Lec 18: User Authentication (2)

CSED415: Computer Security

Spring 2024

Seulbae Kim

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Administrivia

- Lab 04 is out!
  - Two weeks (due May 5th)
  - About authentication and entropy

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

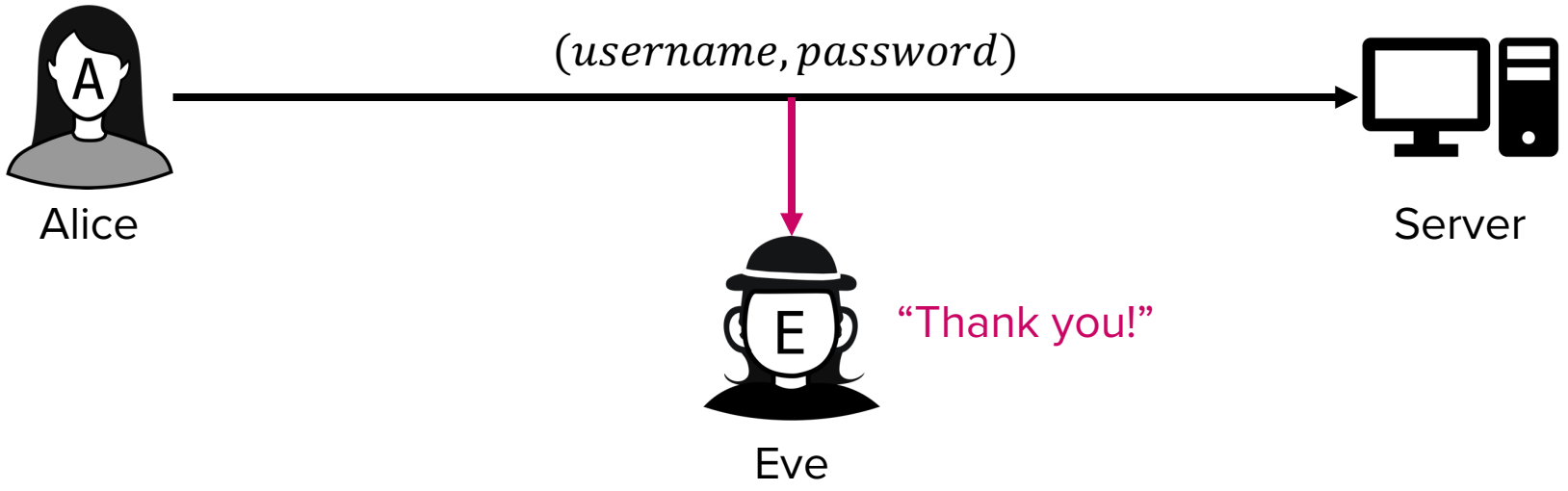
- Password-based
- Challenge-response
- Biometric
- Zero-knowledge
- Multi-factor

Today's topic!

Challenge-Response Authentication

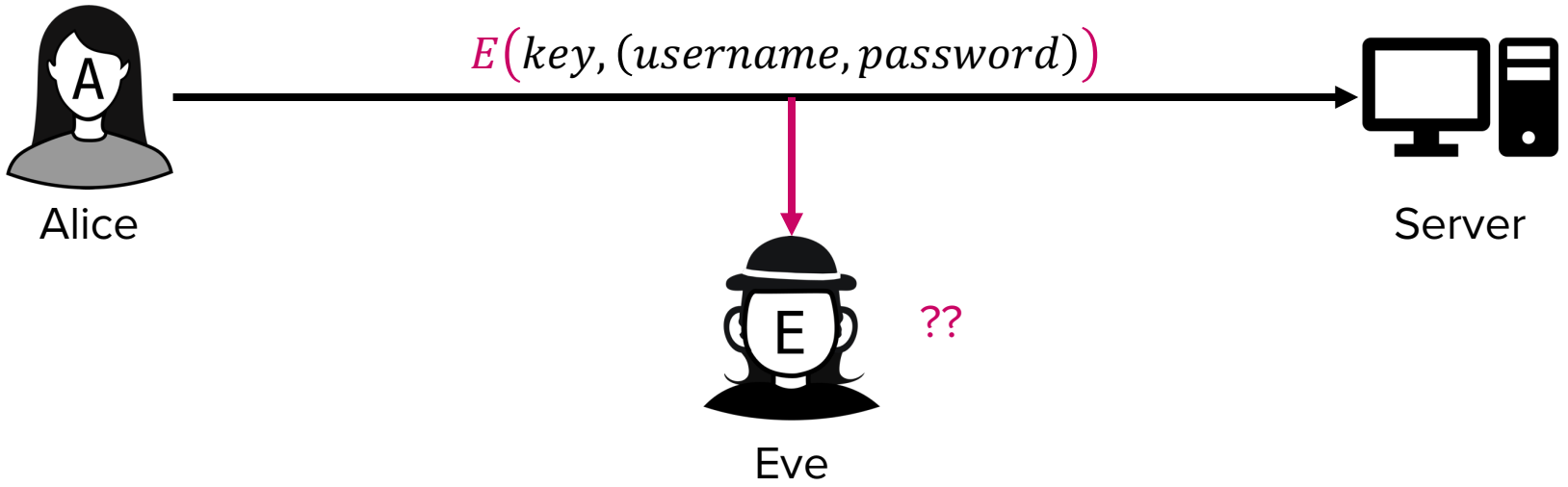
Transmitting a password

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



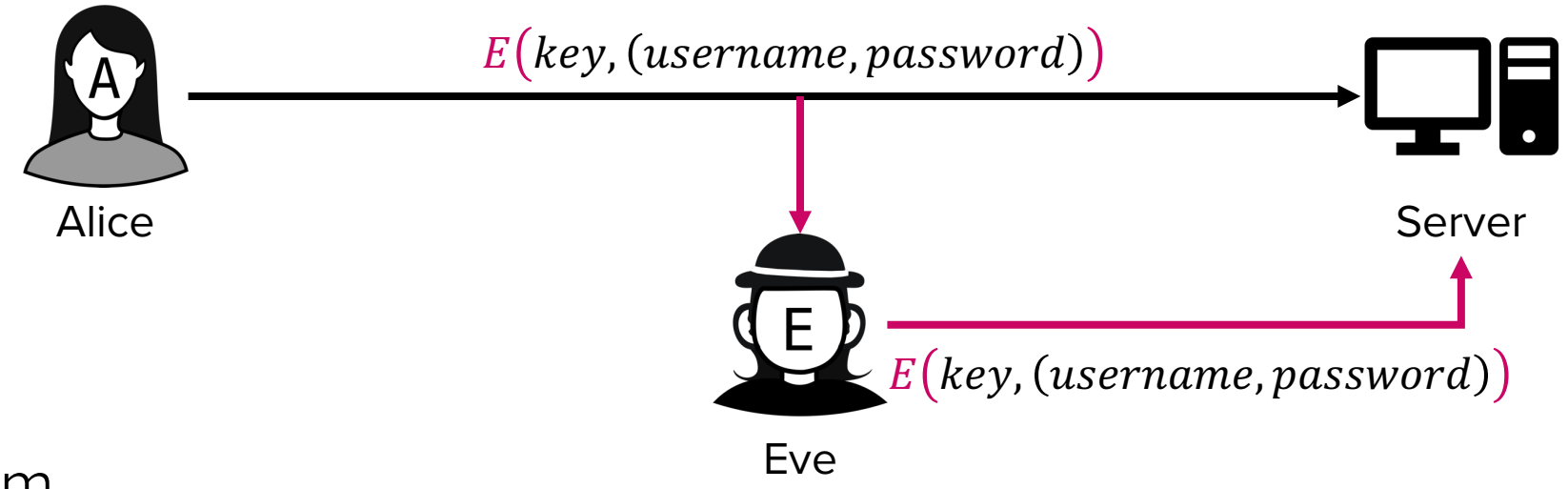
Transmitting a password

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



Transmitting a password

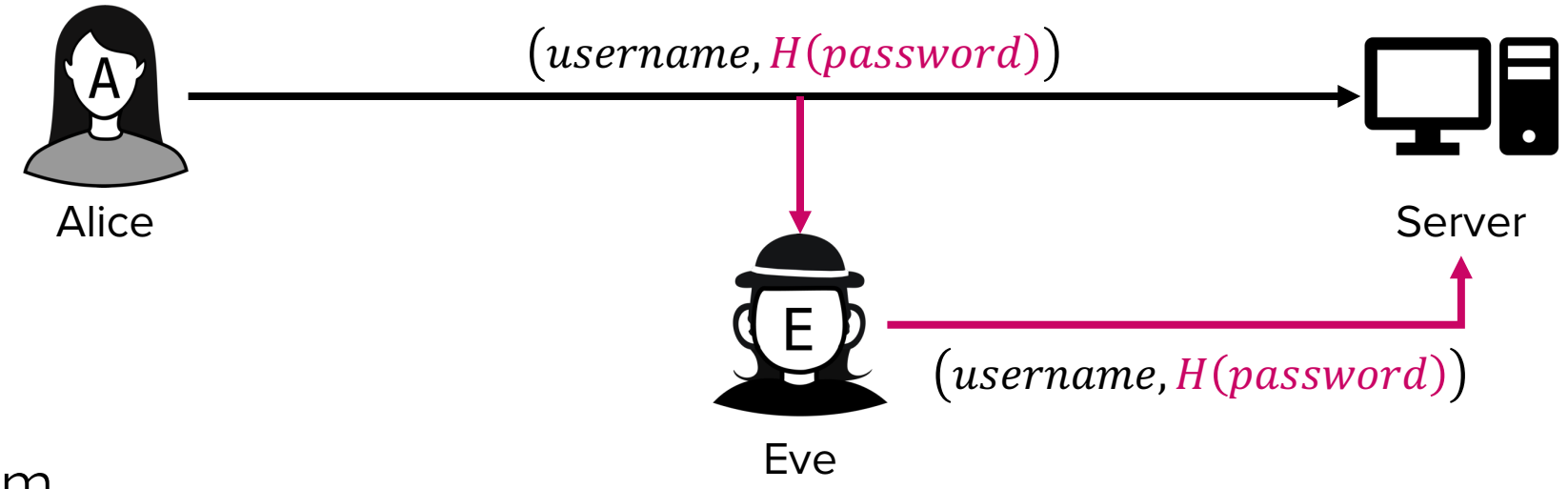
- How should a user transmit a password to a system?
  - Slightly better idea: Send the password over an encrypted channel



  - 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



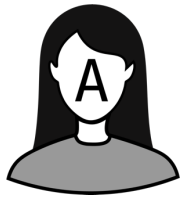
  - Problem
    - Hashing does not provide any extra security, since the hash can also be replayed

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

Hi, I'm Alice

Server

Challenge: C

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

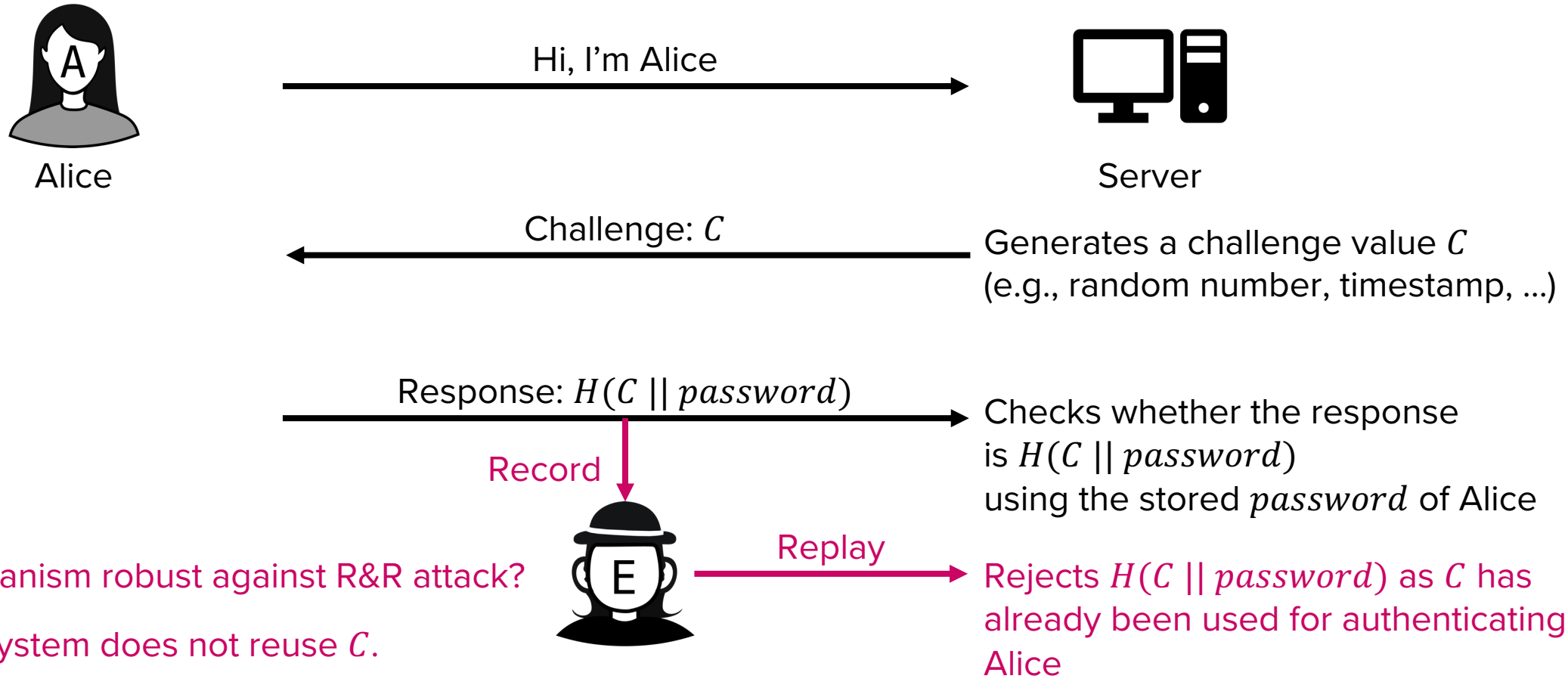
Response: $H(C \mid\mid password)$

Checks whether the response
is $H(C \mid\mid password)$
using the stored $password$ of Alice

Challenge-response authentication

POSTECH

- Idea

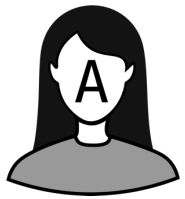


Q) Is the mechanism robust against R&R attack?

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

Challenge-response authentication

- Idea



Alice

Hi, I'm Alice

Server

Challenge: C

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

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

Checks whether the response
is $H(C \,||\, password)$
using the stored $password$ of Alice

Record

Replay

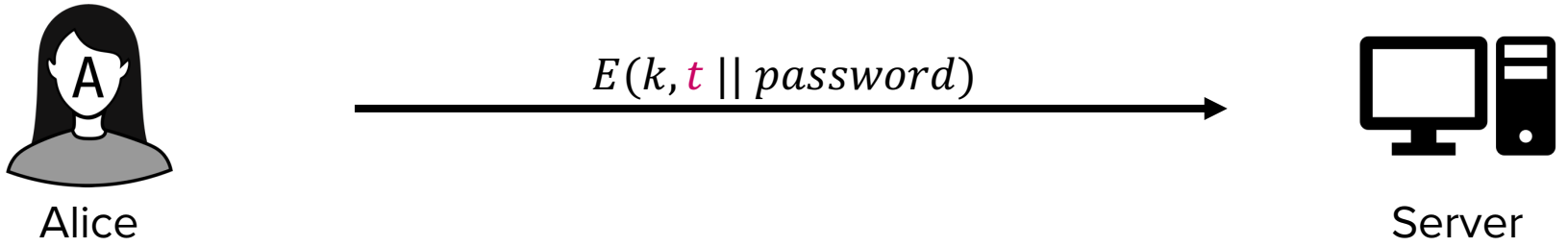
Rejects $H(C \,||\, password)$ as C has already been used for authenticating Alice

Q) Can the attacker recover Alice's pasword?

A) No! H is one-way.

Challenge-response in practice

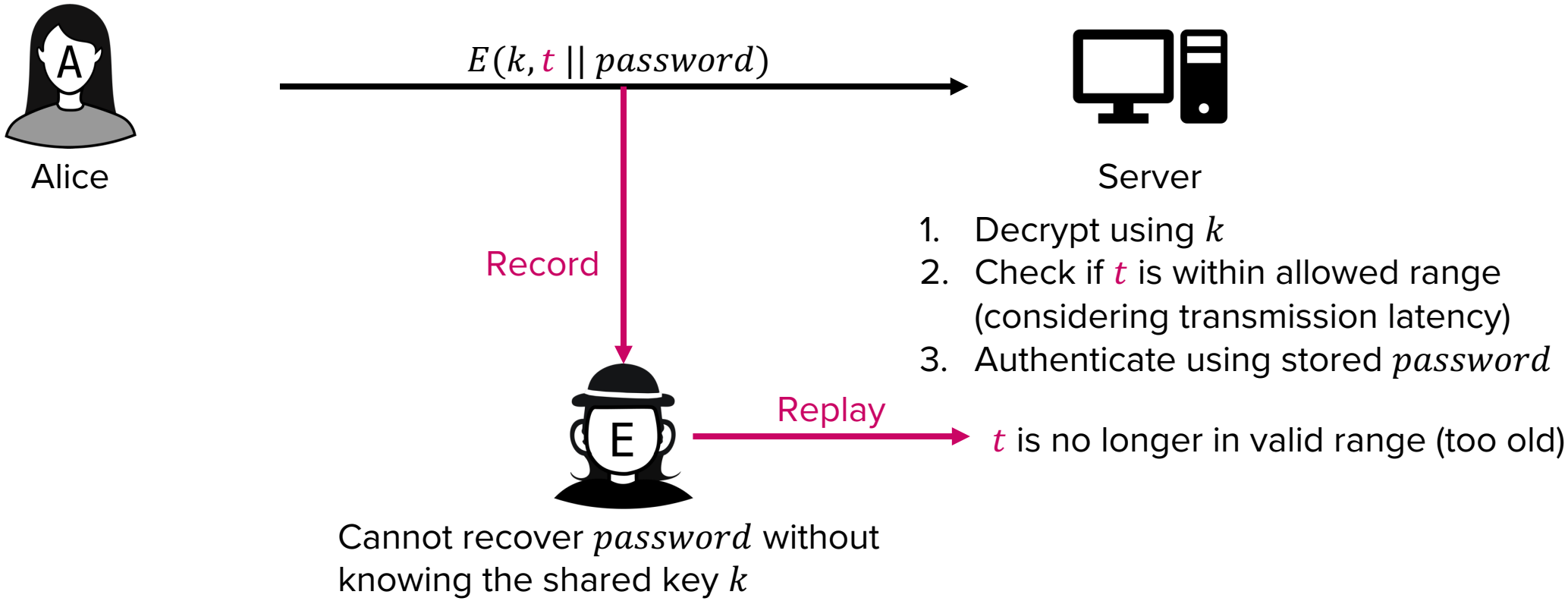
- Symmetric key-based implementation
  - Using shared key k and timestamp t (current time)



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

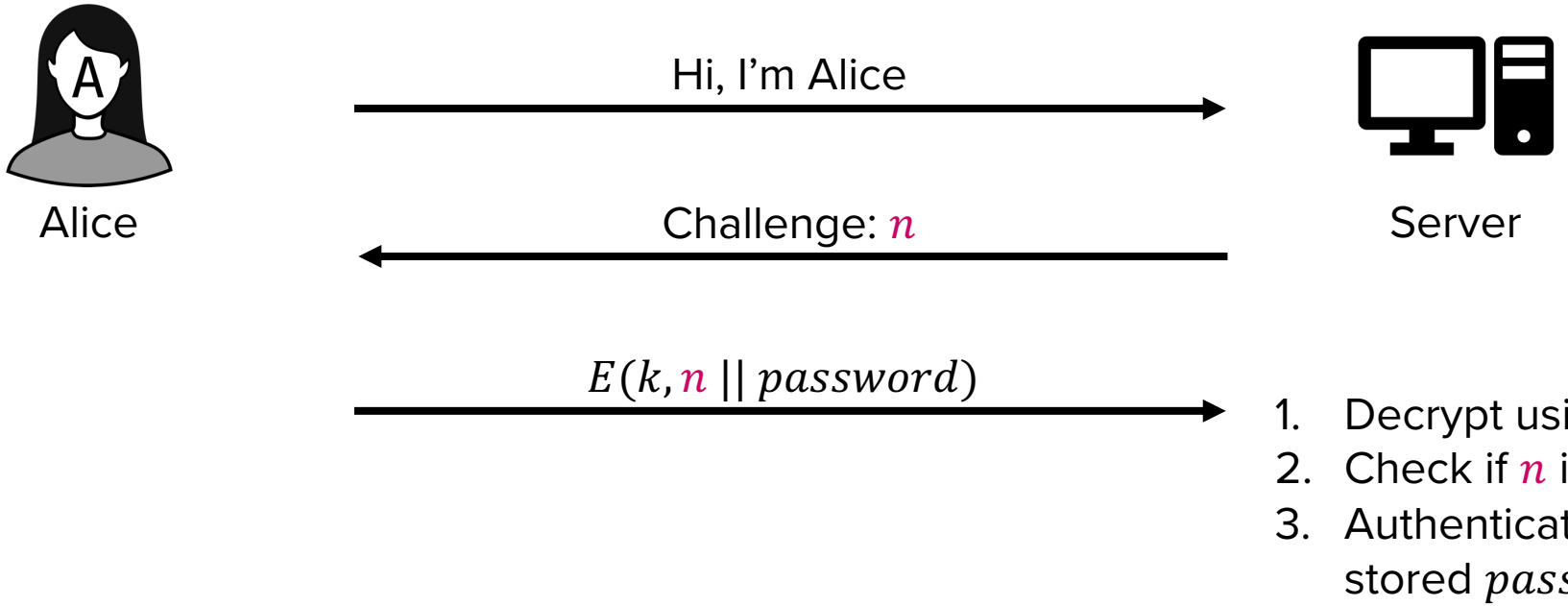
Challenge-response in practice

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



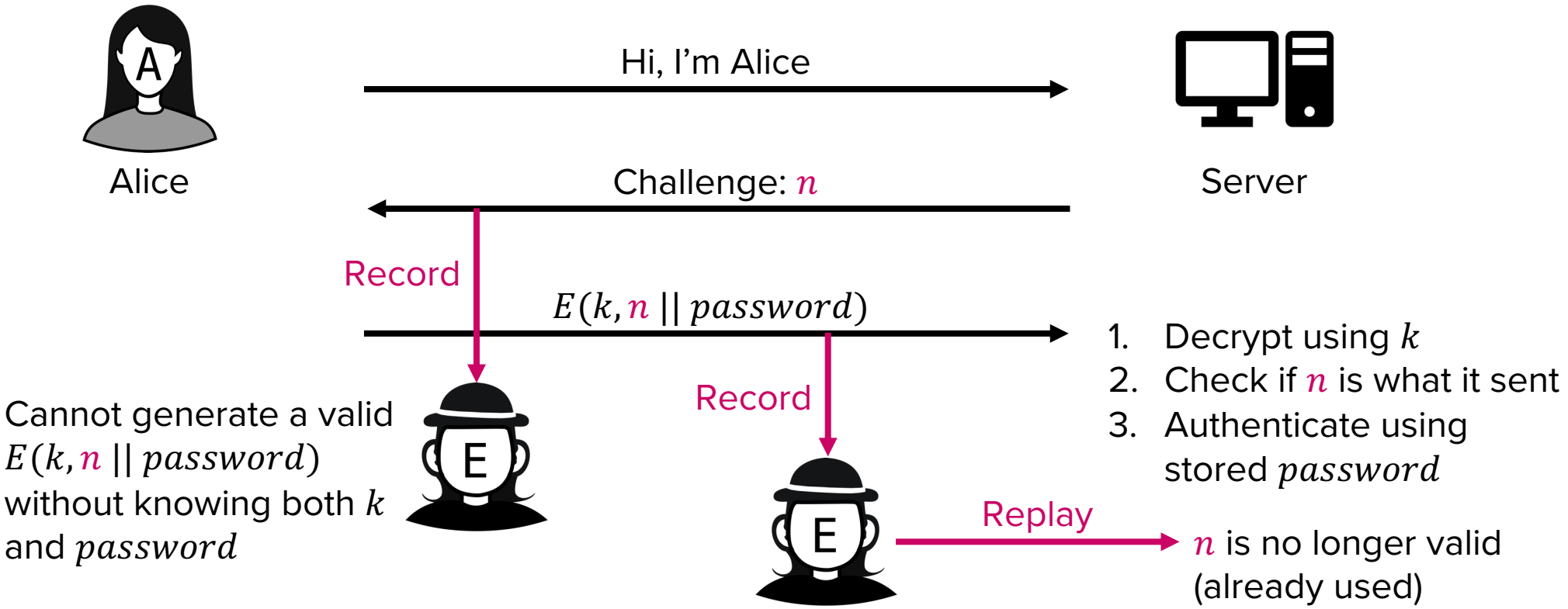
Challenge-response in practice

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



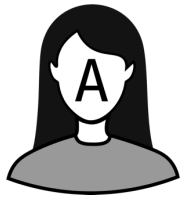
Challenge-response in practice

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



Challenge-response in practice

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



Alice

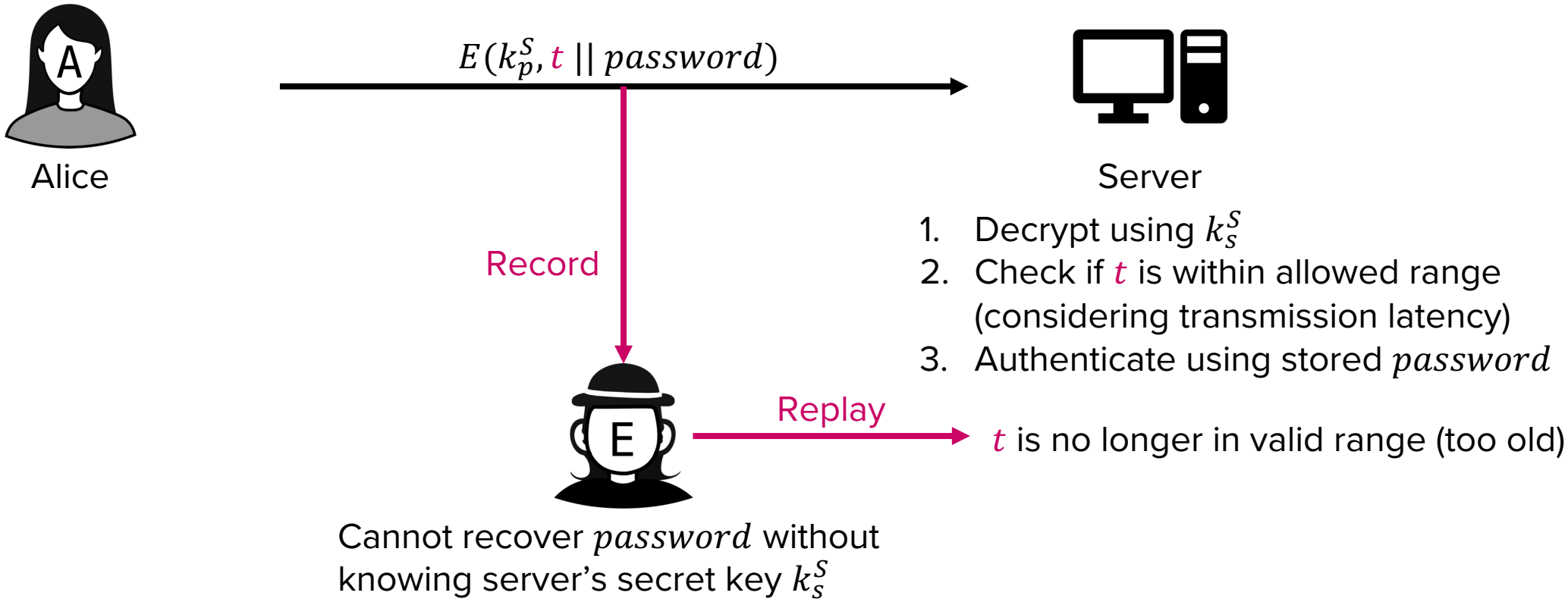
$E(k_p^S, t \,||\, password)$

Server

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

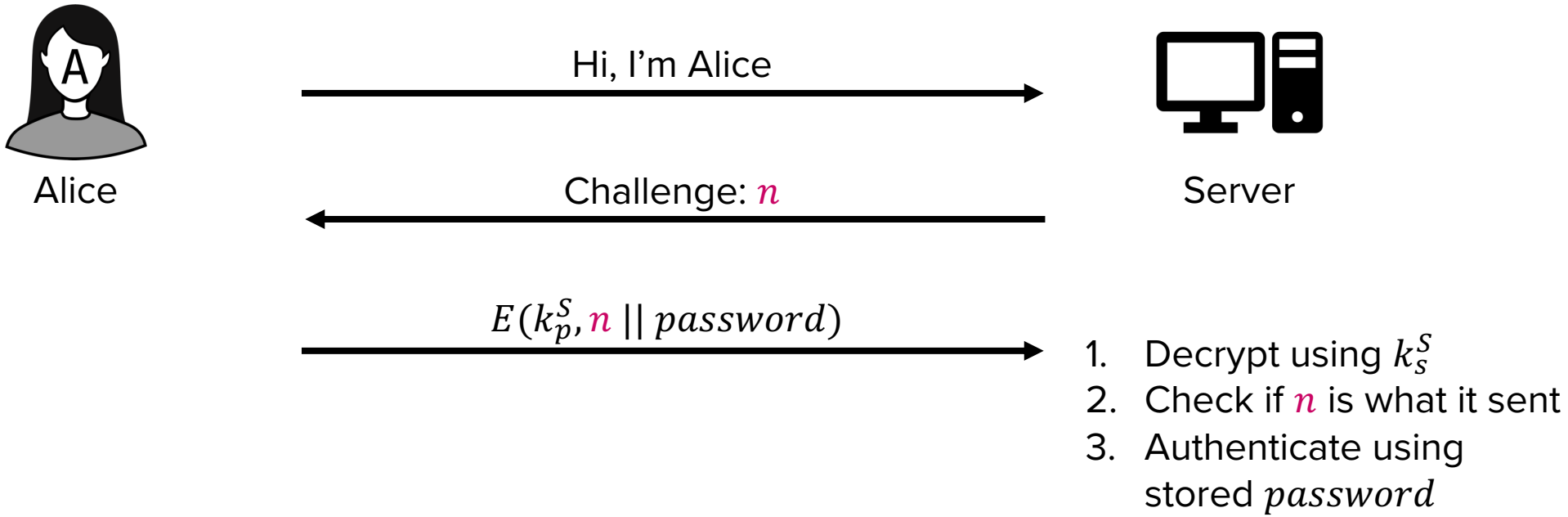
Challenge-response in practice

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



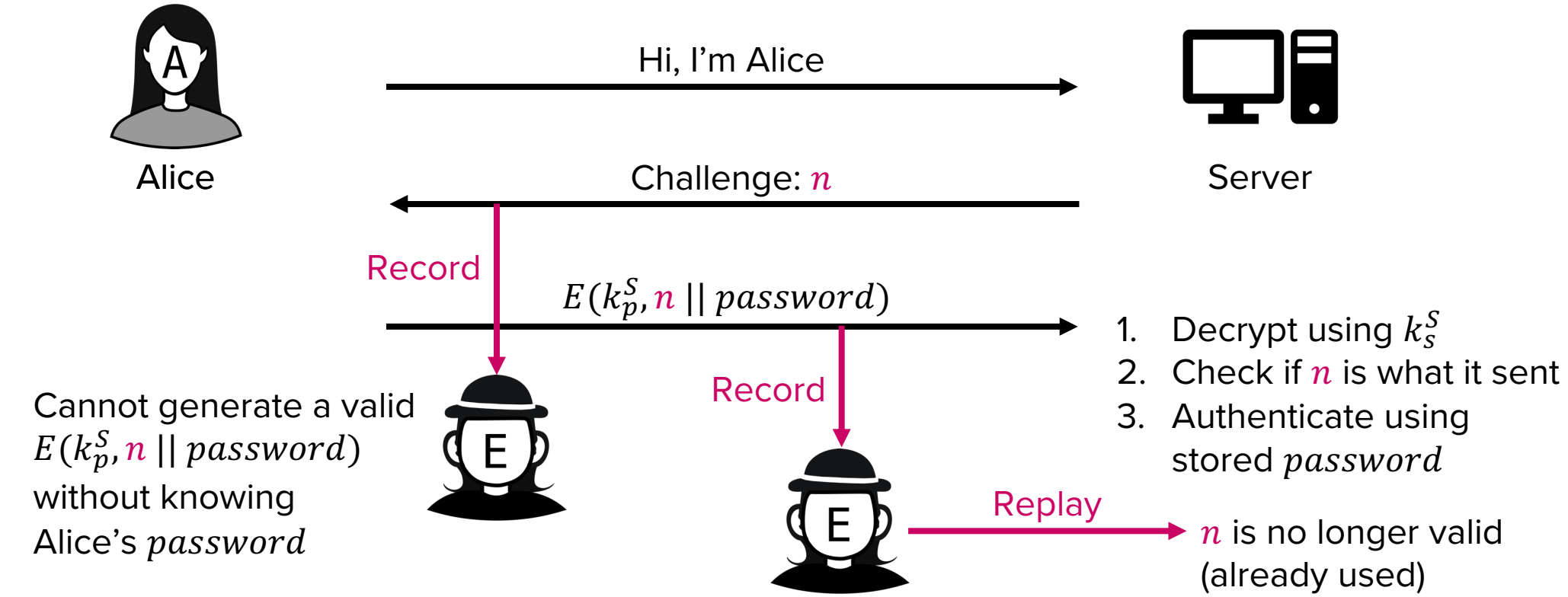
Challenge-response in practice

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



Challenge-response in practice

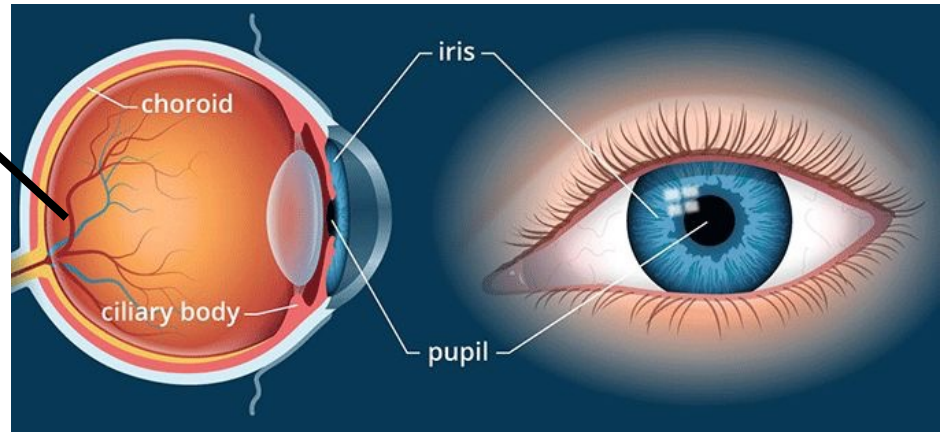
- Asymmetric key-based implementation
  - Using public key k_p^S, secret key k_s^S, and a nonce n



Biometric Authentication

Biometric authentication

- Using “something you are”
  - 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



Img: All About Vision

Biometric authentication

- Advantages of using what 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

Zero-knowledge Authentication

You identity matters

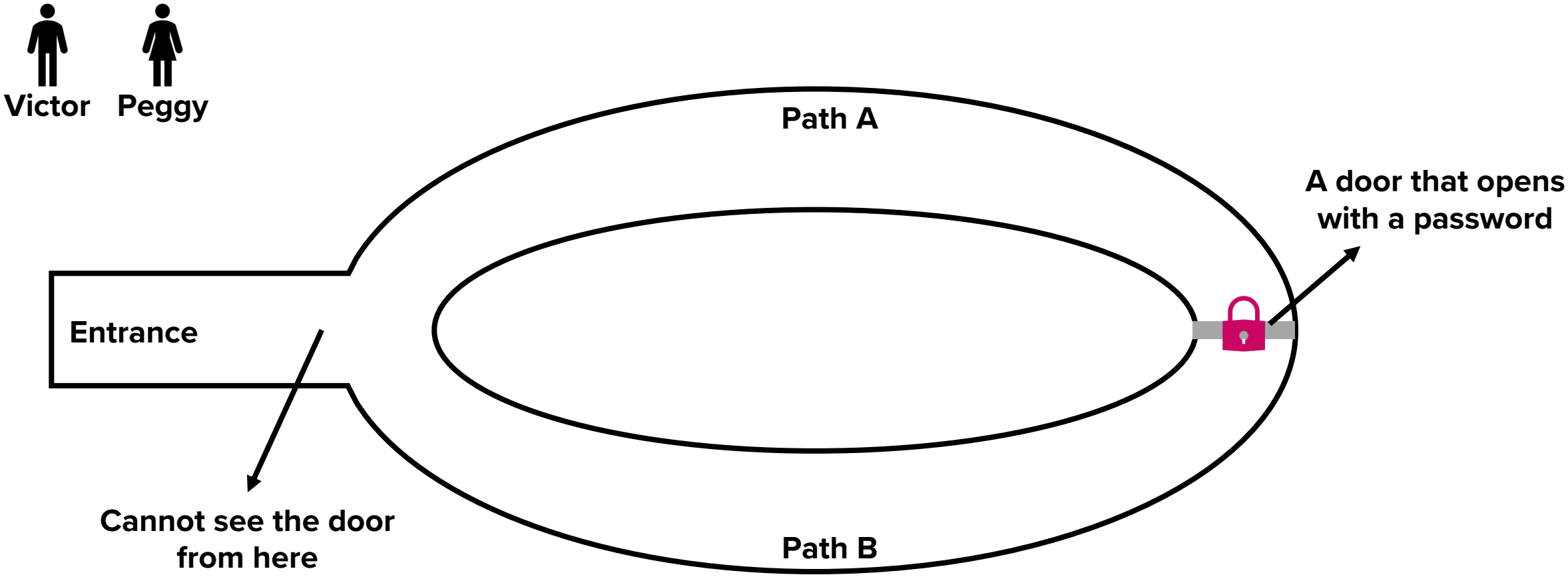
- Problem of existing authentication methods
  - Your identity is revealed during authentication
    - What you know (password / challenge-response)
    - What you have (token)
    - What you are (biometric information)

Zero-knowledge proofs (ZKP)

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

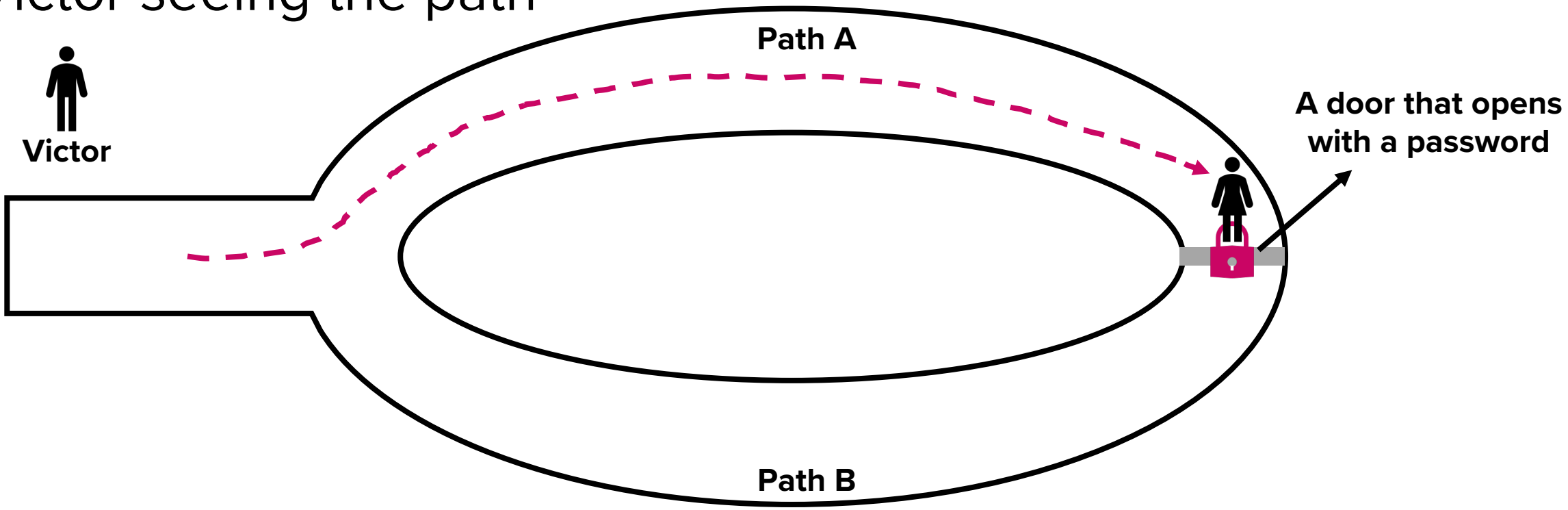
→ Can Peggy authenticate without revealing her identity?

The Ali Baba cave example



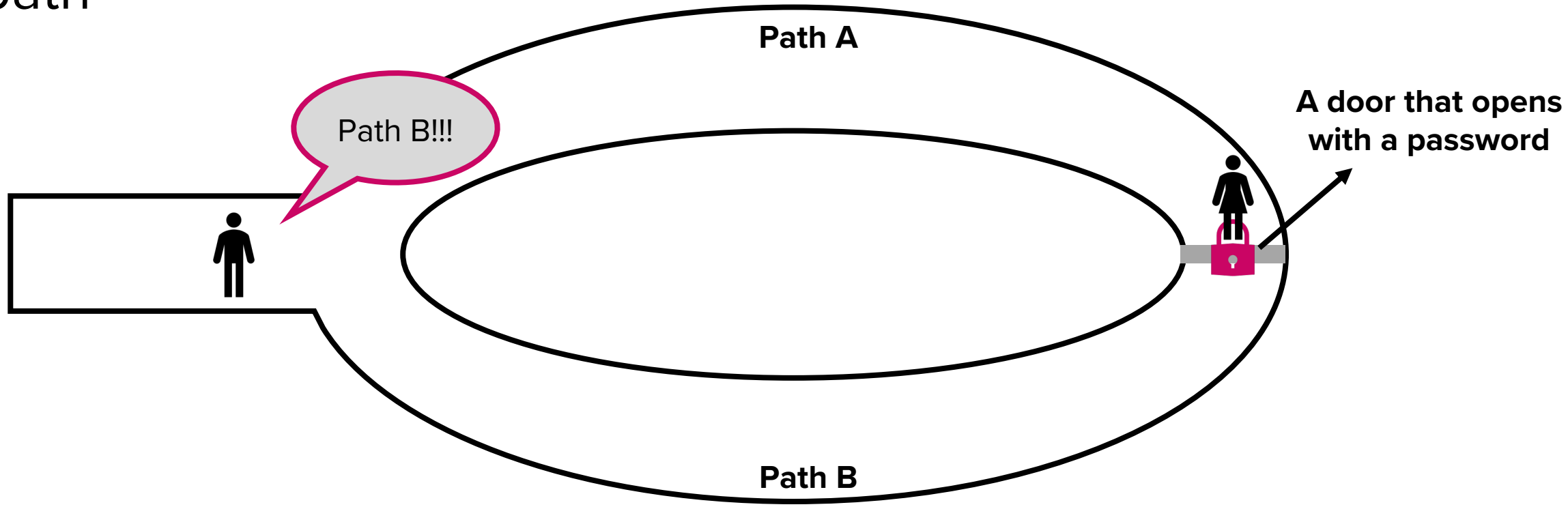
The Ali Baba cave example

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



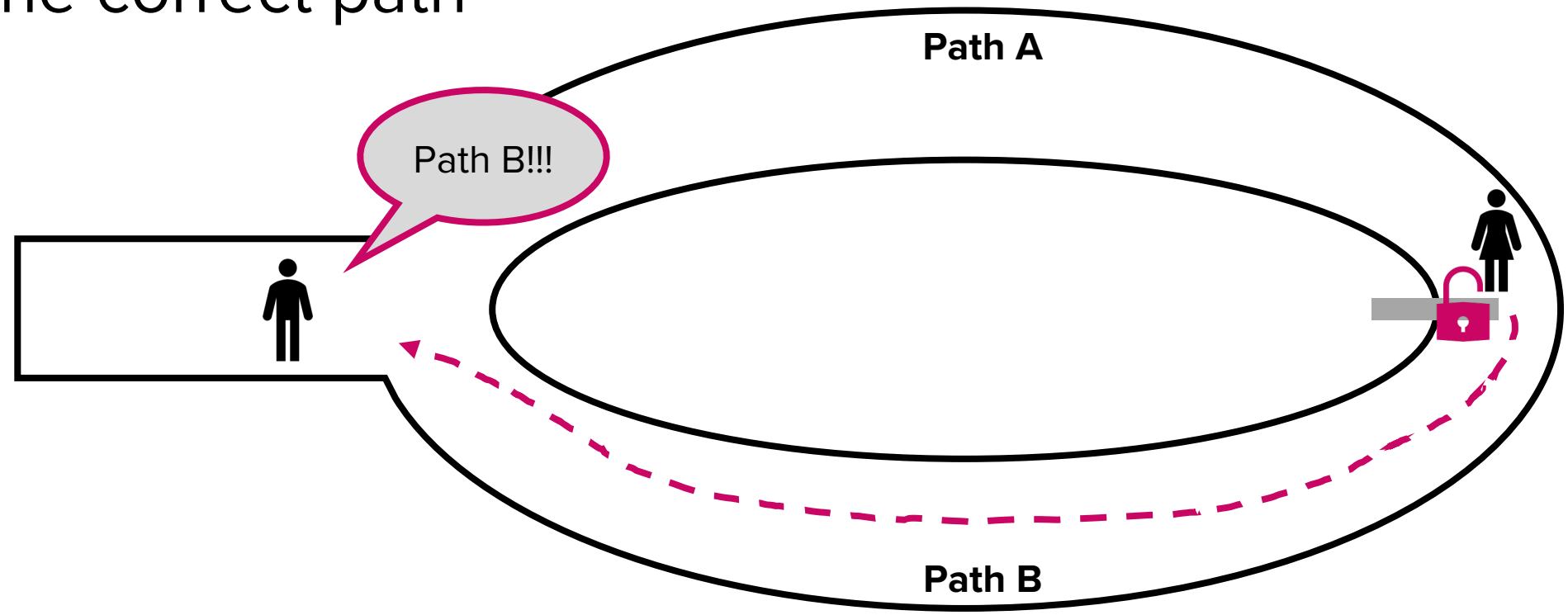
The Ali Baba cave example

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



The Ali Baba cave example

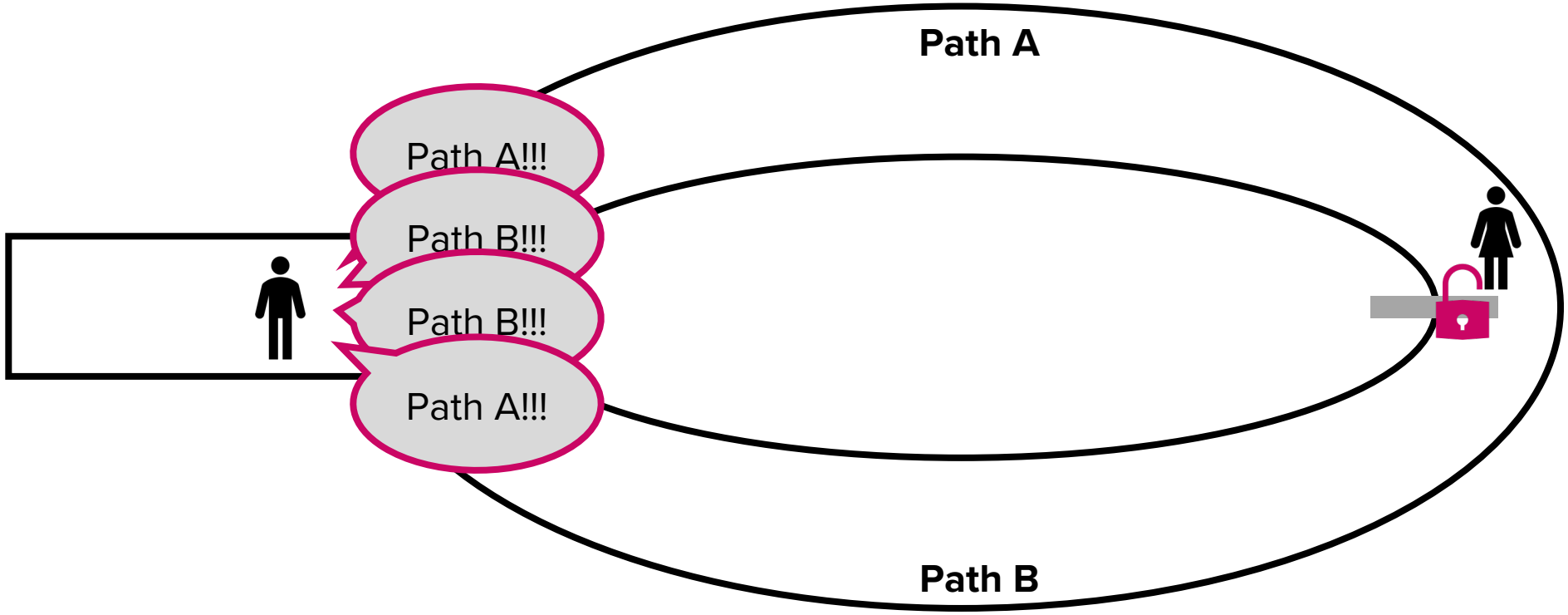
3. If Peggy knows the password, she can return to Victor using the correct path



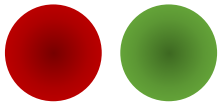
* If Peggy doesn't know the password, she still has a 50% chance to succeed

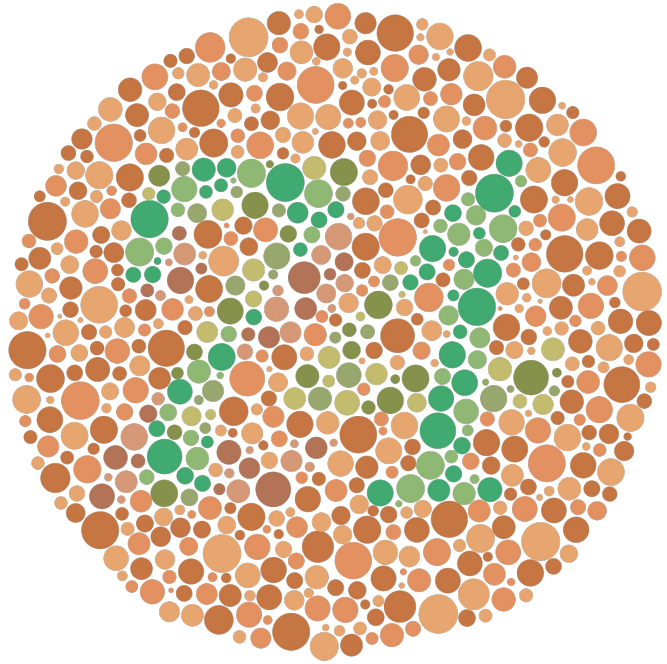
The Ali Baba cave example

4. Repeat multiple times until Victor is confident



Color-blind Victor example

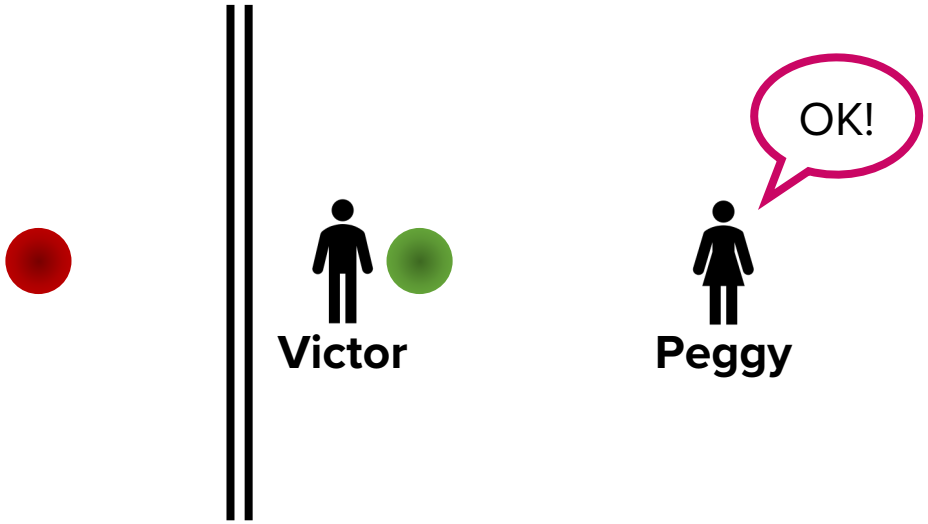
- Victor has a “red-green color blindness”
  - He cannot tell red from green
- Setting
  - Prepare two balls 
    - One red ball, one green ball
    - All properties (weight, size, …) are identical except for the color
  - Peggy should prove to Victor that the two balls have different colors



Ishihara Plate #9

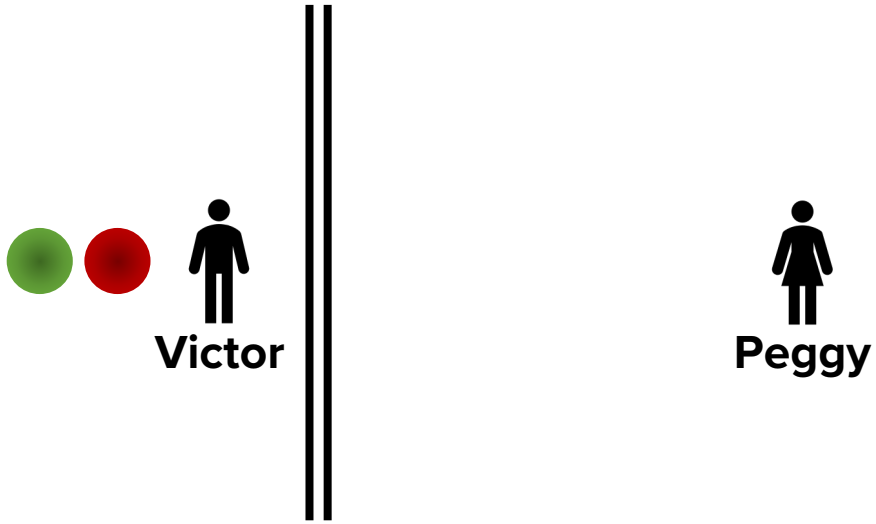
Color-blind Victor example

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



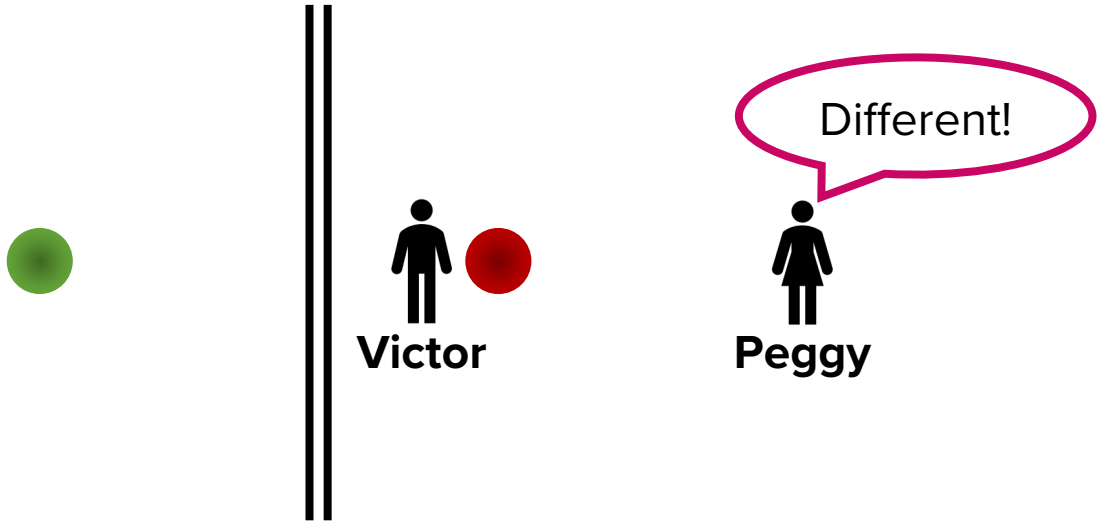
Color-blind Victor example

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



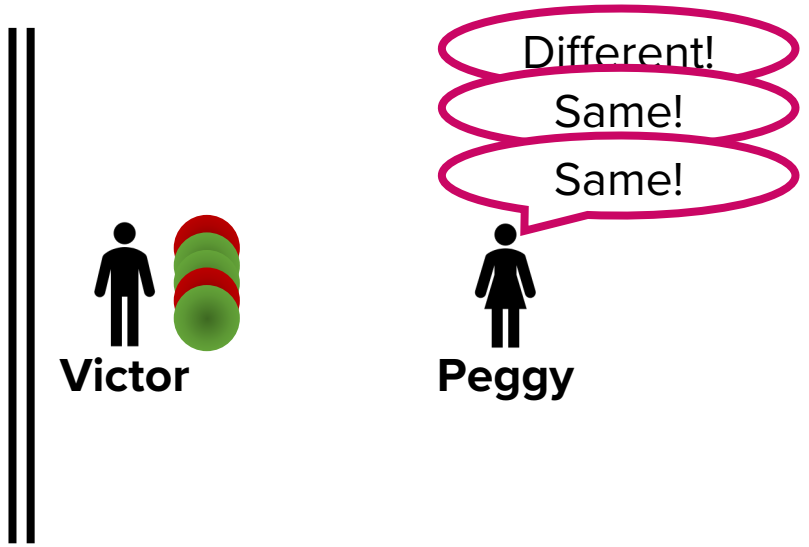
Color-blind Victor example

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

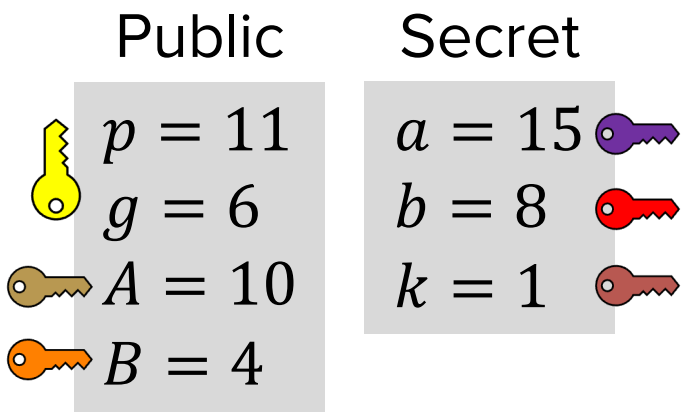
- Probability that Peggy is also color-blind but gets the answer right is 50%
  - Experiment repeated 10 times, probability that Peggy does not know the secret becomes $\frac{1}{2^{10}}$ (less than 0.1%)
- Victor learns that the two balls are distinguishable without learning the color of each ball

ZKP for user authentication

- Secure Remote Password (SRP) protocol
  - User authentication using ZKP
  - Server does not store client's password
  - Verify that the client knows the password w/o the password (ZKP!)

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 $k = A^b \ mod \ p = g^{ab} \ mod \ p$



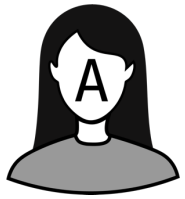
SRP protocol

POSTECH

Public: prime p, generator g

- Step 1: Registration

Secret:
Password $pass$
$x = H(pass \,||\, salt)$
(x is then discarded)



$Alice$, randomly chosen $salt$, verifier $v = g^x \; mod \; p$

Store: $(Alice, salt, v)$

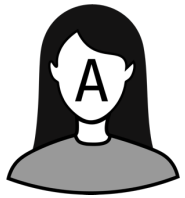
SRP protocol

- Step 2: Key sharing

Public: prime p, generator g

$salt$ A u, B

Secret:
Password $pass$



$username = Alice$

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

$salt$

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

Fetch Alice's $salt$

$A = g^a \ mod \ p$

Generate random secret a

Generate random param u

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

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

Generate random secret b

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

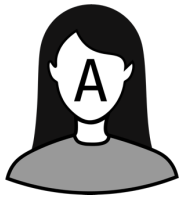
SRP protocol

- Step 3: Mutual authentication

Public: prime p, generator g

$salt$ A u, B

Secret:
Password $pass$



$username = Alice$

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

$salt$

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

Fetch Alice's $salt$

$A = g^a \ mod \ p$

Generate random secret a

Generate random param u

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

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

Generate random secret b

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

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

Compute and send M_1

If valid,
compute and send M_2

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

If valid, login success!

SRP protocol

- Strengths
  - 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
- Weaknesses
  - 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)
  - …
- (usually) 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 (e.g., scan QR code)
    - e.g., Google's Authenticator app
  - User device generates $TOTP = H(secret \,||\, cur\_time)$
    - Use coarse-grained time (e.g., cur_time is updated every 30 seconds)
  - Server checks that code corresponds to current time's $TOTP$
  - Advantage:
    - Do not need phone network, do not need to trust phone carriers
  - Disadvantage:
    - Needs app installation and setup
    - Server compromise breaks 2FA! Need to re-register all secrets

Evaluating Authentication Method

Evaluating authentication method

- Metric for security: Work factor
  - How much work does an attacker have to do to crack a password?
  - Work factor is proportional to the entropy

  - Easy for an authentic user (recall: psychological acceptability)
  - Hard for an adversary

Evaluating authentication method

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

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

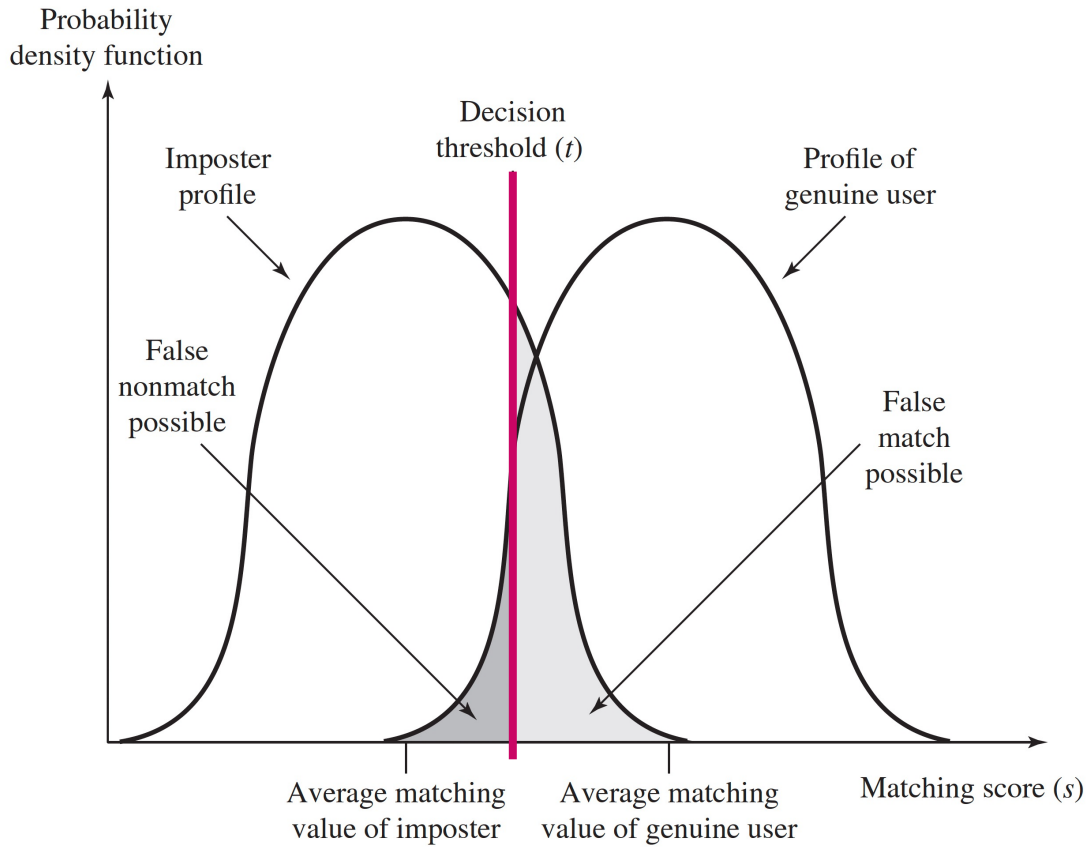
High FP means high exploitability
(bad security)

High FN causes inconvenience
(bad usability)

→ 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 threshold to get:
    - Increased security (FP⬇)
    - Decreased convenience (FN⬆)
  - Decrease 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?