
Color-blind Victor example

2. Victor enters a room and makes a random decision about switching the ball (switch or not switch)



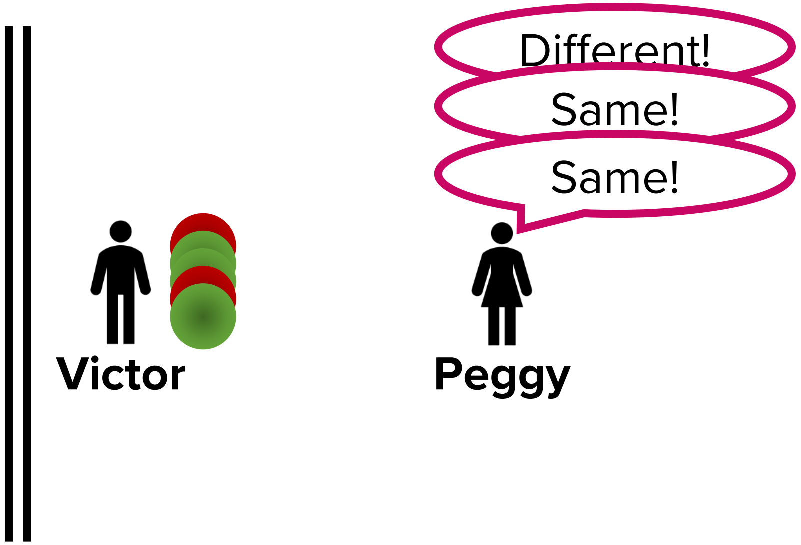
Color-blind Victor example

3. Victor shows the ball to Peggy and asks if he switched the balls



Color-blind Victor example

4. Repeat steps 1-3 until Victor is confident



Color-blind Victor example

- Probability that Peggy is also color-blind but gets the answer right is 50%
  - Experiment repeated 10 times, probability that Peggy does not know the secret becomes $\frac{1}{2^{10}}$ (less than 0.1%)
- Victor learns that the two balls are distinguishable without learning the color of each ball

ZKP for user authentication

- Secure Remote Password (SRP) protocol
  - User authentication using ZKP
  - Server does not store user's password
  - Server verifies that the user knows the password w/o seeing the password (zero-knowledge!)

ZKP for user authentication

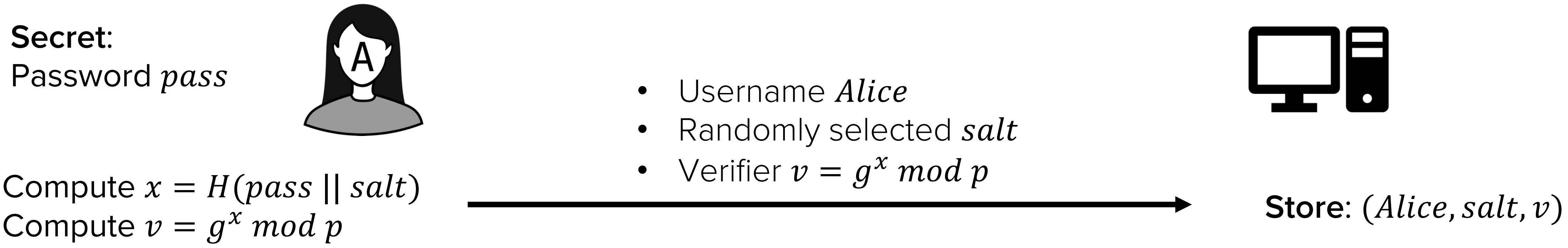
- Recap: Diffie-Hellman key exchange
  - Public information
    - Large prime number p and its generator g
  - Secret information
    - Alice's secret key a and Bob's secret key b
  - Exchange
    - Alice sends $A = g^a \mod p$ to Bob, Bob sends $B = g^b \mod p$ to Alice
  - Key derivation
    - Alice derives a shared key $k = B^a \mod p = g^{ab} \mod p$
    - Bob derives the same shared key $k = A^b \mod p = g^{ab} \mod p$

Public	Secret
$p = 11$	$a = 15$
$g = 6$	$b = 8$
$A = 10$	$k = 1$
$B = 4$	

SRP protocol

Public: Prime p, generator g

- Step 1: Registration



Secret:
Password $pass$

Compute $x = H(pass \,||\, salt)$
Compute $v = g^{x}\ mod\ p$

- Username $Alice$
- Randomly selected $salt$
- Verifier $v = g^{x}\ mod\ p$

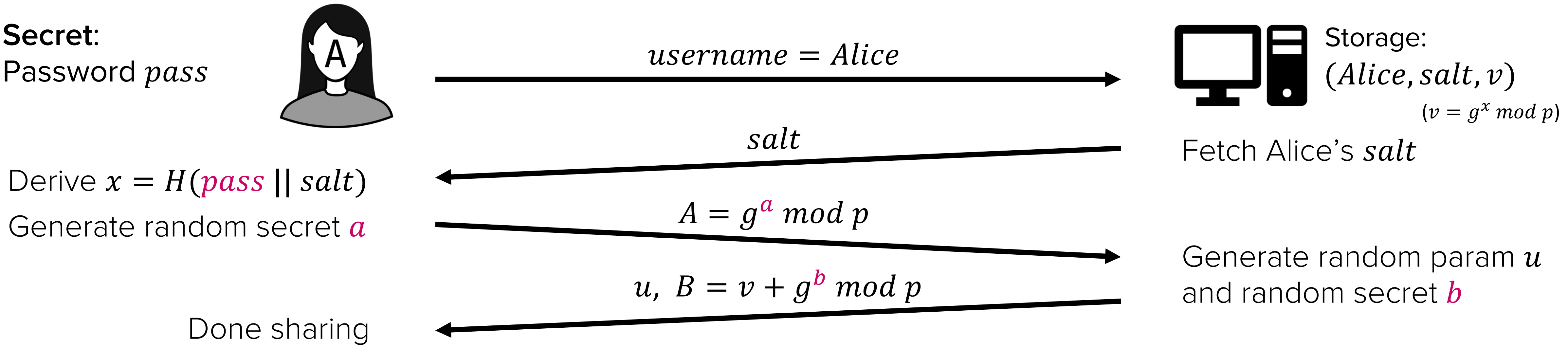
Store: $(Alice, salt, v)$

SRP protocol

Public: Prime p, generator g

$salt \quad A \quad u \quad B$

- Step 2: Parameter sharing



Secret:
Password $pass$

Derive $x = H(pass \,||\, salt)$

Generate random secret a

Done sharing

$username = Alice$

$salt$

$A = g^a \bmod p$

$u, \ B = v + g^b \bmod p$

Storage:
$(Alice, salt, v)$
$(v = g^x \bmod p)$

Fetch Alice's $salt$

Generate random param u and random secret b

SRP protocol

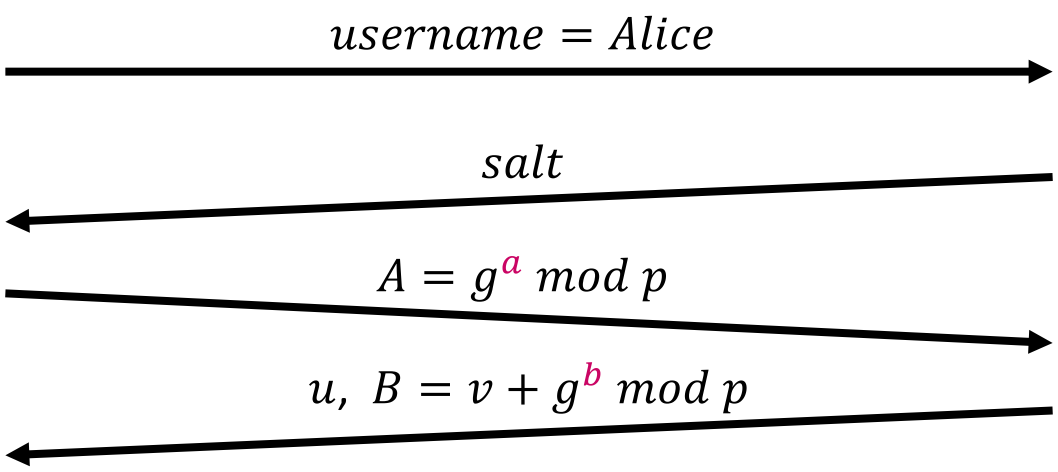
Public: Prime p, generator g

$salt \quad A \quad u \quad B$

- Step 3: Session key derivation

Secret:
Password $pass$

Storage:
$(Alice, salt, v)$
$(v = g^x \bmod p)$



$username = Alice$

$salt$

Fetch Alice's $salt$

Derive $x = H(pass \,||\, salt)$

Generate random secret a

$A = g^a \bmod p$

Generate random param u and random secret b

$u, \; B = v + g^b \bmod p$

Done sharing

Derive:

- $S = (B - g^x \bmod p)^{a+ux}$
- Session key $K = H(S)$

Same key has been derived without revealing $pass$

Derive:

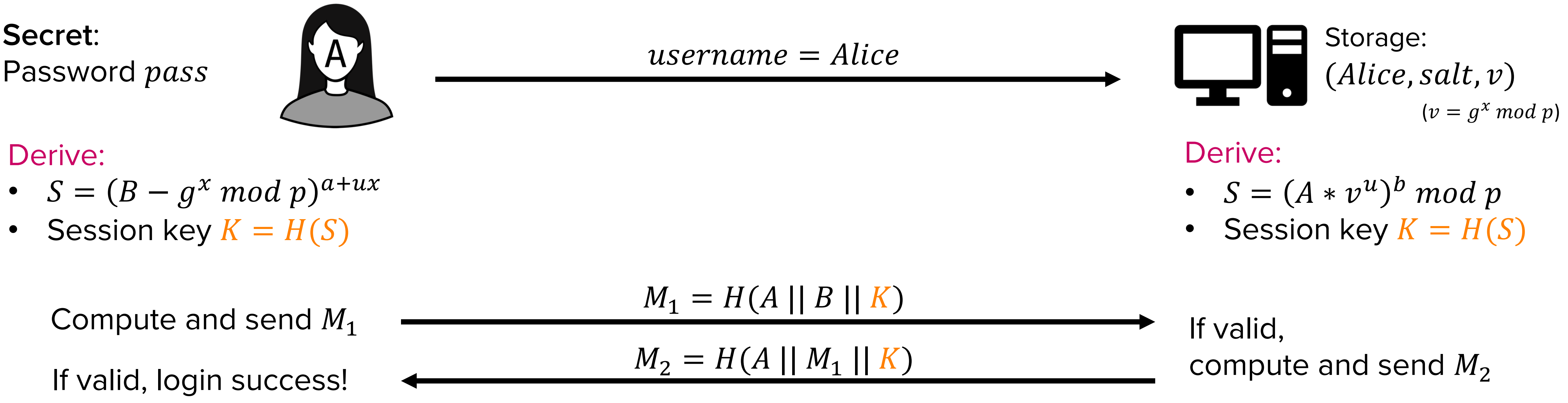
- $S = (A * v^u)^b \bmod p$
- Session key $K = H(S)$

SRP protocol

- Step 4: Mutual authentication

Public: Prime p, generator g

$salt \quad A \quad u \quad B$



Secret:
Password $pass$

$username = Alice$

Storage:
$(Alice, salt, v)$
$(v = g^x \; mod \; p)$

Derive:

- $S = (B - g^x \; mod \; p)^{a+ux}$
- Session key $K = H(S)$

Derive:

- $S = (A * v^u)^b \; mod \; p$
- Session key $K = H(S)$

Compute and send M_1

$M_1 = H(A \,||\, B \,||\, K)$

If valid,
compute and send M_2

$M_2 = H(A \,||\, M_1 \,||\, K)$

If valid, login success!

SRP protocol

- Strengths
  - Resistant to leaks
    - Server does not store any password
  - Resistant to dictionary attacks
    - $pass$ or $x = H(pass \,||\, salt)$ are never sent in public
  - Resistant to active attacks
    - Mallory cannot derive the session key K from any publicly transmitted information
- Weakness
  - Slow!

Multi-factor Authentication

Multi-factor authentication (MFA)

- User provides two or more identifications
  - What you know (password) + what you are (fingerprint)
  - What you know (password) + what you also know (PIN)
  - …
- Fortifies inherently weak password-based authentication by providing an additional layer of security
  - Leaked passwords → Fingerprint cannot be leaked
  - Brute-forcing → Cannot brute-force fingerprint

Practical MFA implementation

- Password + One-time code sent via SMS message
  - Server stores the user’s phone number
  - Advantage:
    - Easy to implement
    - Compromised server does not automatically break security unless the user’s phone is also compromised
  - Disadvantage:
    - Phone network and carriers should be trusted
    - Could lead to phising attacks

Practical MFA implementation

- Password + One-time code sent via SMS message
    - Known attacks:
        - SIM swapping
            - Attacker collects various personal information of the victim
            - The attacker impersonates the victim and convinces the victim’s phone carrier to port the number to a new SIM card
            - The victim loses phone connection and the attacker’s phone is activated with the victim’s phone number
            - The attacker attempts to log into a service using victim’s leaked credentials
            - The attacker receives the one-time login code sent to the victim and breaks 2FA
            - The victim should make phone calls for recovery, but cannot do so without a number

Practical MFA implementation

- Password + Time-based one-time passwords (TOTP)
  - Server and user device agree on a secret value
    - Google's Authenticator app allows users to scan a QR code to register secret
  - User device generates $TOTP = H(secret \mid\mid cur\_time)$
    - Use coarse-grained time (e.g., cur_time is updated every 30 seconds)
  - User enters the $TOTP$ and server checks if it is valid
  - Advantages:
    - Do not need network connection, do not need to trust phone carriers
  - Disadvantages:
    - Needs extra steps for app installation and setup
    - If the server is compromised, all secret values need to be re-registered

Evaluating Authentication Method

Evaluating authentication method

- Metric for usability and security: Confusion matrix
  - True/False: Intended/Unintended
  - Positive/Negative: Allow/Disallow

		System	
		Allow	**Disallow**
User	**Alice logs in as Alice**	True Positive	False Negative
	Attacker logs in as Alice	False Positive	True Negative

High FN causes inconvenience (bad usability)

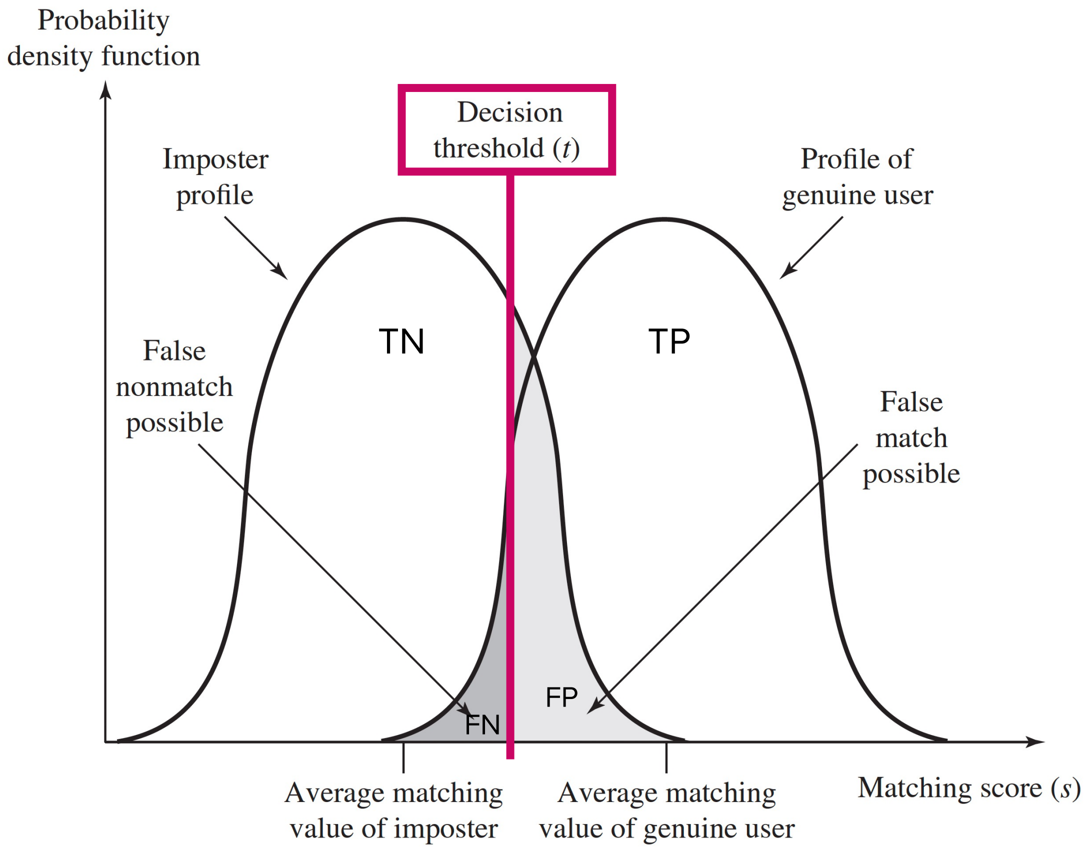
High FP means high exploitability (bad security)

→ Goals:

- Very high TP
- Very low FN
- Zero FP

Evaluating authentication method

- A dilemma: There is no clean separation between imposter and user profiles
  - Increase the threshold to get:
    - Increased security (FP⬇)
    - Decreased convenience (FN⬆)
  - Decrease the threshold to get:
    - Decreased security (FP⬆)
    - Increased convenience (FN⬇)



Profiles of a biometric characteristic of an imposter and an authorized user

Summary

- User authentication is hard
- Password-based auth is a long-lasting solution
- Strengthen passwords with password managers and MFA

Coming up next

- Authentication: To open the front door or not
  - Coarse-grained control for the entire system accessibility
- Access control: After opening the door to a user
  - Fine-grained control for system resources

Questions?