Authentication I

CSE 565: Fall 2024
Computer Security

Xiangyu Guo (xiangyug@buffalo.edu)

University at Buffalo

Announcement

- Please sign-up at course Piazza.
- Reminder of Quiz 0 (**Due 09/19**).
- You must obtain full score of the Quiz.

Review of Last Lecture

- Key Exchange: how we establish the shared secret key
  - Diffie-Hellman Key Ex protocol
- Public Key Cryptography
  - Trapdoor function
  - RSA
  - ElGamal: PKC from Diffie-Hellman
  - Digital Signature & Certificates

Today's Topic

- Common techniques for authenticating users, locally and remotely
  - **Password-based authentication**
  - Token-based authentication
  - Biometric-based authentication
- Security challenges associated with different authentication methods
- Mitigations designed to address some of the above security challenges

Overview

Authentication

- Previously: Message Authentication
  - Message Authentication Code (Keyed Hash) to confirm that the message came from the stated sender (its authenticity) and has not been changed in transit (its *integrity*).
- Today: **User/entity Authentication**
  - Allow a user/computer to prove his/her/its identity to another entity (e.g., a system, a device, a remote server).

Entity Authentication

- How do we authenticate a human user to a system?
  - System is often remote server
  - Authenticate: ascertain who is interacting with the system
    - Necessary to apply appropriate security policy.
    - Only the intended subject should be able to authenticate to the system as that subject.

Entity Authentication

- How do we authenticate a human user to a machine?
  - Provide *identity* and *proof* of identity
  - Identity examples:
    - Name, username, student ID, others?
- User registration is required prior to an authentication protocol

Entity Authentication

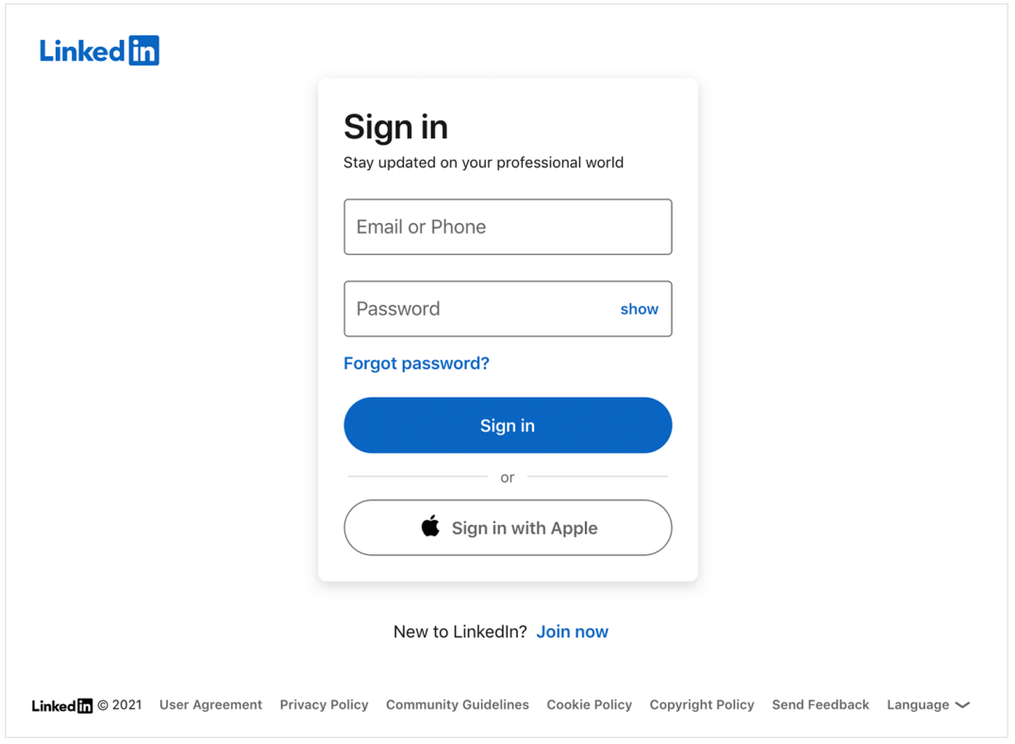
- How can Alice prove that she's really Alice?
  - Three types of authentication factors
    - Password: **Something you know.**
    - Token: **Something you have.**
    - Biometrics: **Something you are.**
- Each factor can be used independently, or combined for multi-factor authentication.
  - Typically two-factor

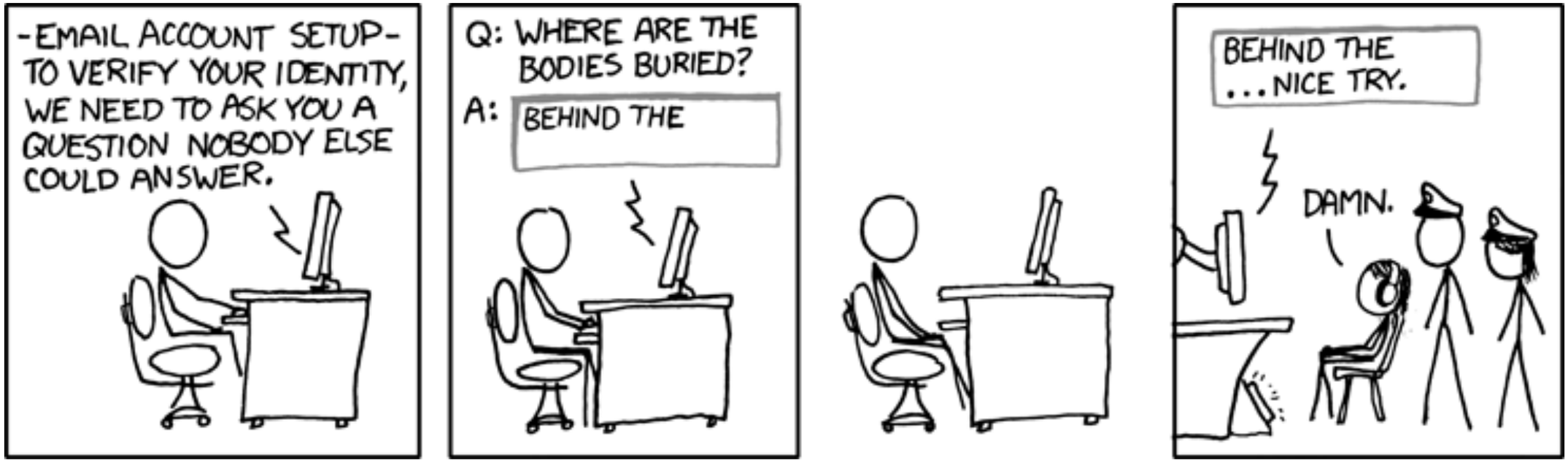
Password-based Authentication: Basics

“Something you know”

- A **secret** that only the real Alice should know
  - A secret *passcode*.
    - Examples: PIN, password
      - PIN: Personal Identification Number (misnomer. Usually used for authentication, not identification.)
  - A secret about Alice
    - Examples: mother’s maiden name, first pet, mortgage payment
- Technically, only proves knowledge of secret, not that it’s really Alice
  - Secrets leak, can be shared, guessed.

“Something you know”





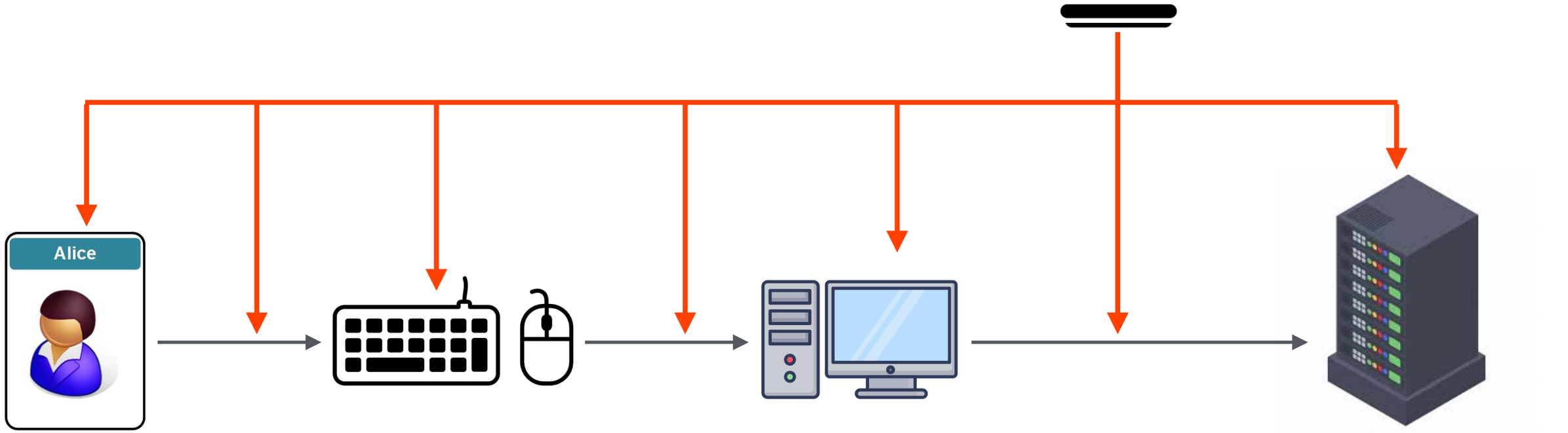
Password-Based Authentication

- How does Alice prove she knows the password?
  - Simplest: Alice provides the password to the system.
- Problems?
  - **Passive** adversary may observe password in transit
    - Need secure channel to protect confidentiality
  - **Active** adversary may impersonate the system
    - Alice needs a way of *authenticating the system*

Attacking Passwords

- Alice uses a keyboard to type her password into client software (e.g. browser) that sends it on to the remote system for authentication.
- Which points can Eve attack?

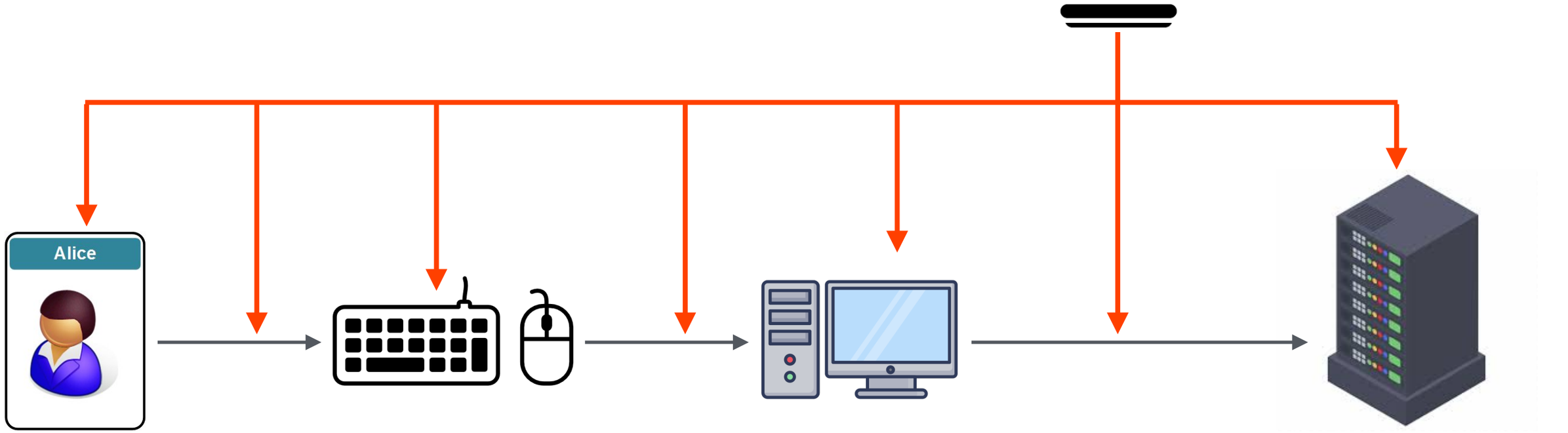
Eve



Attacking Passwords

- Get it from Alice
- Intercept it
- Get it from the system

Eve



Alice

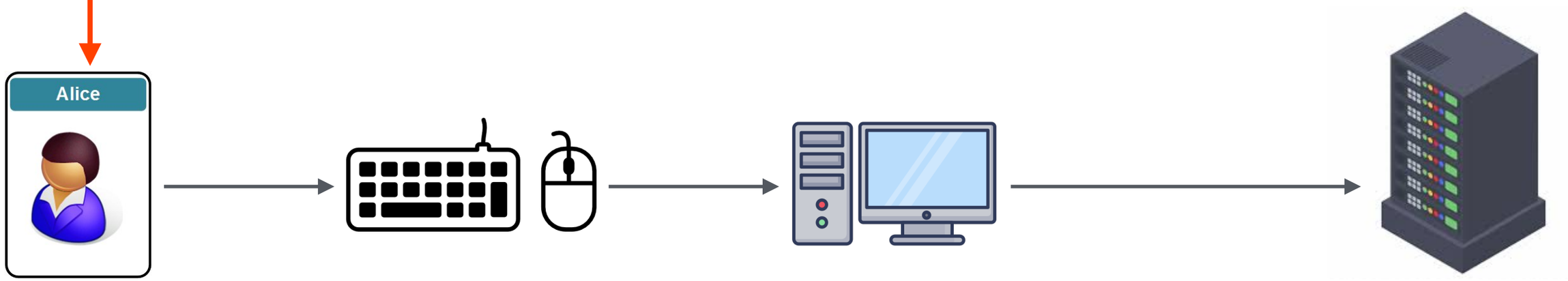
Alice's computer

Remote system

Attacking Passwords

- **Get it from Alice**
- Intercept it
- Get it from the system

Eve



Alice

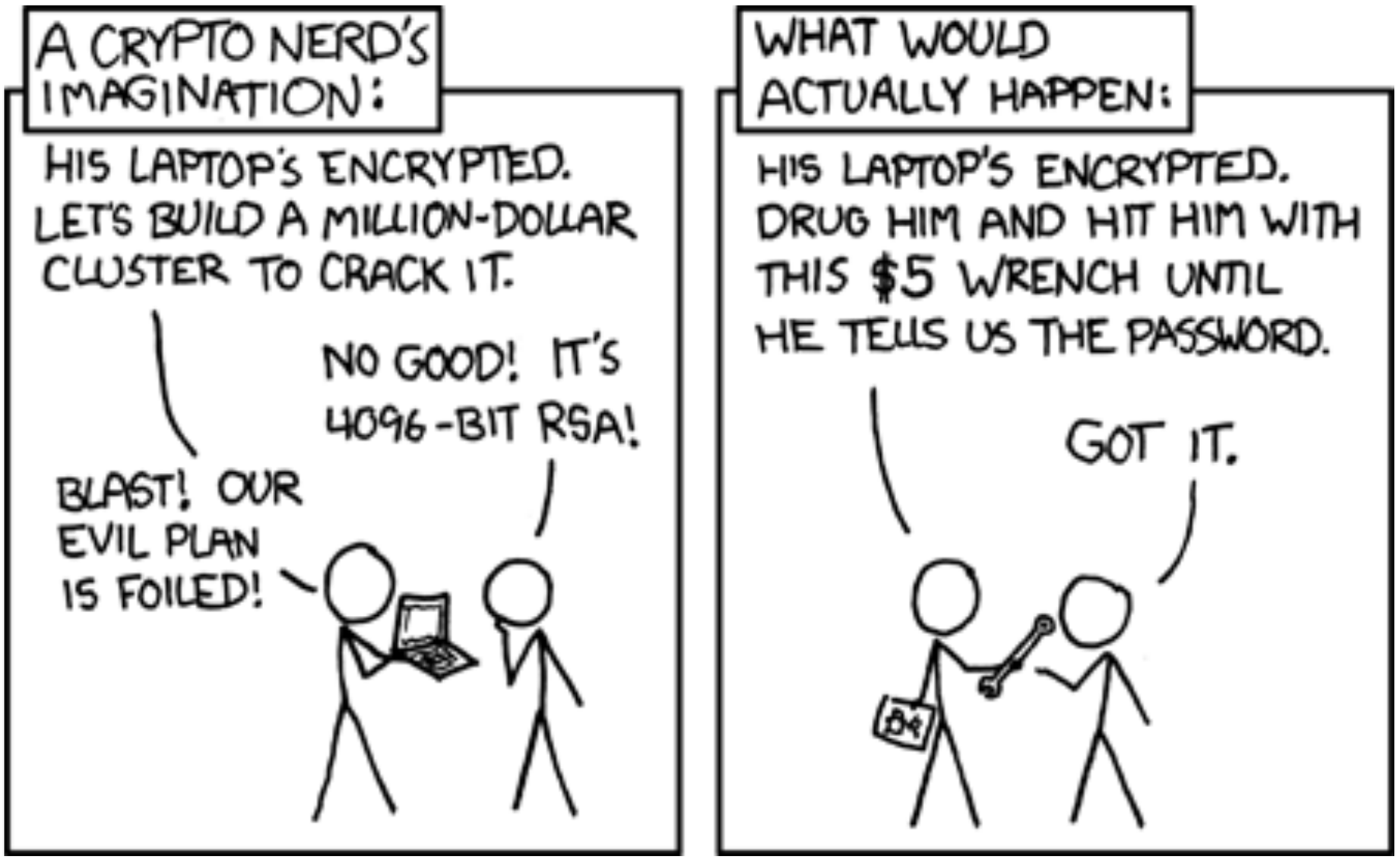
Alice's computer

Remote system

Attacking Passwords

- Is Alice invested in keeping it a secret?
  - Debit card PIN number?
  - Personal email password?
  - Netflix password?
  - Corporate network password?
- Is it written down somewhere?
  - Good against remote attackers
  - Not good against targeted local attacks (co-workers, family, abusers)
  - Know your threat model!
- Can it be guessed based on available knowledge about Alice?

Attacking Passwords



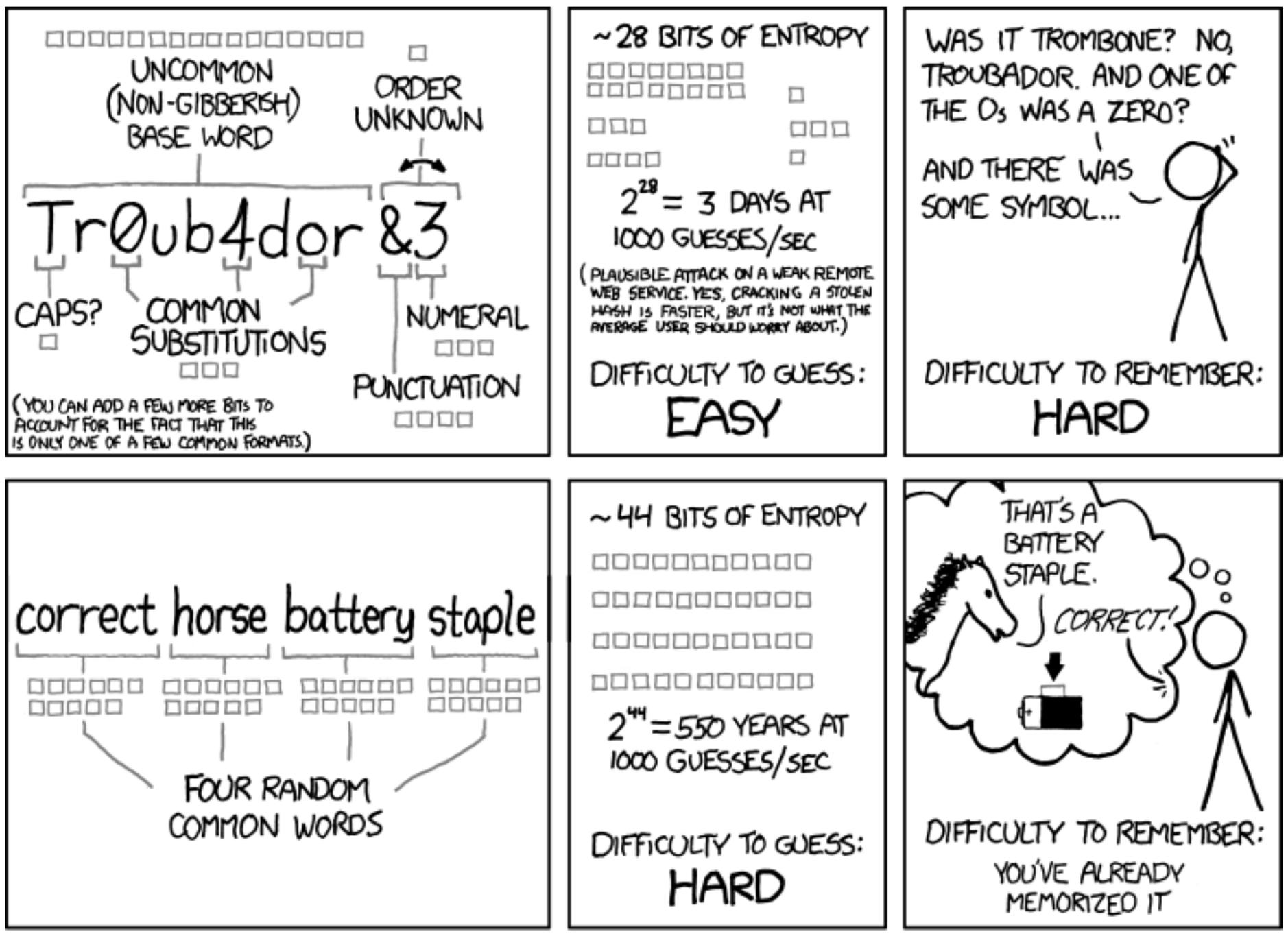
Strong passwords

- Challenge: come up with passwords that are hard to guess, but easy to remember.
- Common password rules:
  - Composition: Letters and numbers, mixed case, symbols, banned dictionary
  - Length
  - Lifetime
- Unintended consequences
  - Required letters/symbols $\Longrightarrow$ common substitution; Predictable pattern.
  - Monthly change requirement $\Longrightarrow$ Incremental; Recycling; Weaker pwd.

Top 20 most common passwords according to NordPass[3]

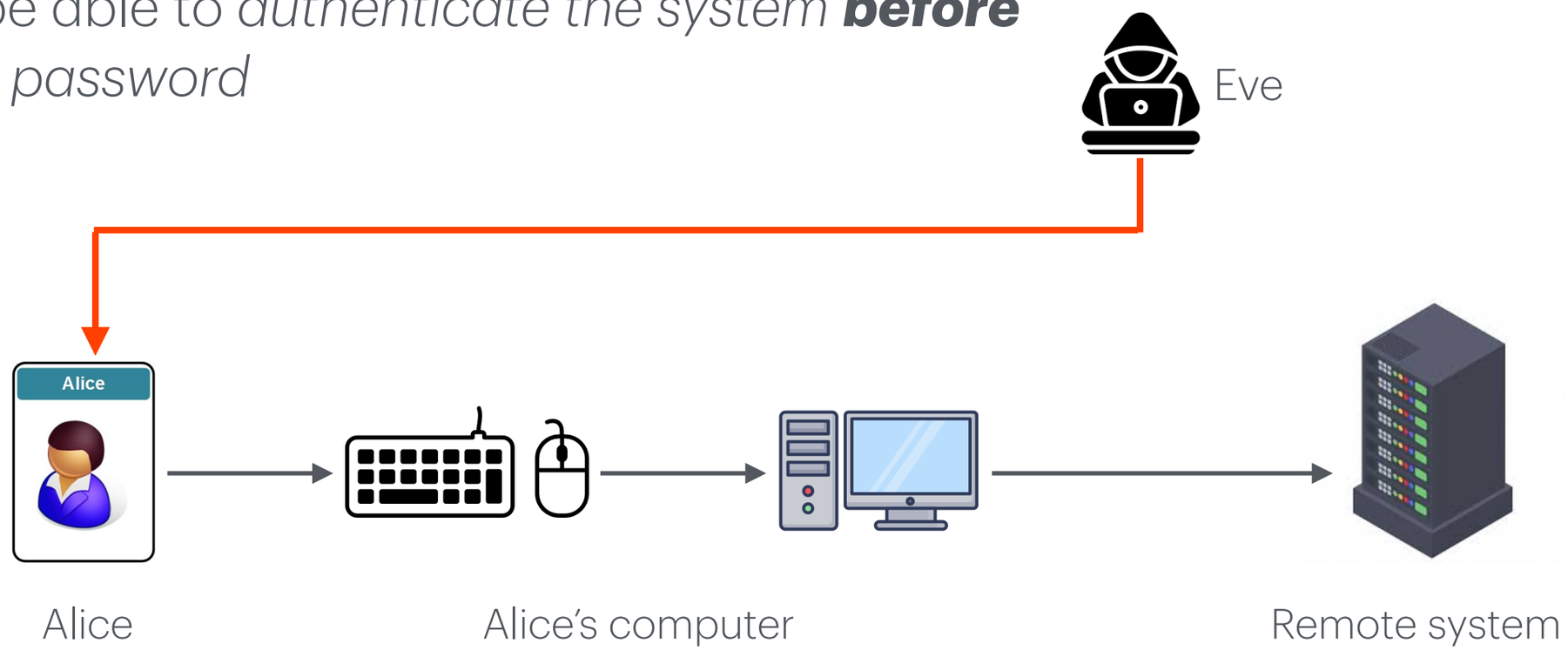
Rank	2021
1	123456
2	123456789
3	12345
4	qwerty
5	password
6	12345678
7	111111
8	123123
9	1234567890
10	1234567
11	qwerty123
12	000000
13	1q2w3e
14	aa12345678
15	abc123
16	password1
17	1234
18	qwertyuiop
19	123321
20	password123

Strong passwords



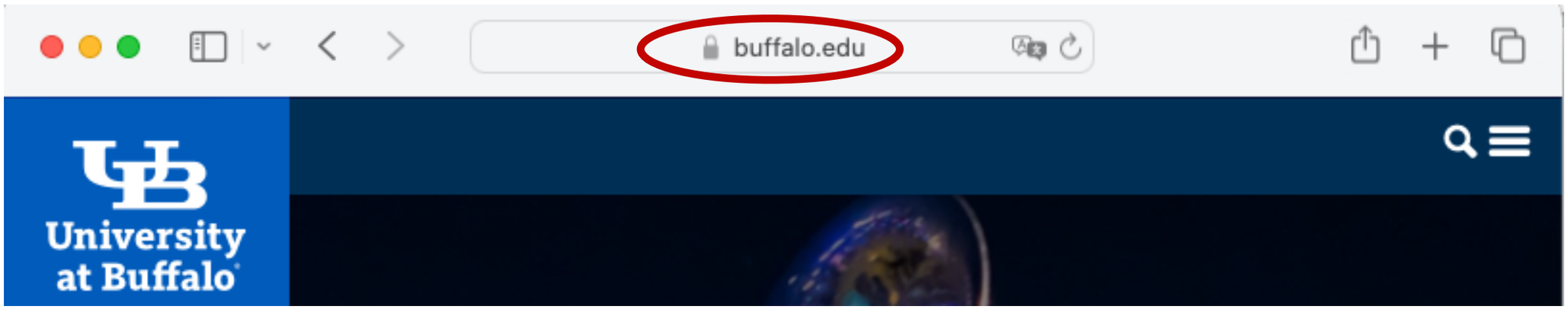
Attacking Passwords

- Can Eve trick Alice into revealing her password?
- How does Alice know she is logging into the real system?
  - **Phishing**!
  - Tricking Alice into revealing her password by Impersonating the system she is trying to access
- Alice has to be able to *authenticate the system* ***before*** *providing her password*



Phishing

- How can Alice authenticate the system?
- HTTPS certificates validate the domain name in the URL.



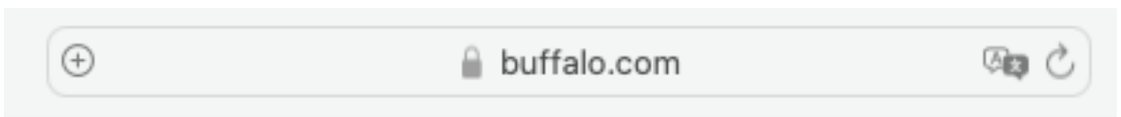
- What does it really tell you?
- That you are communicating with a server owned by UB?
- **No**. Only that you are communicating to www.buffalo.edu and your connection is secure (confidentiality and integrity are protected) against passive and active attackers on the link.

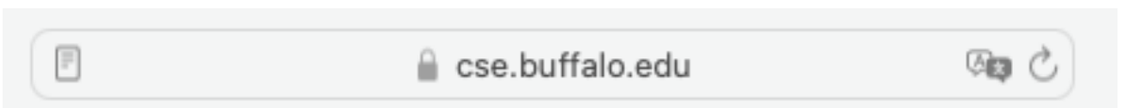
Phishing

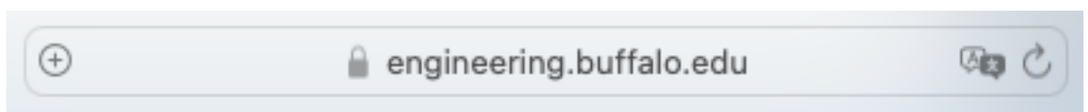
- How do you know www.buffalo.edu is a legitimate UB web site?
- What about:
  - www.cse.buffalo.edu
  - www.buffalo.cse.edu
  - www.cse-buffalo.edu
  - www.buffalo.net
  - .....

Phishing

- How do you know www.buffalo.edu is a legitimate UB web site?
- A user is *expected* to know which domains are associated with the entity they are trying to interact with.
- And how to properly parse the URL
- Some browsers now highlight the domain portion





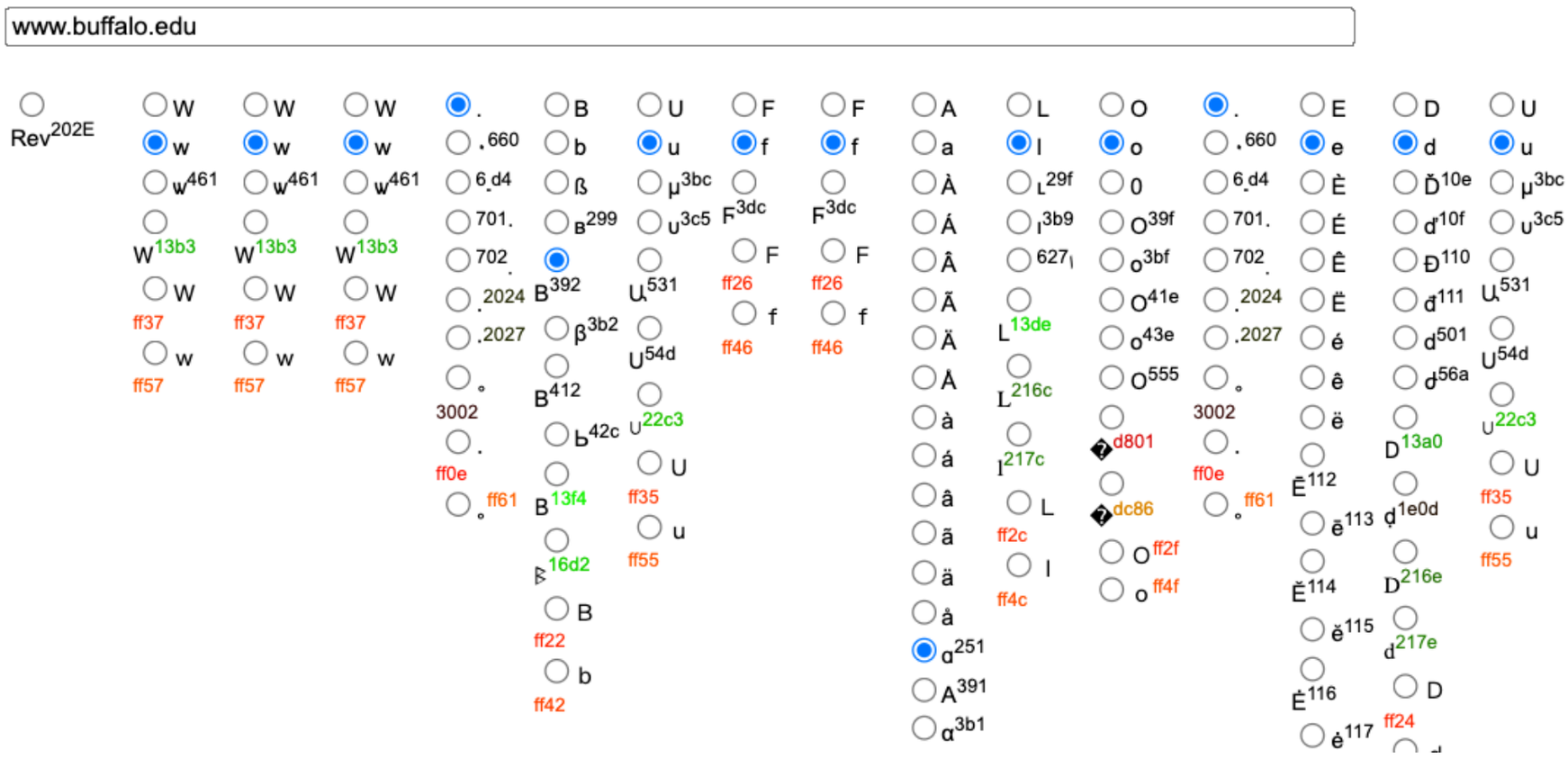


Phishing

- What if the user knows which domain to visit?
- **Homoglyphs**: symbols that appear identical or very similar
- Attacker: Register domain names that look just like the victim domain, but using a different character set.

Phishing

- https://www.irongeek.com/homoglyph-attack-generator.php



Unicode URL to give out: www.Buffalo.edu

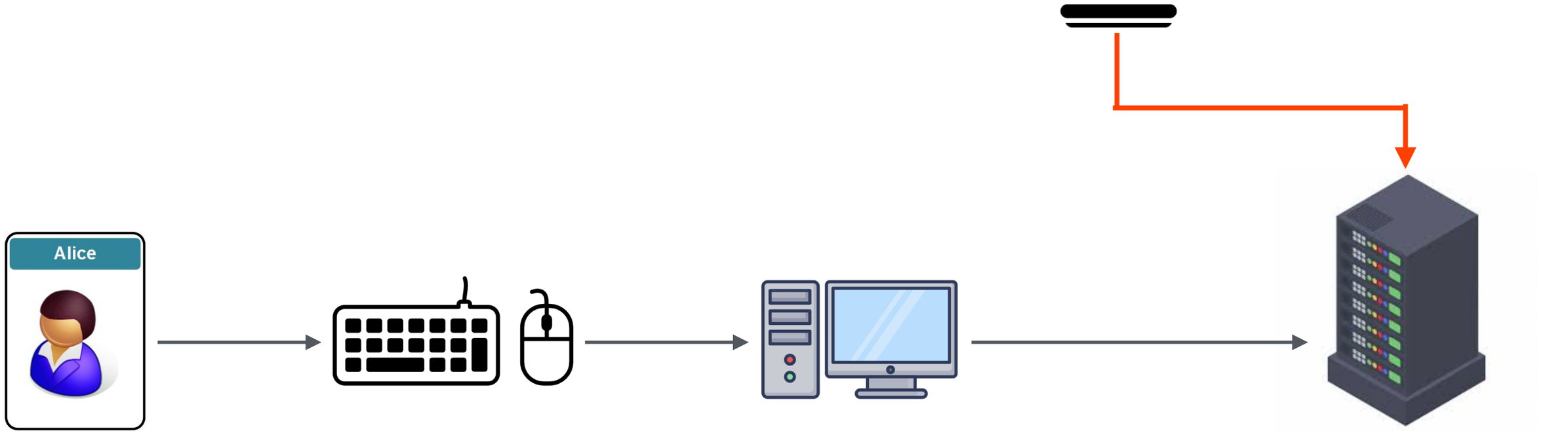
Encoded label to set up in DNS: www.xn--ufflo-2jc75t.edu

Password-based Authentication: Defend against Direct Attack

Attacking Passwords

- Get it from Alice
- Intercept it
- **Get it from the system**

Eve



Alice

Alice's computer

Remote system

Attacking Passwords

- **Direct attack**: Use system as an *oracle;* try to log in with different passwords
  - Defense: Minimize error information
  - Defense: Slowing down password verification
  - Defense: Limit number of login attempts per user
- Attacker: Try different users for common passwords
- Compromise password database
  - Huge yield compared to user-side attacks
  - https://haveibeenpwned.com/
  - Password reuse issues

Protecting Passwords

- How can the system verify that the password Alice entered is correct?

- Naive solution:
    - Store a copy of the password and compare provided copy to the stored one.
- Problem?
    - If system is compromised, passwords are revealed
    - Same passwords may be used on other systems

Protecting Passwords

- Other solutions?
  - Hint: System does not need to know the password, only be able to verify it is correct.
- What if the system stores a cryptographic hash of the password?
  - H(password)
  - Hash must be *pre-image resistant*
- Better… but still problematic.

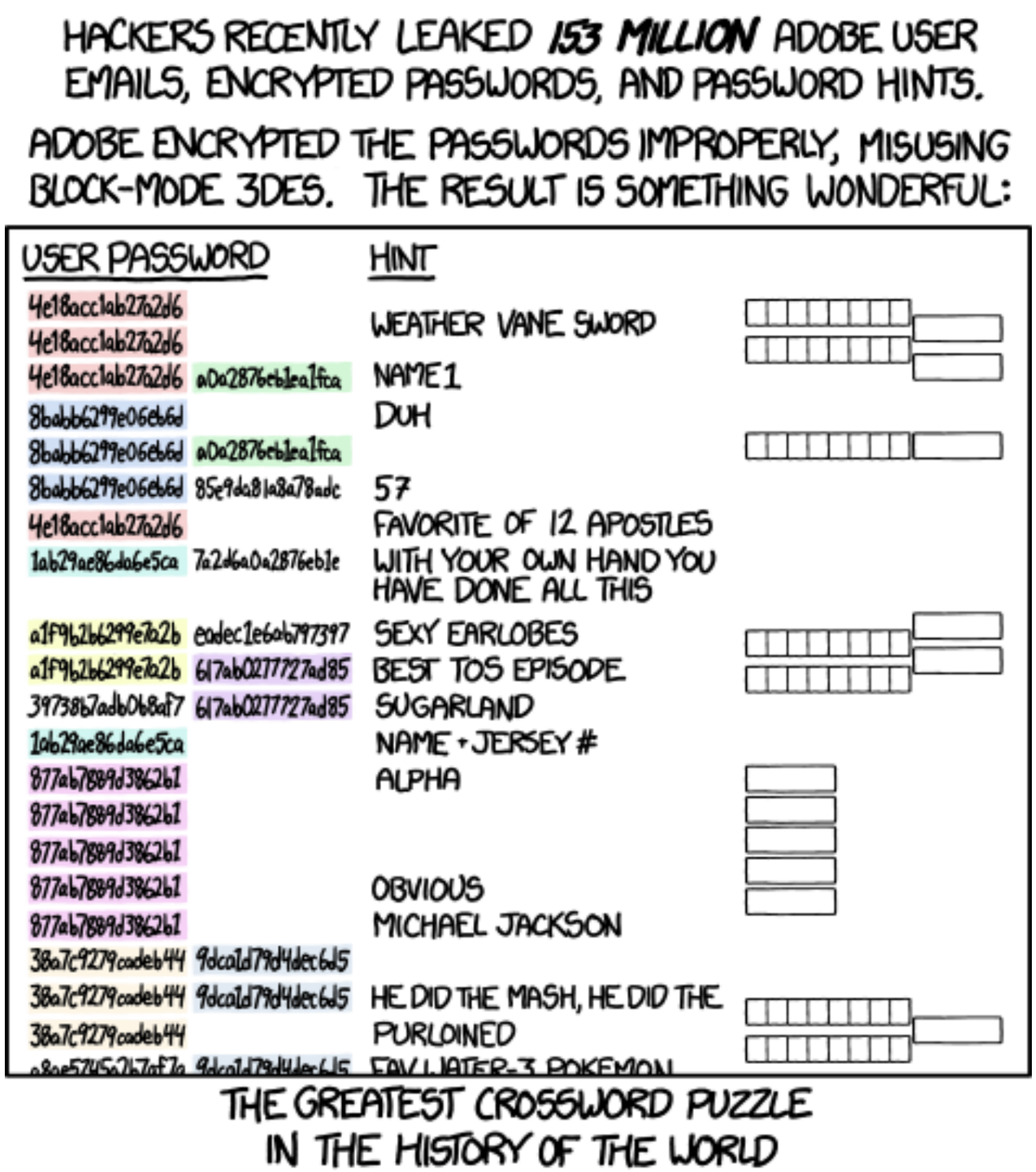
Attacking Passwords

- Given a hash of a password, Eve can use it to validate guesses
  - Also, obvious which users have identical passwords
- **Dictionary attacks**
  - Dictionary: collection of possible, or likely, password strings
  - Try every string in the dictionary until the correct entry is found.
- Pre-compute hashes of all strings in the dictionary, then perform reverse look-ups by hash to find corresponding password.

Attacking Passwords

- **Dictionary attacks** cost example
  - Assume passwords are composed of upper or lower case letters or digits
  - 26 + 26 + 10 = 62 ≈ 64 possible values per character
  - $64^n = 2^{6n}$ possible passwords of length n
  - For $n = 6$, 2^{36} possible password strings
    - ≈ 10 TB to store all possible 6-character passwords and respective SHA-1 hashes
- Can be reduced using techniques like *rainbow tables*.

Attacking Passwords



THE GREATEST CROSSWORD PUZZLE IN THE HISTORY OF THE WORLD

Protecting Passwords

Make Dictionary Attack Harder

- **Salting**
  - Note, the attacker only had to compute one dictionary of hashes that could then be used for any user's password hash from any system.
  - We can parameterize, or "salt", password hashes with unique random numbers
    - *Public* salting: Instead of storing $H(p)$, store $(r, H(r \| p))$, where r is random string (salt).
  - Pre-computation is no longer possible. Attacker must compute unique hashes for every target.
    - What if the database is hacked & r is leaked?

Protecting Passwords

Make Dictionary Attack Harder

- **Salting**
  - Note, the attacker only had to compute one dictionary of hashes that could then be used for any user's password hash from any system.
  - We can parameterize, or "salt", password hashes with unique random numbers
    - *Secret* salting: Instead of storing $H(p)$, store $H(r||p)$, where r is a *short* random string (e.g. 12 bits).
      - Verification needs to enumerate all (4096 for 12 bits) possible r; Tolerable for typical user login.
  - Attacker's work is extended for 4096 times no matter what.

Protecting Passwords

Make Dictionary Attack Harder

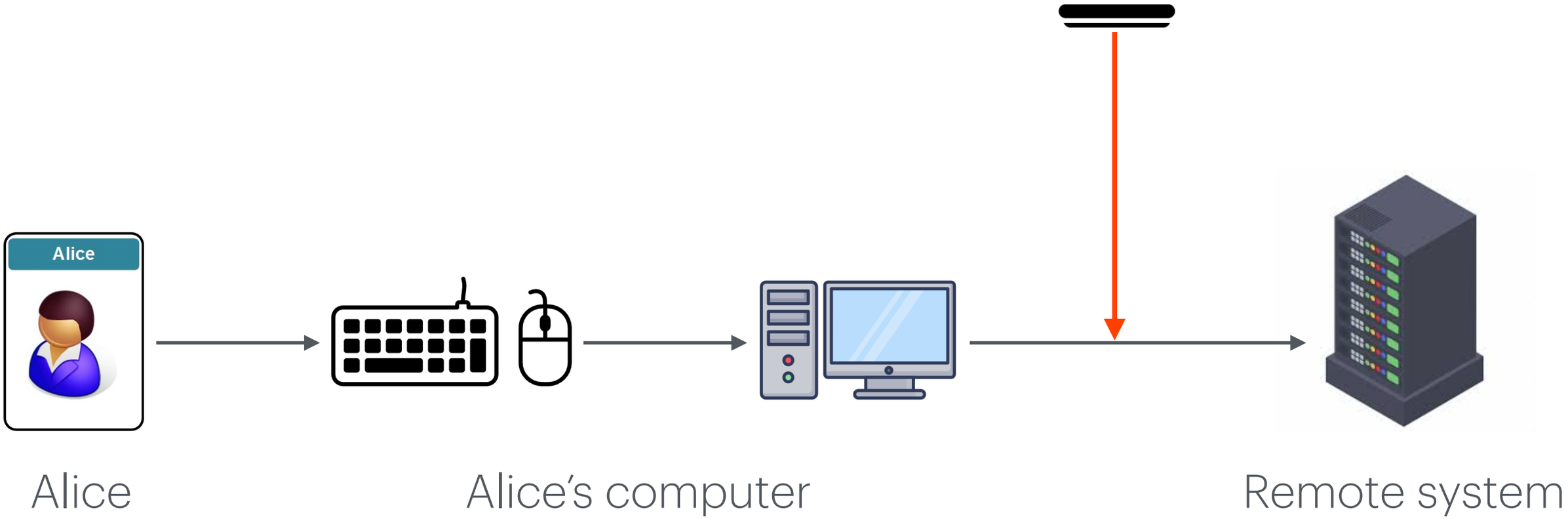
- **Slow hash function**
  - The computation to verify a password for a given user on a legitimate system happens relatively infrequently, but an attacker attempting to crack a password hash must perform many, many attempts
- Conclusion: Use a deliberately slow and resource-consuming hashing function
  - PBKDF2: Time consuming but can be accelerated using parallel comp.
  - Scrypt: Time & space consuming. Provably no good time/space tradeoff exists.

Password-based Authentication: Defend against Evasdropper

Attacking Passwords

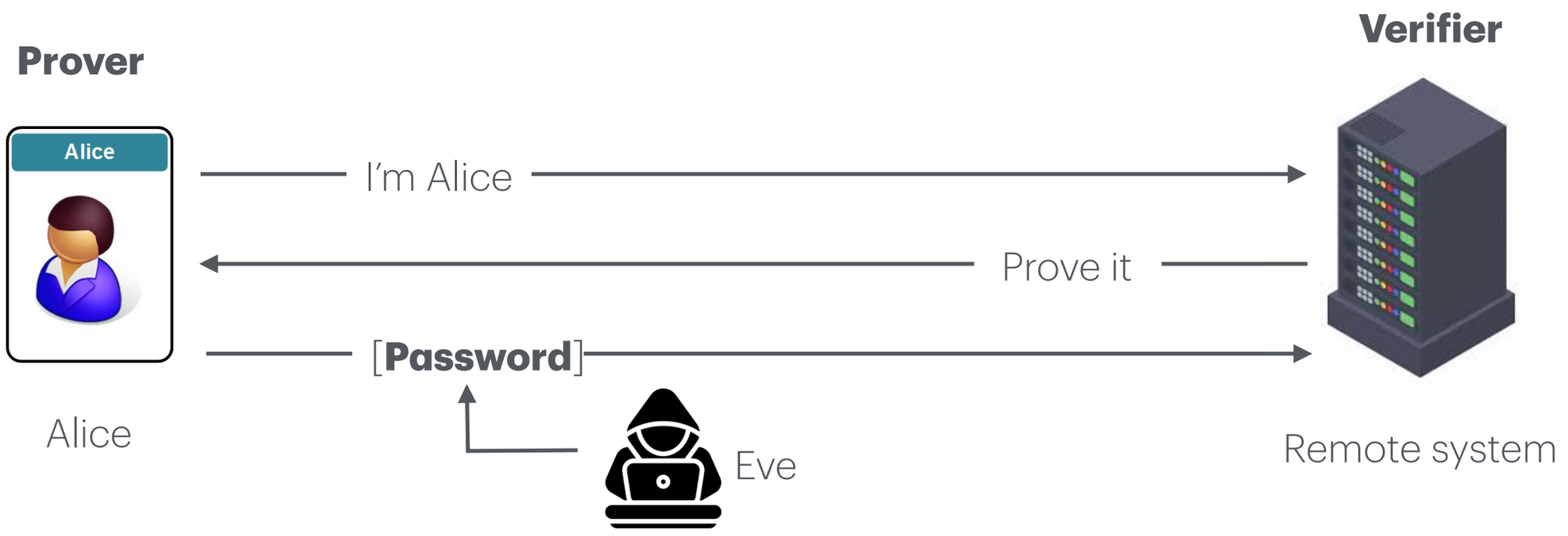
- Get it from Alice
- **Intercept it**
- Get it from the system

Eve



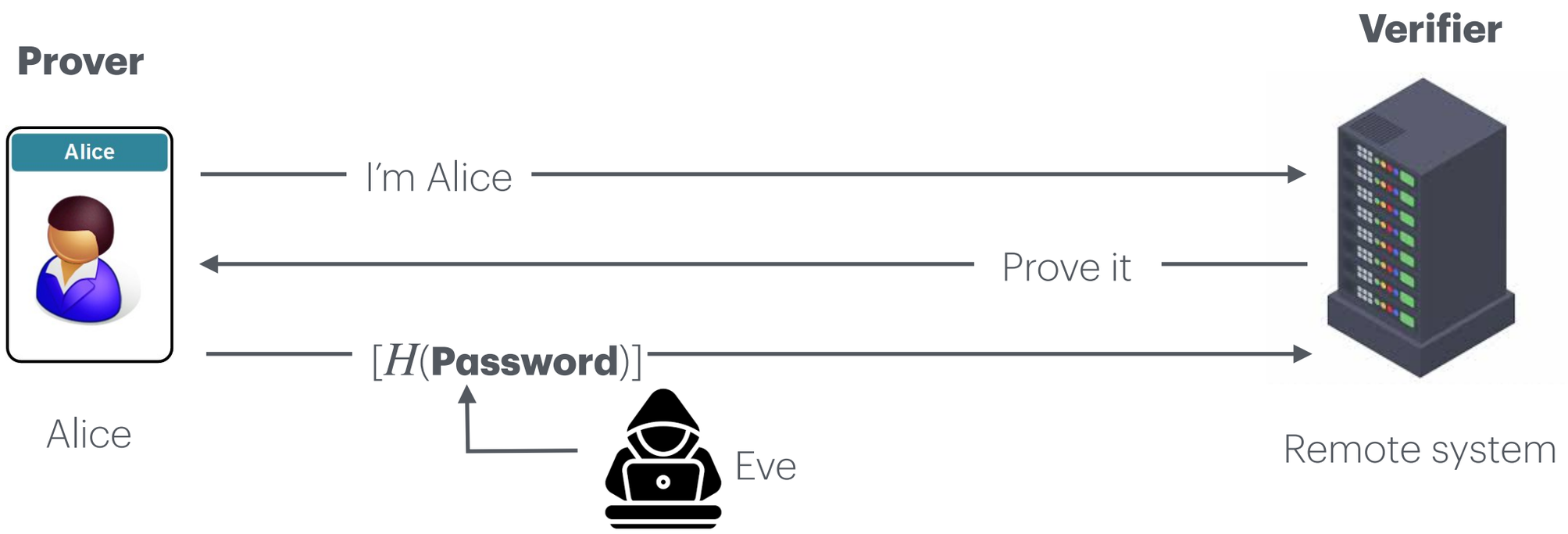
Attacking Passwords

- Alice sends password directly to remote system for authentication.
  - Sounds OK?: anyway the connection will be encrypted.
  - Be conservative:
    - We have seen so many attacks on encrypted comm. protocols (e.g. SSL)
- Once leaked, the attacker can freely use it & deduce more information



Attacking Passwords

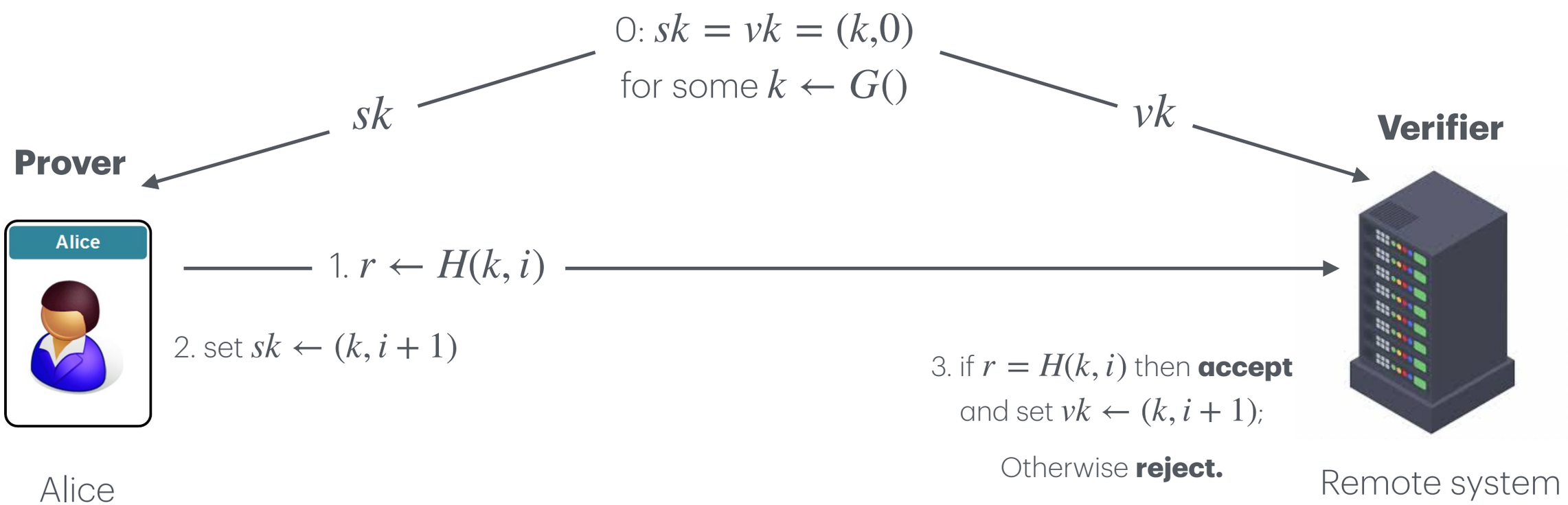
- Can Alice send password **hash** to remote system for authentication?
  - Not feasible for most salting method (added at the server side)
  - Still susceptible to *replay attack*.



Solution: One-time password

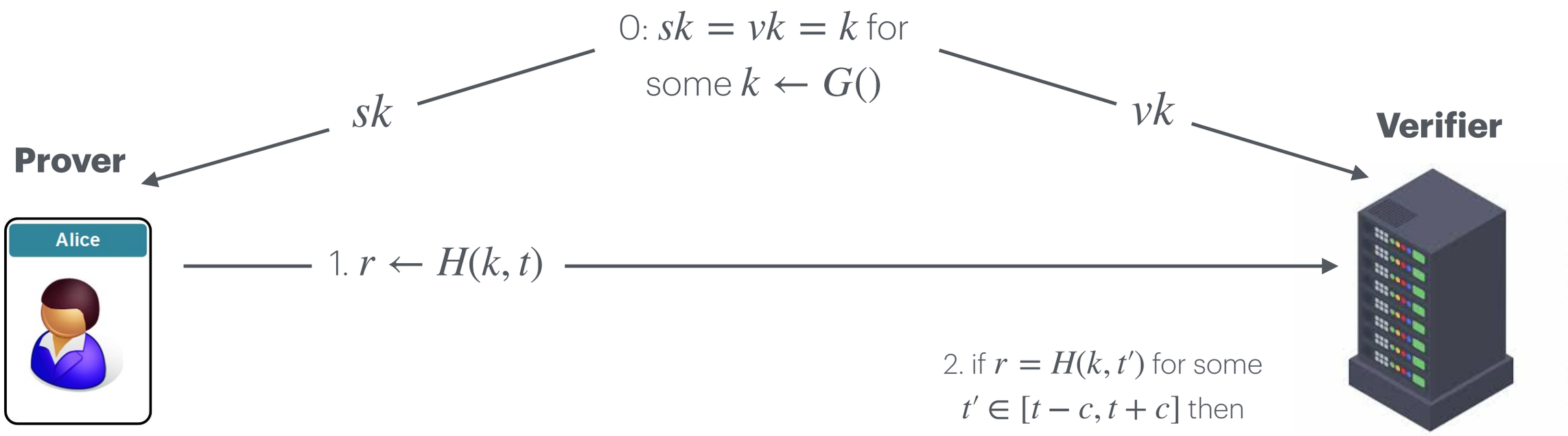
- Each password is used **only once**.
- Such authentication can be realized in the following ways:
  - The user and the system initially agree on a sequence of passwords
    - Simple solution but requires maintenance of the shared list
  - The user updates her password with each instance of the authentication protocol
    - E.g., the user sends the new password encrypted under a key derived from the current password
- Crucially relies on the correct communication of the new passwords to the system

Hash-based One-Time Password (HOTP)



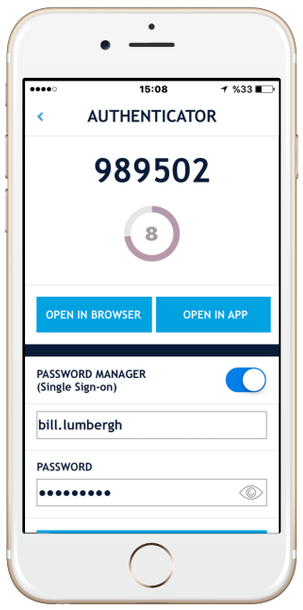
- What is this (sk, vk)?
  - Created at registration time. E.g. Hardware-embedded. Combined with token-based authentication.
- Problem: Needs explicit counter i. Can still be "almost static" if the protocol not executed frequently.

Time-based One-Time Password (TOTP)



Alice

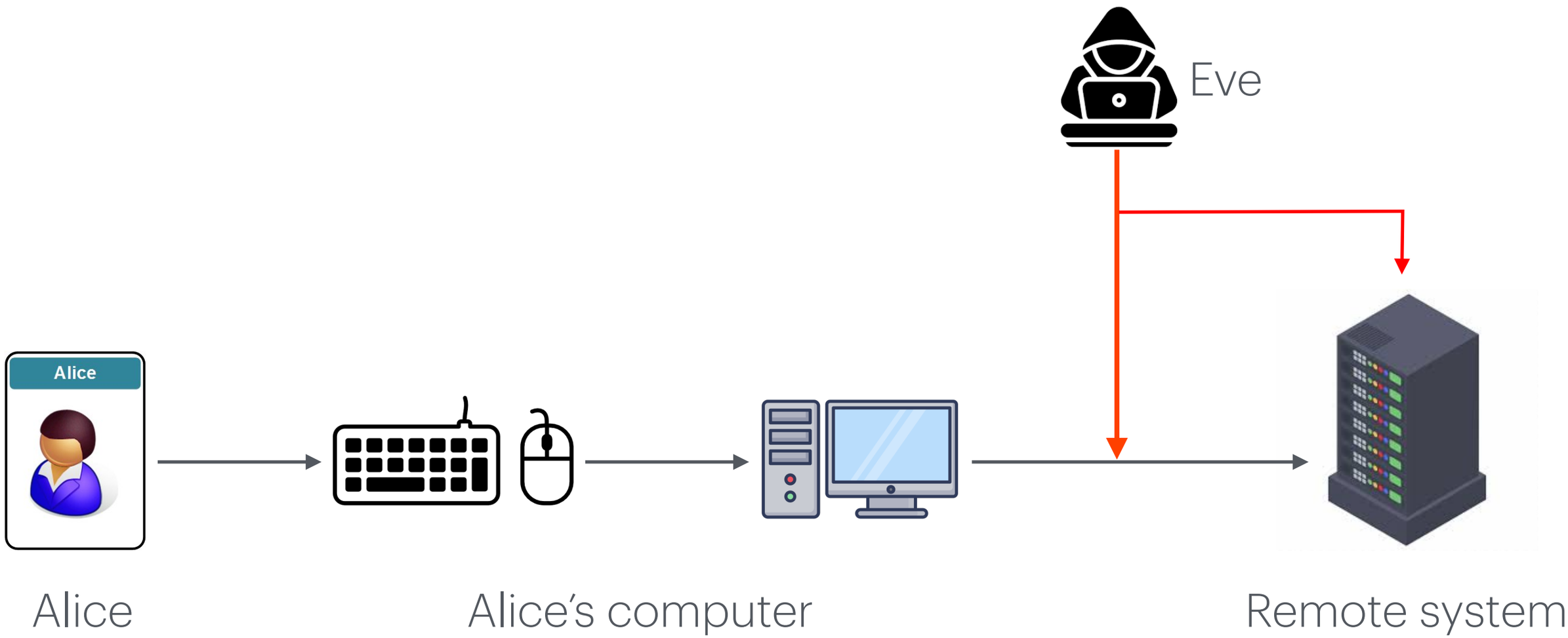
Remote system



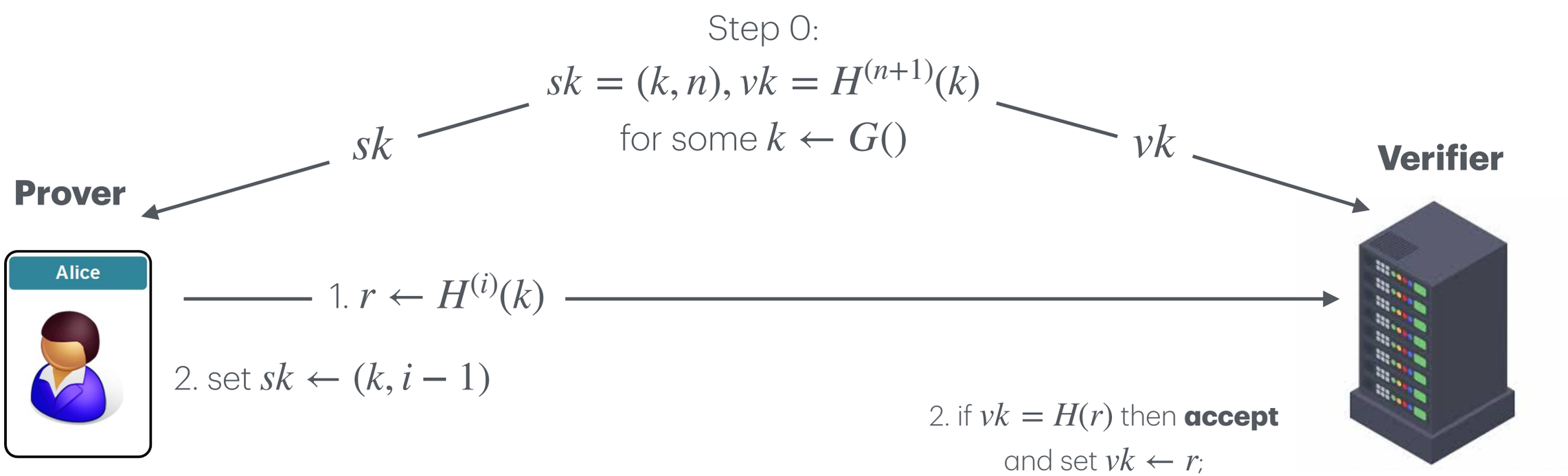
- t: a counter derived from current time.
- Server allow certain clock skew from the Prover: check all $t' \in [t \pm c]$
- The password is usually short (eg. 6 digits)

S/Key System

- A SPoF in HOTP & TOTP: Server stores vk for all clients.
  - If leaked, needs re-registration of all users.



S/Key System



Alice

Remote system

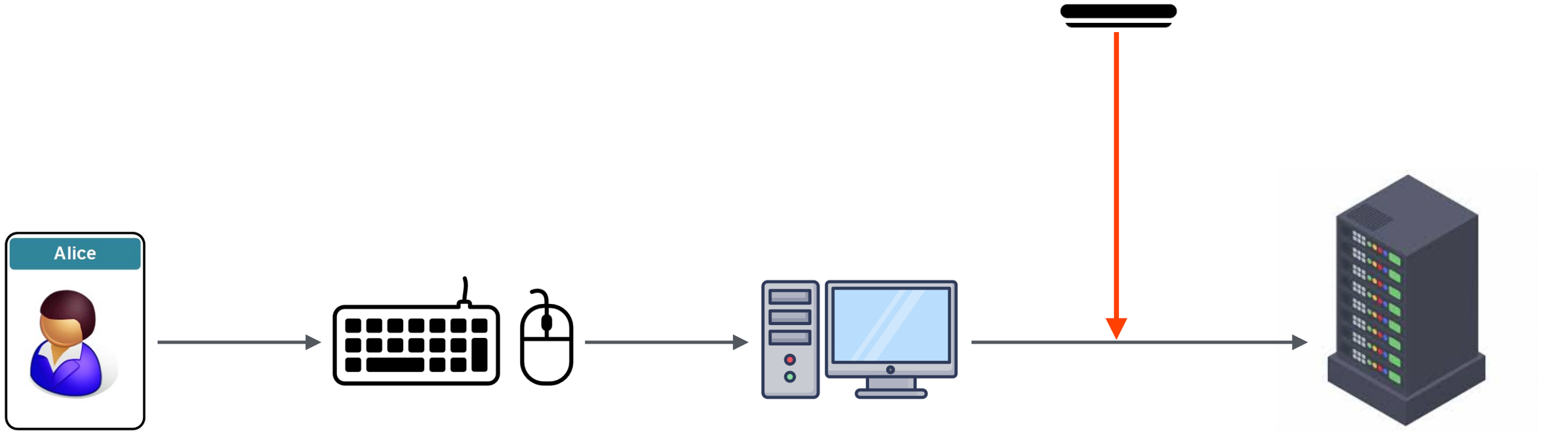
- Secure even if vk is leaked (assuming $H^{(n)}$ is one-way, e.g. SHA256).
- Problem:
  - Need re-registration after n authentications: usually $n < 10^6$
  - r needs to be long enough (>128 bits) to ensure one-wayness

Password-based Authentication: Defend against Active Attack

Attacking Passwords

- Get it from Alice
- ~~**Intercept it**~~ Phishing-then-impersonate
- Get it from the system

Eve

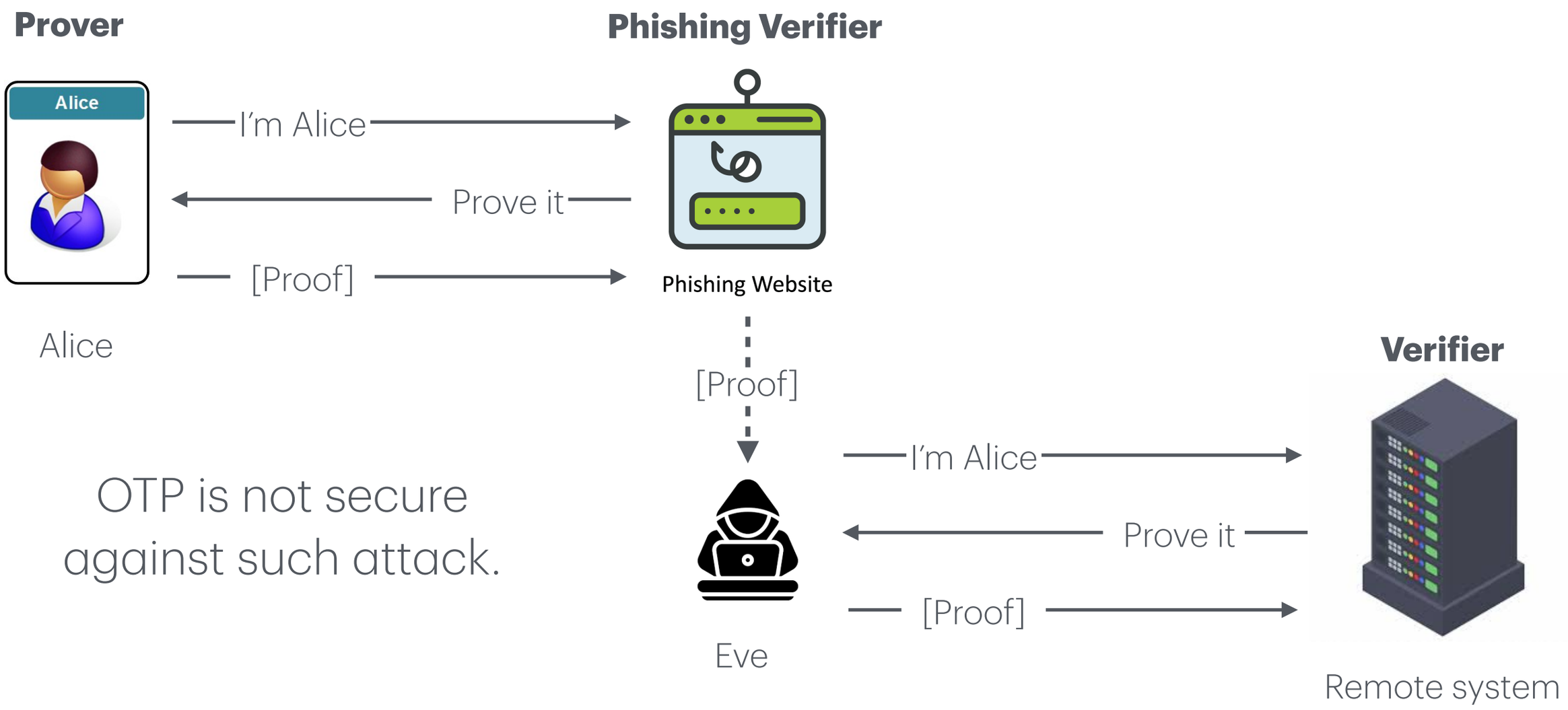


Alice

Alice's computer

Remote system

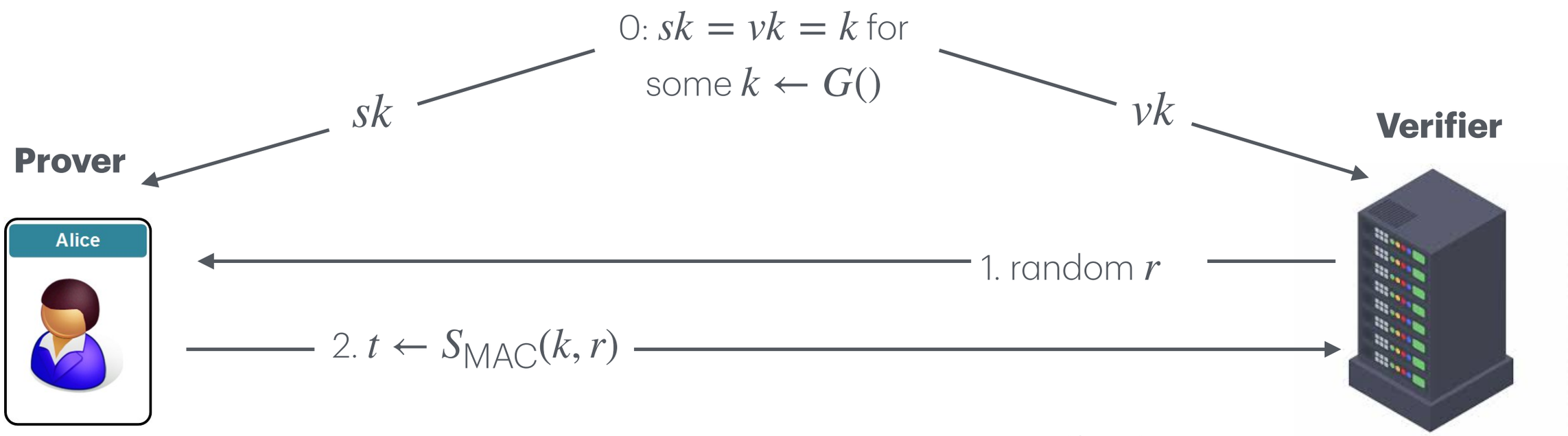
Active (Offline) Attack



OTP is not secure against such attack.

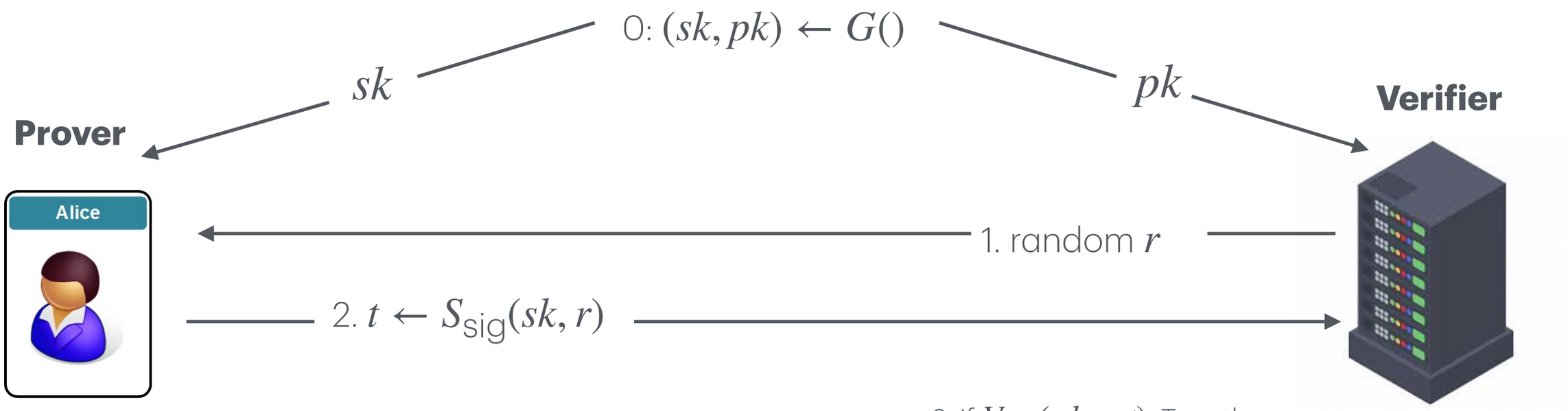
- **Offline** fake ATM:
  - interacts with user; *later* tries to impersonate user to real ATM
- **Offline** phishing:
  - phishing site interacts with user; *later* authenticates to real site

Challenge-response protocols



- $(S_{\text{MAC}}, V_{\text{MAC}})$: a secure MAC
- Why is it secure against an active attacker?
  - What if vk is leaked?

Challenge-response protocols



- $(S_{\text{sig}}, V_{\text{sig}})$: a secure digital signature
- Secure against an active attacker even if pk is leaked.
  - What about Man-in-The-Middle?

Summary for Password-Based Authentication

- “Secret only you know”
- Issue: Can be leaked at various points
- Mitigations
  - Salting: defend against dictionary attack, leakage at remote system/database
  - One-time password: defend against communication intercepting
    - Idea: convince the verifier without leaking the secret
    - *Replace* static password: often used in token-based authentication. Generator (and secret key) embedded in hardware, e.g. car keyfob
    - *Combined* with static password: e.g. the Microsoft Authenticator, DUO

Acknowledgement

- The slides of this lecture is developed heavily based on
  - Slides from Prof Nadia Heninger's lecture on Computer Security (https://cseweb.ucsd.edu/classes/wi23/cse127-a/slides/16-authentication.pdf)
  - Slides from Prof Ziming Zhao's past offering of CSE565 (https://zzm7000.github.io/teaching/2023springcse410565/index.html)
  - Slides from Prof Marina Blanton's past offering of CSE565 (https://www.acsu.buffalo.edu/~mblanton/cse565/)
  - Slides from Prof Hongxin Hu's past offering of CSE565

Questions?