

2550 Intro to cybersecurity

L4

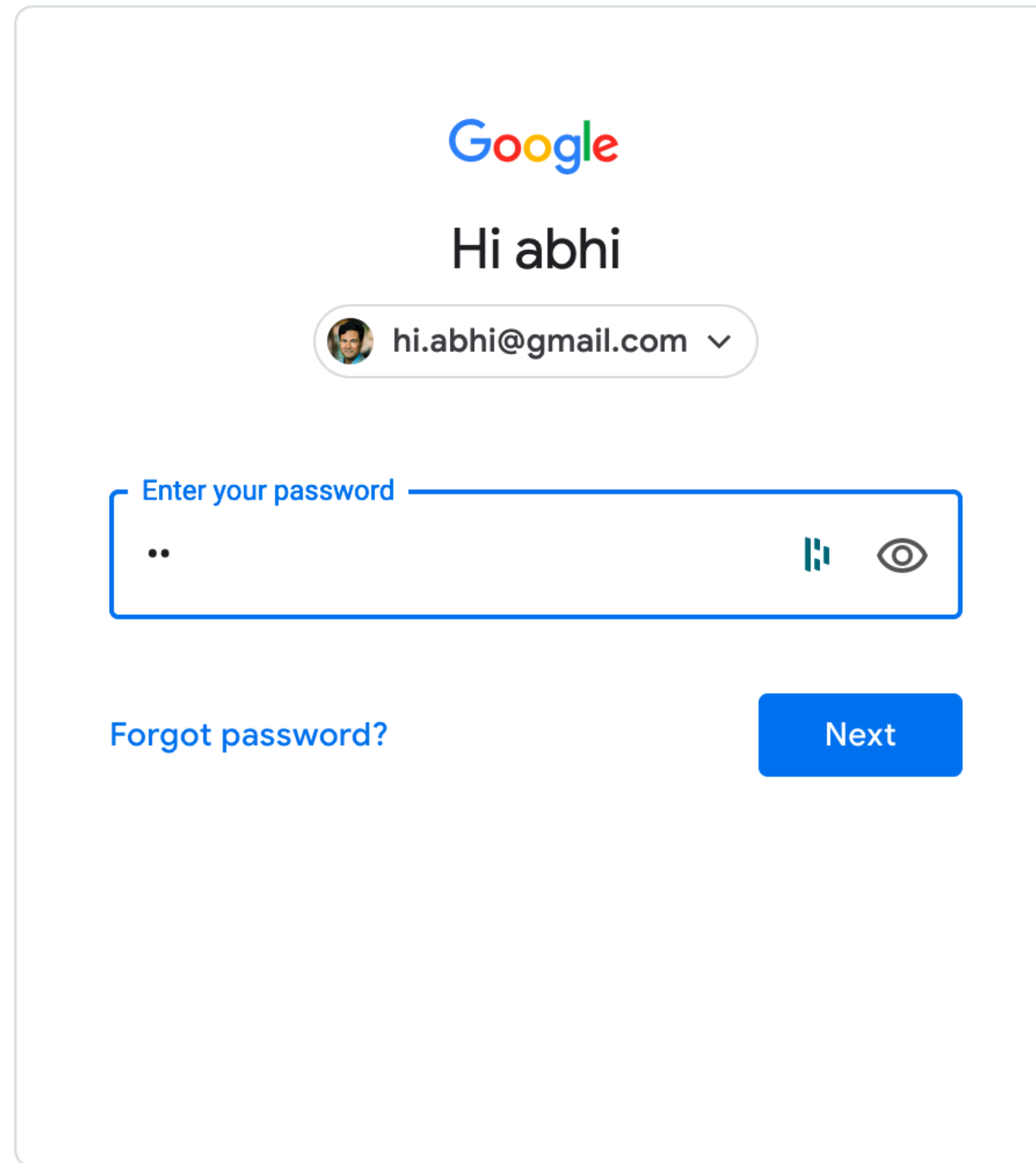
abhi shelat/Ran Cohen

Best practices so far:

Password Storage Summary

- 1. Never store passwords in plain text**
 - 2. Always salt and hash passwords before storing them**
 - 3. Use hash functions with a high work factor**
 - 4. Implement honeywords to detect breaches**
- These rules apply to any system that needs to authenticate users
 - Operating systems, websites, etc.

Still one problem?



The screenshot shows a Google login interface. At the top is the Google logo. Below it, the text "Hi abhi" is displayed. A dropdown menu shows the email address "hi.abhi@gmail.com" with a small profile picture icon to the left and a downward arrow to the right. Below the email is a password input field with a blue border and the placeholder text "Enter your password". The field contains two dots and has icons for password strength and visibility on the right. Below the password field are two links: "Forgot password?" and a blue "Next" button.

Password Recovery/Reset

- Problem: hashed passwords cannot be recovered (hopefully)



“Hi... I forgot my password. Can you email me a copy? Kthxbye”

- This is why systems typically implement password **reset**
 - Use out-of-band info to authenticate the user
 - Overwrite `hash(old_pw)` with `hash(new_pw)`
- Be careful: its possible to crack password reset

Cracking Password Reset

- Typical implementations use **Knowledge Based Authentication (KBA)**
 - What was your mother's maiden name?
 - What was your prior street address?
 - Where did you go to elementary school

Cracking Password Reset

- Typical implementations use **Knowledge Based Authentication (KBA)**
 - What was your mother's maiden name?
 - What was your prior street address?
 - Where did you go to elementary school
- Problems?


Cracking Password Reset

- Typical implementations use **Knowledge Based Authentication (KBA)**
 - What was your mother's maiden name?
 - What was your prior street address?
 - Where did you go to elementary school
- **Problems?**
 - This information is widely available to anyone
 - Publicly accessible social network profiles
 - Background-check services like Spokeo

Cracking Password Reset

- Typical implementations use **Knowledge Based Authentication (KBA)**
 - What was your mother's maiden name?
 - What was your prior street address?
 - Where did you go to elementary school
- **Problems?**
 - This information is widely available to anyone
 - Publicly accessible social network profiles
 - Background-check services like Spokeo
- **Experts recommend that services not use KBA**
 - When asked, users should generate random answers to these questions

Other roots of identity



Account recovery

hi.abhi@gmail.com ▾

Enter the last password you remember using with this Google Account

Enter last password

[Try another way](#)

English (United States) ▾

[Help](#)

[Privacy](#)

[Terms](#)

Forgot username or password

Identification

[Have a question? >](#)

Help us verify your identity.

For your security, please choose one of the options to verify your identity and provide the other requested information.

Choose one

Social Security number ▾

Social Security number

xxx-xx-xxxx

[Don't have a Social Security number? >](#)

Account type

Chase ATM/debit/prepaid card or credit card

Chase commercial loan

Other Chase account (e.g., checking, savings, mortgage application, commercial term loan, auto loan or lease)

Choosing Passwords

Bad Algorithms

Better Heuristics

Password Reuse

Password Reuse

- People have difficulty remembering >4 passwords
 - Thus, people tend to reuse passwords across services
 - What happens if any one of these services is compromised?
- Service-specific passwords are a beneficial form of compartmentalization
 - Limits the damage when one service is inevitably breached
- Use a password manager
- Some service providers now check for password reuse
 - Forbid users from selecting passwords that have appeared in leaks

Sites



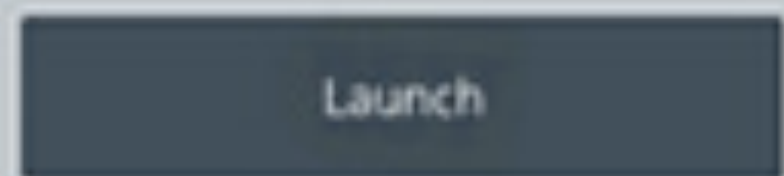
Favorites (8)



AirBnB
fan@lastpass.com



Amazon
fan@lastpass.com



Best Buy
fan@lastpass.com



Dropbox
fan@lastpass.com



Evernote
fan@lastpass.com



Facebook
fan@lastpass.com



Pocket
fan@lastpass.com



Twitter
fan@lastpass.com

Banking and Finance (3)

Read Only • Shared Folder



Bank of America
fan@lastpass.com



Fidelity
fan@lastpass.com



Mint
fan@lastpass.com



Dashlane

The screenshot shows the Dashlane desktop application interface. On the left is a dark sidebar with navigation options: VAULT (Passwords, Secure Notes, Personal Info, Payments, IDs, Receipts) and SECURITY (Identity Dashboard, Password Health). At the bottom of the sidebar, it shows 'Getting Started' at 90% and a premium status of '5 days left of Premium'. The main area displays a list of 22 logins, each with a green database icon, a name, and a username. The list includes:

- 1.1 (wgr614v5) admin
- 1.20 (812b) root
- 1.50 (airstation) root
- 10.0.1.2 admin
- 10.0.1.50 812A12 (5Ghz)
- 10.250.224.2 ashelat
- 104.131.125.119 abhi@arqspin.com
- 11.1 (airstation) root
- 11.1 (dd-wrt) root

The image shows two smartphone screens. The background screen displays the Firefox Lockwise app's login list, which is identical to the desktop version. The foreground screen shows the app's onboarding screen with the Firefox Lockwise logo and the text: 'Take your passwords everywhere', 'Get Started', and 'To use Firefox Lockwise, you'll need a Firefox Account with saved logins. Learn more >'. The time on the foreground phone is 1:30.



Home

Notify me

Domain search

Who's been pwned

Passwords

API

About

Donate

';--have i been pwned?

Check if you have an account that has been compromised in a data breach

264

pwned websites

4,859,717,682

pwned accounts

61,081

pastes

59,268,789

paste accounts

Two Factor Authentication

Biometrics

SMS

Authentication Codes

Smartcards & Hardware Tokens

Types of Secrets

- Actors provide their secret to **log-in** to a system
- Three classes of secrets:
 1. Something you know
 - Example: a password
 2. Something you have
 - Examples: a smart card or smart phone
 3. Something you are
 - Examples: fingerprint, voice scan, iris scan

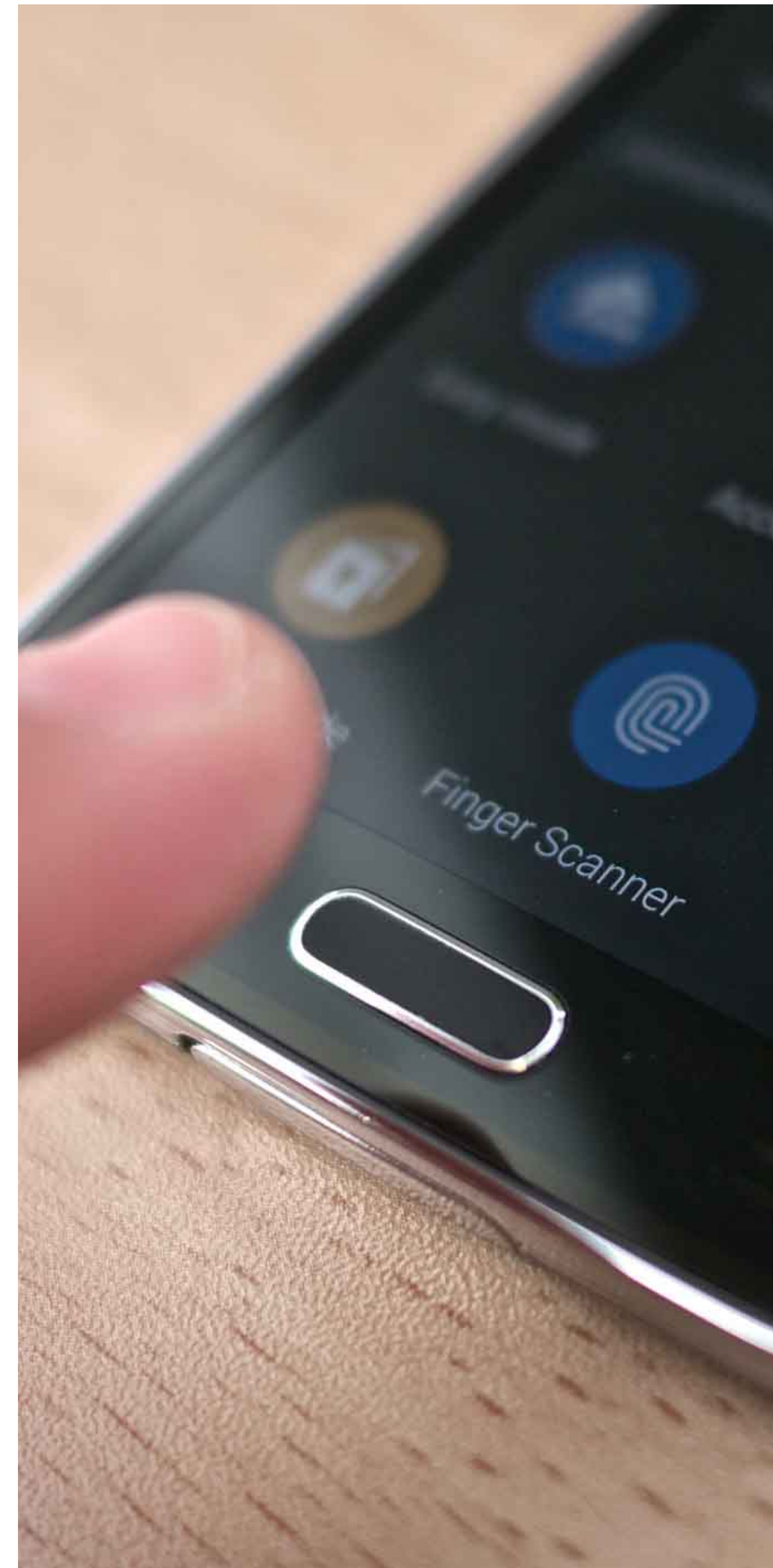
Biometrics

- ancient Greek: bios = "life", metron = "measure"
- Physical features
 - Fingerprints
 - Face recognition
 - Retinal and iris scans
 - Hand geometry
- Behavioral characteristics
 - Handwriting recognition
 - Voice recognition
 - Typing cadence
 - Gait

Fingerprints

- Ubiquitous on modern smartphones, some laptops
- Secure?
 - May be subpoenaed by law enforcement
 - Relatively easy to compromise
 1. Pick up a latent fingerprint (e.g. off a glass) using tape or glue
 2. Photograph and enhance the fingerprint
 3. Etch the print into gelatin backed by a conductor
 4. Profit ;)

https://www.theregister.co.uk/2002/05/16/gummi_bears_defeat_fingerprint_sensors/



Facial Recognition

- Popularized by FaceID on the iPhone X
- Secure?



Facial Recognition

- Popularized by FaceID on the iPhone X
- Secure?
 - It depends



Facial Recognition

- Popularized by FaceID on the iPhone X
- Secure?
 - It depends
- Vulnerable to law enforcement requests
- Using 2D images?
 - Not secure
 - Trivial to break with a photo of the target's face



Facial Recognition

- Popularized by FaceID on the iPhone X
- Secure?
 - It depends
- Vulnerable to law enforcement requests
- Using 2D images?
 - Not secure
 - Trivial to break with a photo of the target's face
- Using 2D images + 3D depth maps?
 - More secure, but not perfect
 - Can be broken by crafting a lifelike mask of the target





Specially processed area

2D images

Silicone nose

3D printed frame



By Press Association

Saturday, October 19, 2019 - 01:20 PM

Google has confirmed the Face Unlock system on its new Pixel 4 smartphone can allow access to the device even when the user has their eyes closed.

Early testers of the phone, as well as security experts, have raised concerns it could lead to unauthorised access to the device.

It has been suggested someone else could gain access to the phone by holding it in front of the face of its sleeping owner, but Google said it meets security requirements.

The technology giant unveiled the new phone earlier this week.

In a statement, Google said: “Pixel 4 Face Unlock meets the security requirements as a strong biometric and can be used for payments and app authentication, including banking apps.

“It is resilient against unlock attempts via other means, like with masks.

“If you want to temporarily disable Face Unlock, you can use lockdown mode to temporarily require a PIN/pattern/password.

Voice Recognition

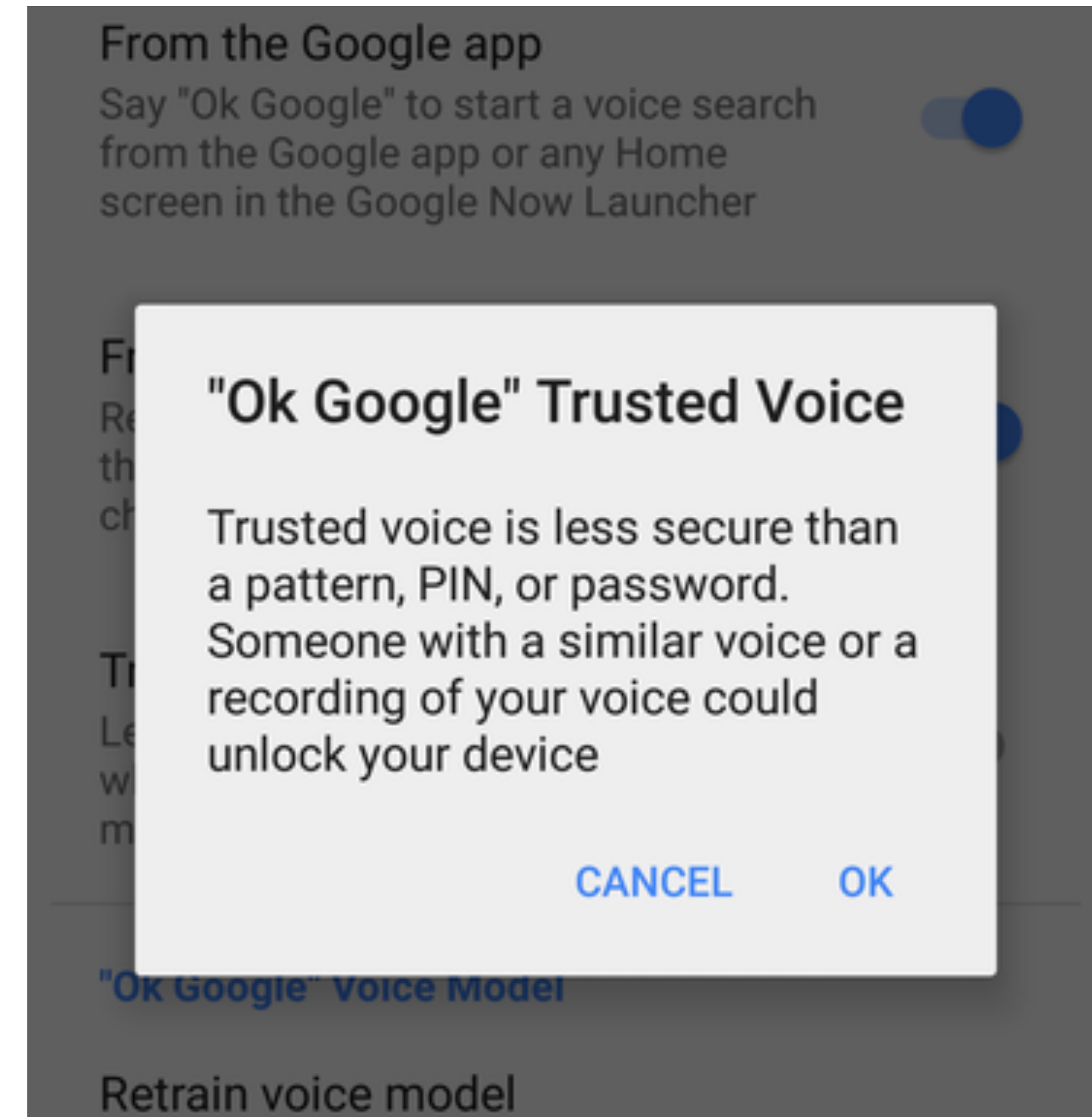
- Secure?
 - Very much depends on the implementation

Voice Recognition

- Secure?
 - Very much depends on the implementation
- Some systems ask you to record a static phrase
 - E.g. say “unlock” to unlock
 - This is wildly insecure
 - Attacker can record and replay your voice

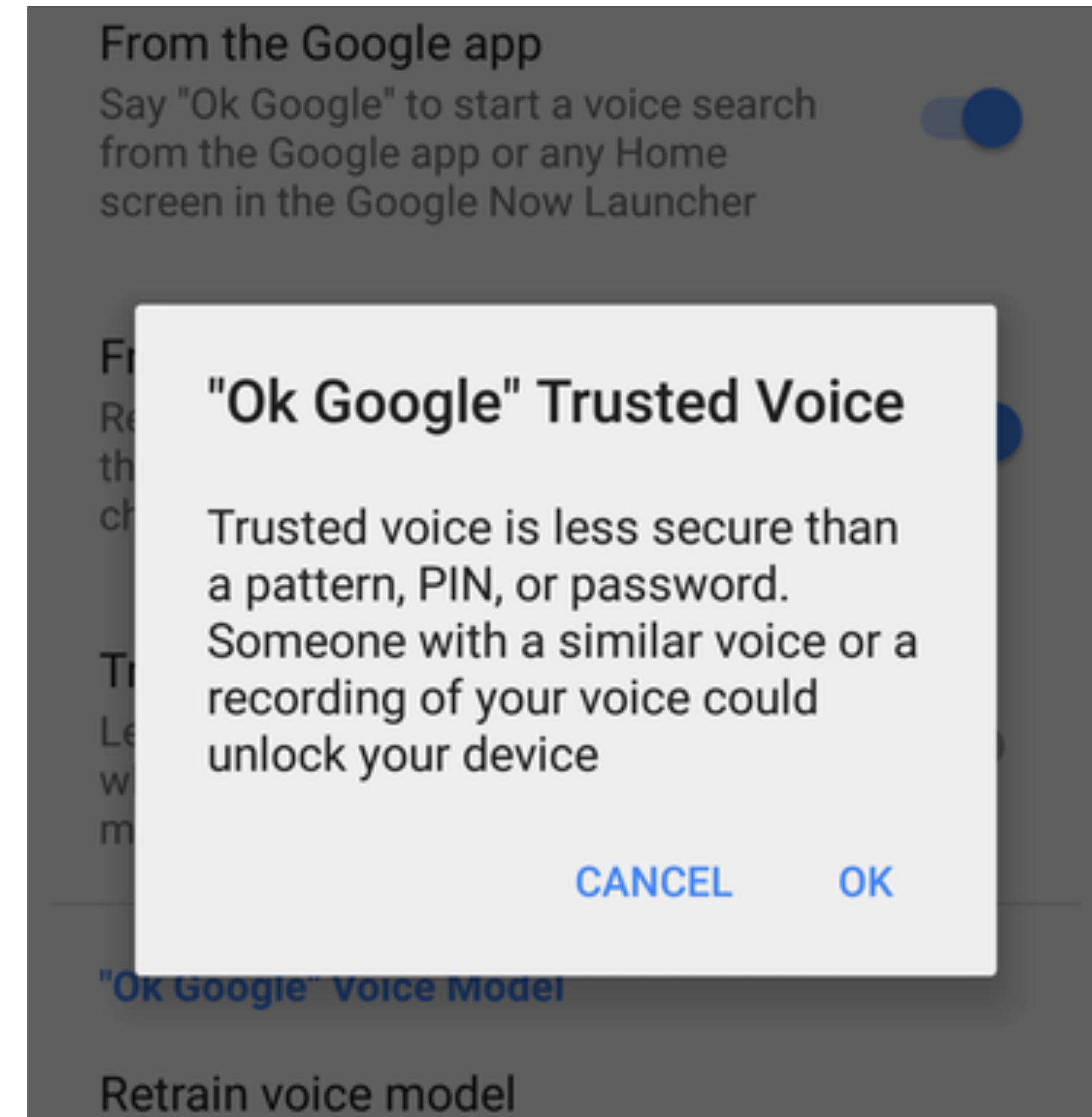
Voice Recognition

- Secure?
 - Very much depends on the implementation
- Some systems ask you to record a static phrase
 - E.g. say “unlock” to unlock
 - This is wildly insecure
 - Attacker can record and replay your voice



Voice Recognition

- Secure?
 - Very much depends on the implementation
- Some systems ask you to record a static phrase
 - E.g. say “unlock” to unlock
 - This is wildly insecure
 - Attacker can record and replay your voice
- Others ask you to train a model of your voice
 - Train the system by speaking several sentences
 - To authenticate, speak several randomly chosen words
 - Not vulnerable to trivial replay attacks, but still vulnerable
 - Given enough samples of your voice, an attacker can train a synthetic voice AI that sounds just like you



Fundamental Issue With Biometrics

- Biometrics are immutable
 - You are the password, and you can't change
 - Unless you plan on undergoing plastic surgery?
- Once compromised, there is no reset
 - Passwords and tokens can be changed
- Example: the Office of Personnel Management (OPM) breach
 - US gov agency responsible for background checks
 - Had fingerprint records of all people with security clearance
 - Breached by China in 2015, all records stolen :(

Something You Have

- Two-factor authentication has become more commonplace
- Possible second factors:
 - SMS passcodes
 - Time-based one time passwords
 - Hardware tokens

SMS Two Factor

- Relies on your phone number as the second factor
 - Key assumption: only your phone should receive SMS sent to your number



SMS Two Factor

- Relies on your phone number as the second factor
 - Key assumption: only your phone should receive SMS sent to your number
- SMS two factor is deprecated. Why?



SMS Two Factor

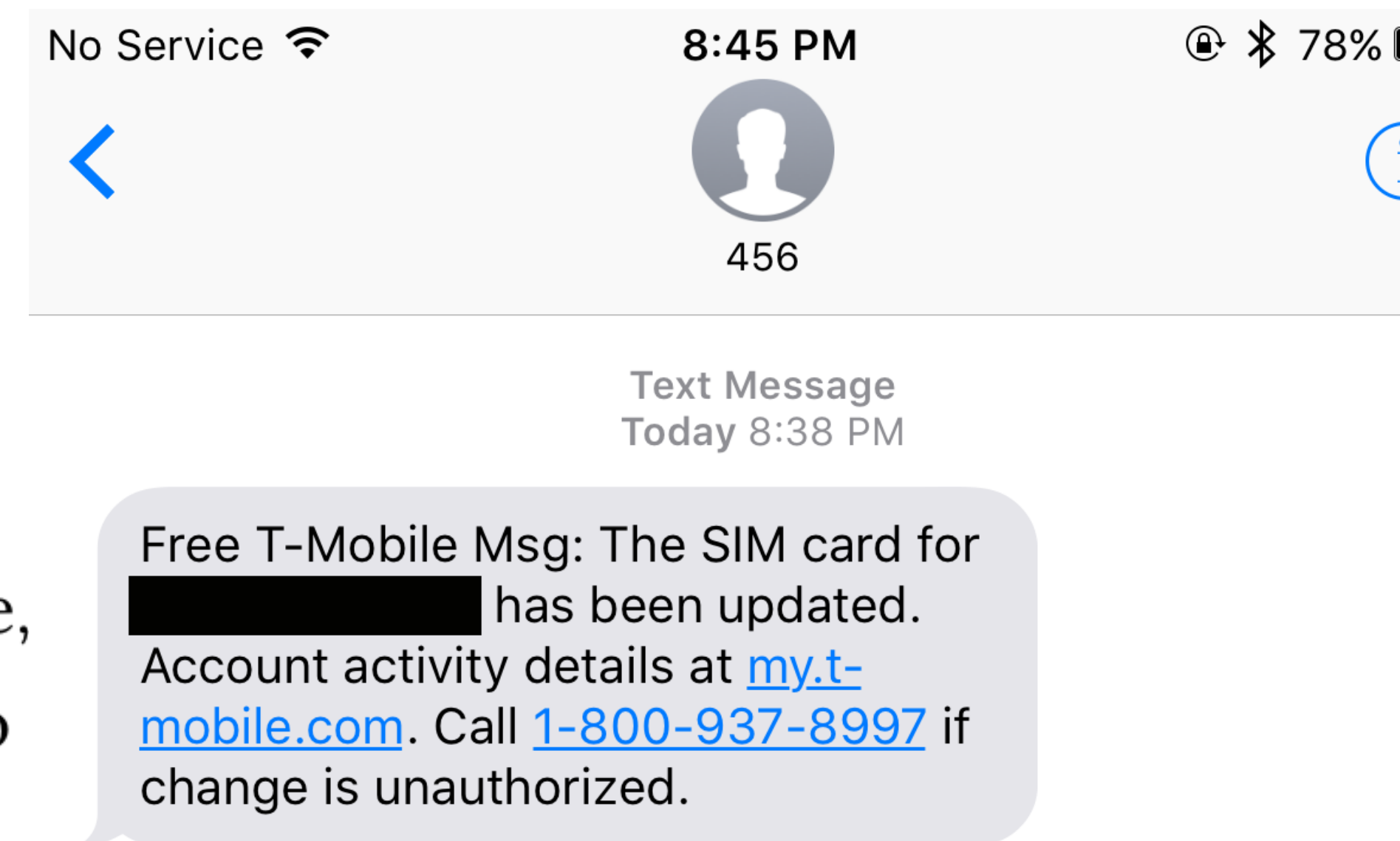
- Relies on your phone number as the second factor
 - Key assumption: only your phone should receive SMS sent to your number
- SMS two factor is deprecated. Why?
- Social engineering the phone company
 1. Call and pretend to be the victim
 2. Say “I got a new SIM, please activate it”
 3. If successful, phone calls and SMS are now sent to your SIM in your phone, instead of the victim
- Not hypothetical: successfully used against many victims



First, criminals call a cell phone carrier's tech support number pretending to be their target. They explain to the company's employee that they "lost" their SIM card, requesting their phone number be transferred, or ported, to a new SIM card that the hackers themselves already own. With a bit of social engineering—perhaps by providing the victim's Social Security Number or home address (which is often available from one of the many data breaches that have happened in the last few years)—the criminals convince the employee that they really are who they claim to be, at which point the employee ports the phone number to the new SIM card.

Game over.

“With someone's phone number,” a hacker who does SIM swapping told me, “you can get into every account they own within minutes and they can't do anything about it.”



One Time Passwords

- Generate ephemeral passcodes that change over time
- To login, supply normal password and the current one time password
- Relies on a shared secret between your mobile device and the service provider
 - Shared secret allows both parties to know the current one time password

Changes every few minutes



ACME INC TOKEN IS:

6883932 



Duo Mobile



Lastpass Authenticator



Google Authenticator

Time-based One-time Password Algorithm

$T0$ = <the beginning of time, typically Thursday, 1 January 1970 UTC>

Tl = <length of time the password should be valid>

K = <shared secret key>

d = <the desired number of digits in the password>

TC = $\text{floor}((\text{unixtime}(\text{now}) - \text{unixtime}(T0)) / Tl)$,

$\text{TOTP} = \text{HMAC}(K, TC) \% 10^d$



Specially formatted
SHA1-based signature

Time-based One-time Password Algorithm

$T0$ = <the beginning of time, typically Thursday, 1 January 1970 UTC>

Tl = <length of time the password should be valid>

K = <shared secret key>

d = <the desired number of digits in the password>

TC = $\text{floor}(\text{unixtime}(\text{now}) - \text{unixtime}(T0)) / Tl$,

$\text{TOTP} = \text{HMAC}(K, TC) \% 10^d$

Specially formatted
SHA1-based signature

Given K , this algorithm can
be run on your phone and by
the service provider

Secret Sharing for TOTP

Enable Two-Step Sign in

An authenticator app generates the code automatically on your smartphone. Free apps are available for all smartphone platforms including iOS, Android, Blackberry and Windows. Look for an app that supports time-based one-time passwords (TOTP) such as Google Authenticator or Duo Mobile.

To set up your mobile app, add a new service and scan the QR code.



If you can't scan the code, enter this secret key manually: fvxo

[USE SMS INSTEAD](#)

[CANCEL](#)

[NEXT STEP](#)

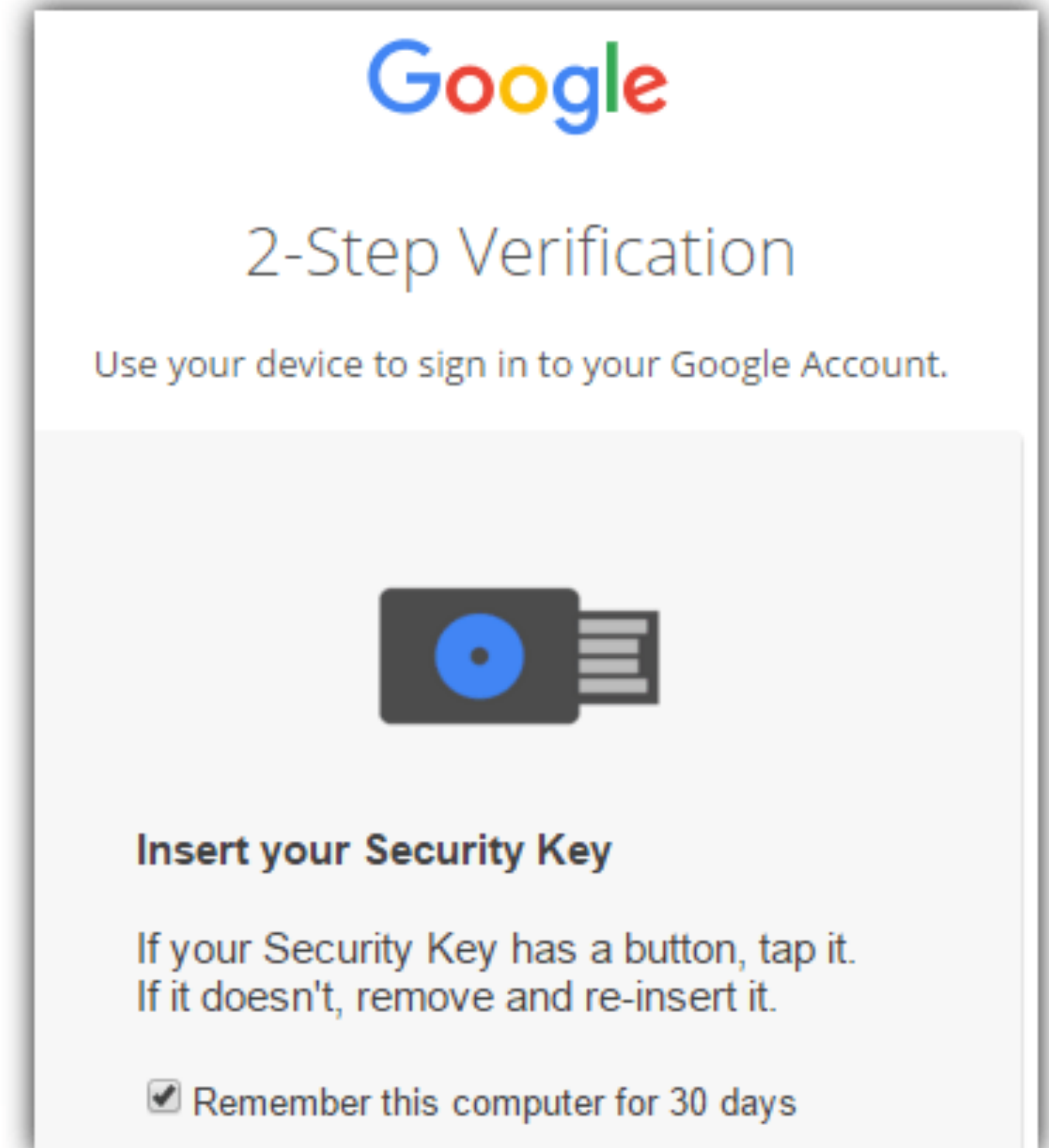
Hardware Two Factor

- Special hardware designed to hold cryptographic keys
- Physically resistant to key extraction attacks
 - E.g. scanning tunneling electron microscopes
- Uses:
 - 2nd factor for OS log-on
 - 2nd factor for some online services
 - Storage of PGP and SSH keys



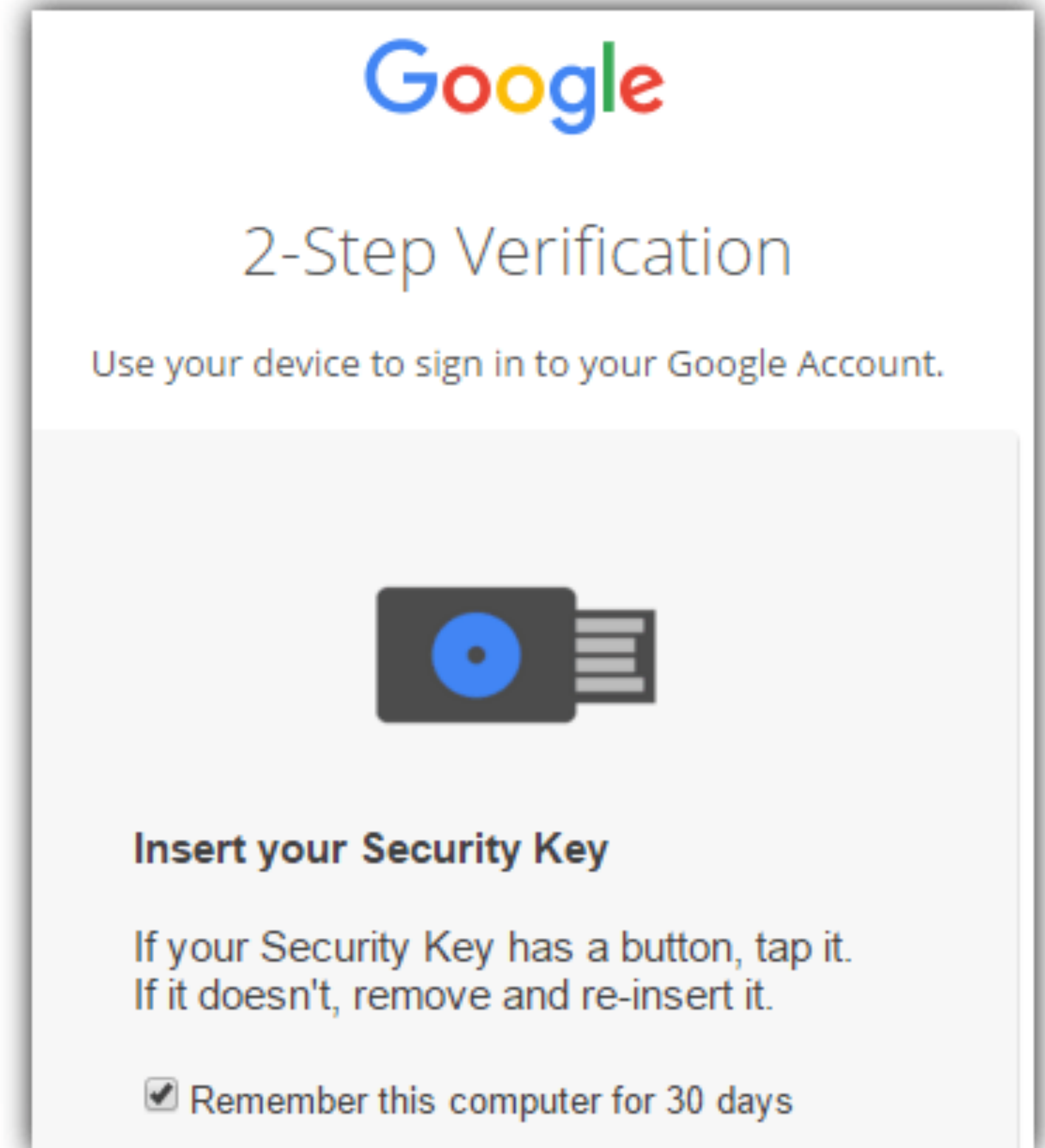
Universal 2nd Factor (U2F)

- Supported by Chrome, Opera, and Firefox (must be manually enabled)
- Works with Google, Dropbox, Facebook, Github, Gitlab, etc.

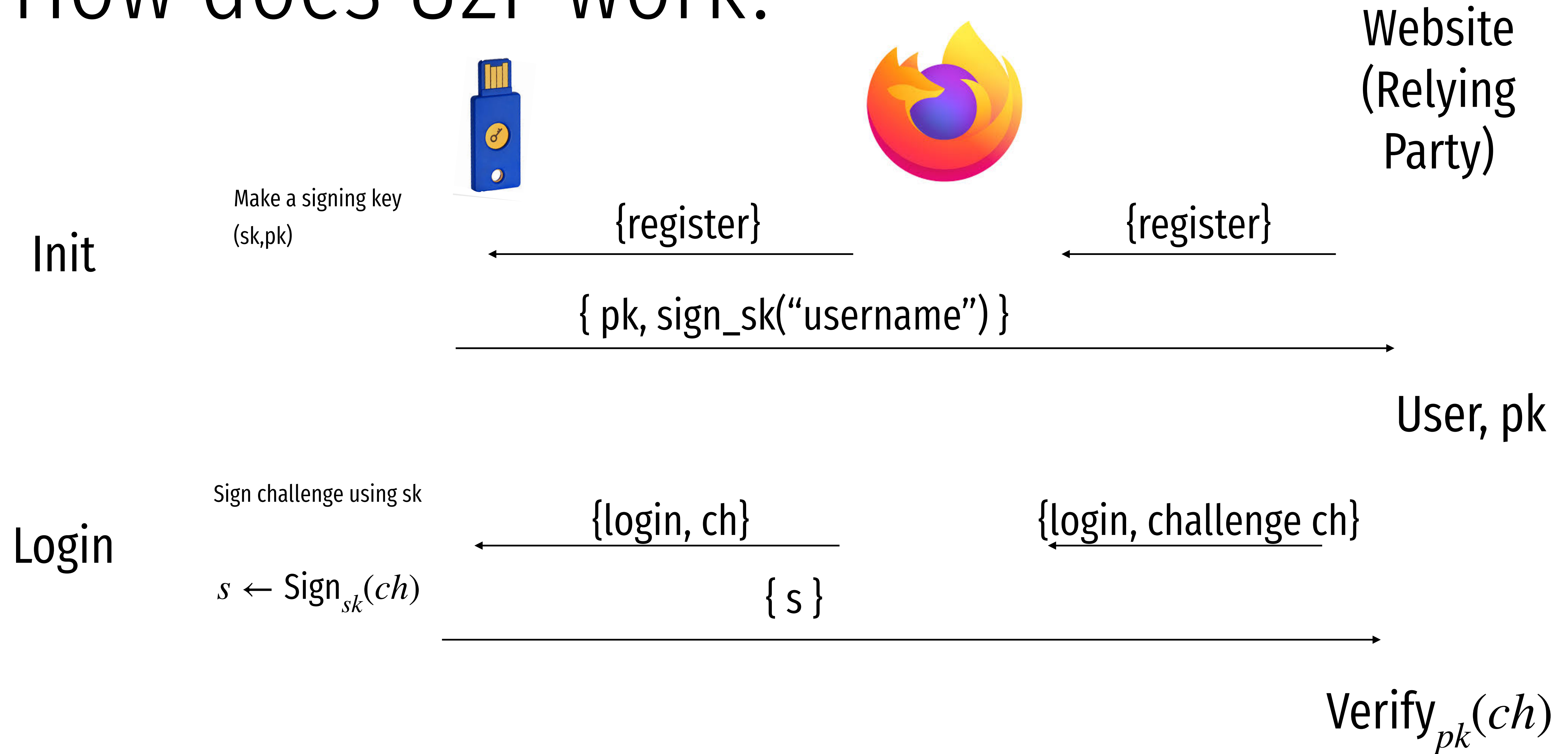


Universal 2nd Factor (U2F)

- Supported by Chrome, Opera, and Firefox (must be manually enabled)
- Works with Google, Dropbox, Facebook, Github, Gitlab, etc.
- Pro tip: always buy 2 security keys
 - Associate both with your accounts
 - Keep one locked in a safe, in case you lose your primary key ;)



How does U2F work?



Vulnerable to simple attack



Welcome

 hi.abhi@gmail.com ▾

Enter your password  

[Forgot password?](#)

[Next](#)

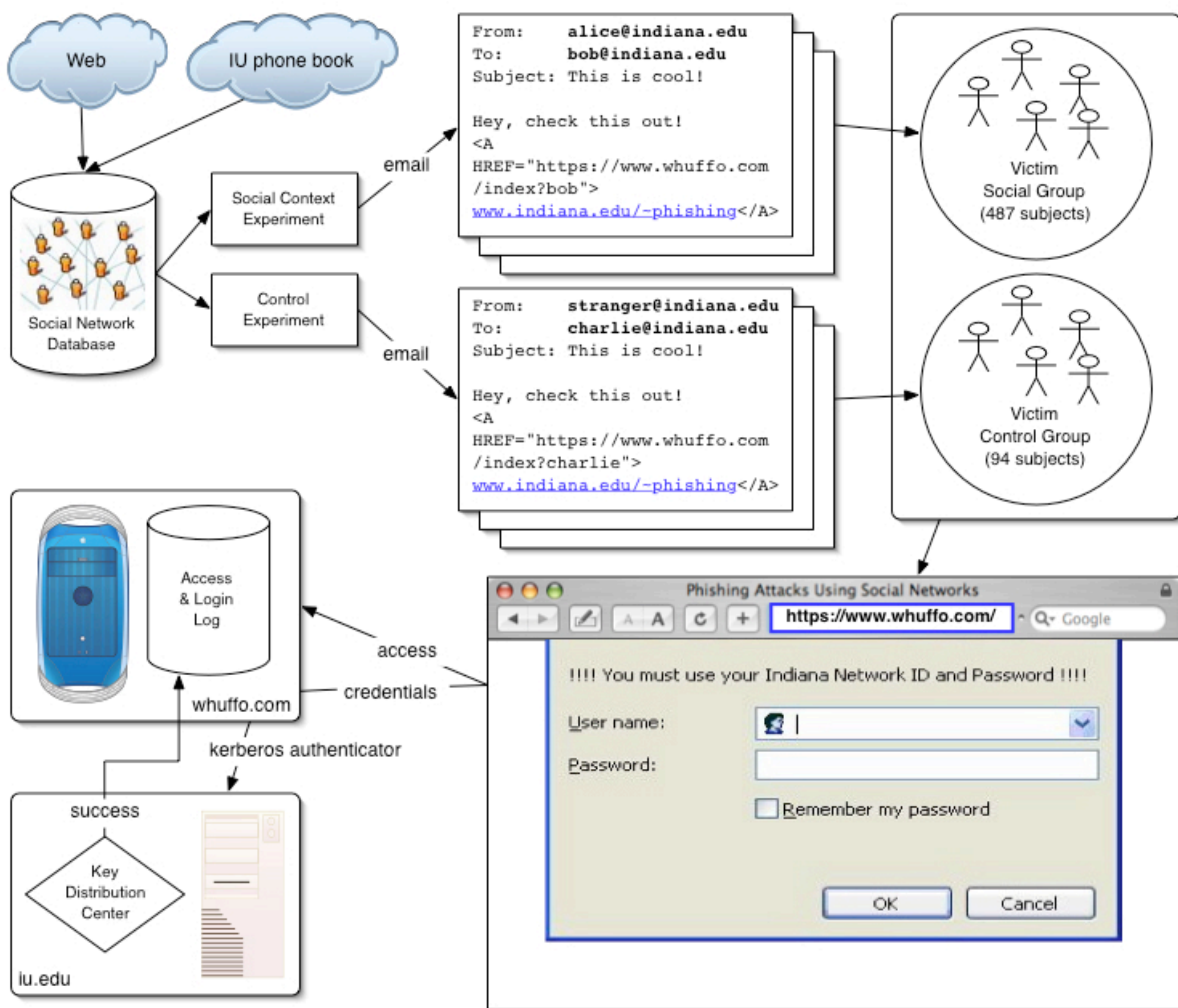
Simple Phishing

Lure: A spammed email with a call to action from a seemingly legitimate source encouraging the user to visit a hook website.

Hook: A website designed to mimic legitimate site and collect confidential information.

Spear Phishing @ IU

Experiment by T. Jagatic, N. Johnson, M. Jakobsson, F. Menczer.



Control Phishing Success Rate:

9-23%

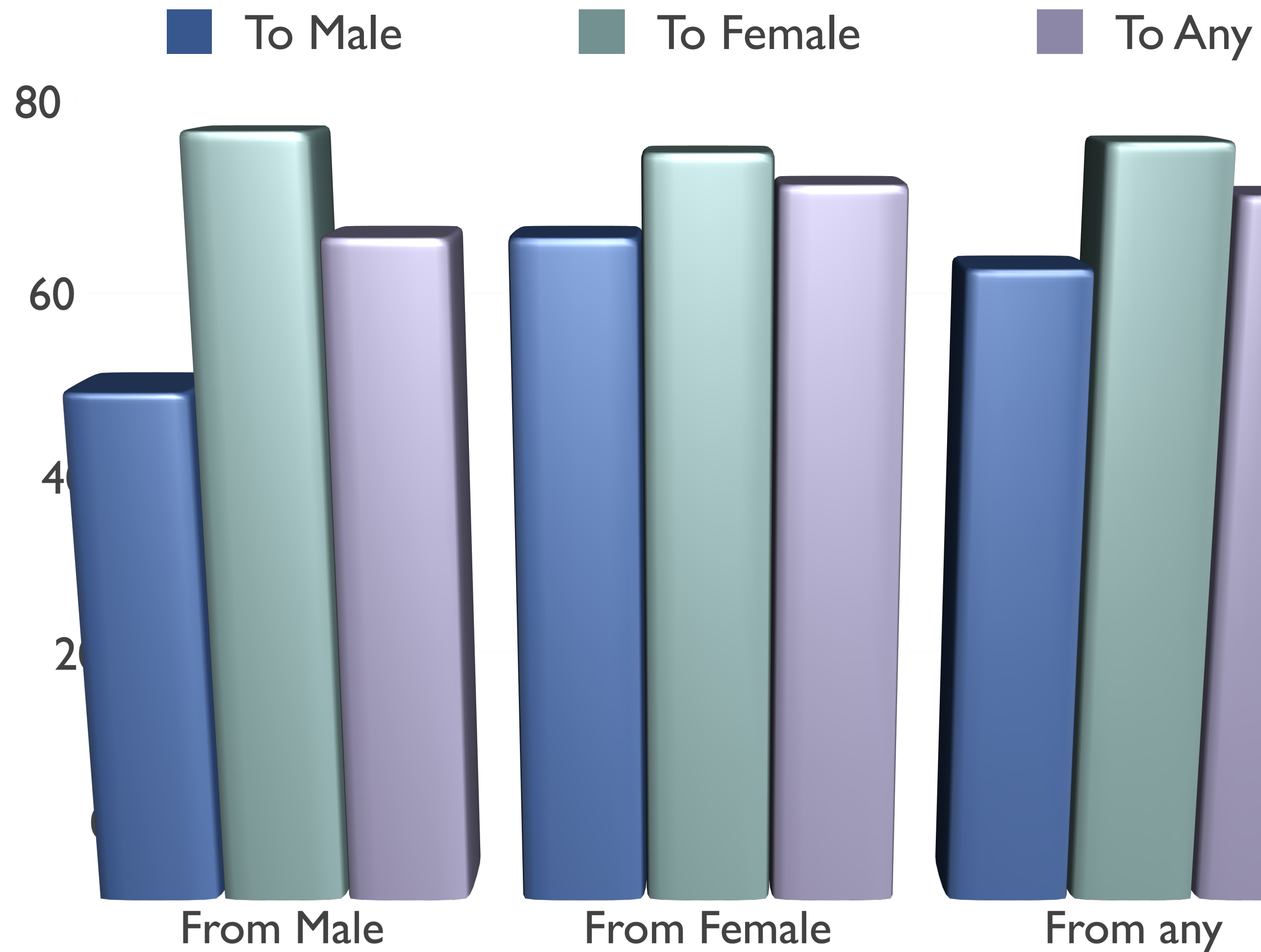
with 95% Confidence Interval

Spear Phishing Success Rate:

68-72%

with 95% Confidence Interval

Spear Phishing Success Rate by Gender



VOIP Phishing

Lure: Get victim to call a bogus 800... number about their account.

Hook: Have the human on the other end extract the victim's information.

From: FlagStar Bank <usflag60536@flagstar.com>

Date: 11 Sep 2007 10:55:21 -0400

To: <samyers@indiana.edu>

Subject: You have one new private message

Dear FlagStar Bank card holder,

You have one new private message.

Please call free 800-870-8124 to listen to your private message.

Copyright ©2007 FlagStar Bank

Source: Steven Myers, IU

From: FlagStar Bank <usflag60536@flagstar.com>

Date: 11 Sep 2007 10:55:21 -0400

To: <samyers@indiana.edu>

Subject: You have one new private message

Dear FlagStar Bank card holder,

You have one new private message.

Please call free 800-870-8124 to listen to your private message.

Copyright ©2007 FlagStar Bank

Source: Steven Myers, IU



Someone has your password

Hi William

Someone just used your password to try to sign in to your Google Account

Details:

Tuesday, 22 March, 14:9:25 UTC

IP Address: 134.249.139.239

Location: Ukraine

Google stopped this sign-in attempt. You should change your password immediately.

[CHANGE PASSWORD](#)

Best,
The Gmail Team

U2F can help prevent this attack



Website
(Relying
Party)

Init

Make a signing key
(sk,pk)

{register}

{register}

{ pk, sign_sk("username") }

User, pk

Login

Sign challenge using sk

{login, challenge ch}

{ s }

U2F can help prevent this attack



Website
(Relying
Party)

Init

Make a signing key
(sk,pk)

{register}

{register}

{ pk, sign_sk("username") }

User, pk

Login

Sign challenge using sk

{login, ch, origin, tls_id}

{login, challenge ch}

$s \leftarrow \text{Sign}_{sk}(ch, \text{url}, \text{tls}_{id})$

{ s }

$\text{Verify}_{pk}(ch, \text{url}, \text{tls}_{id})$

U2F can help prevent tracking

Init

Make a signing key
(sk,pk)



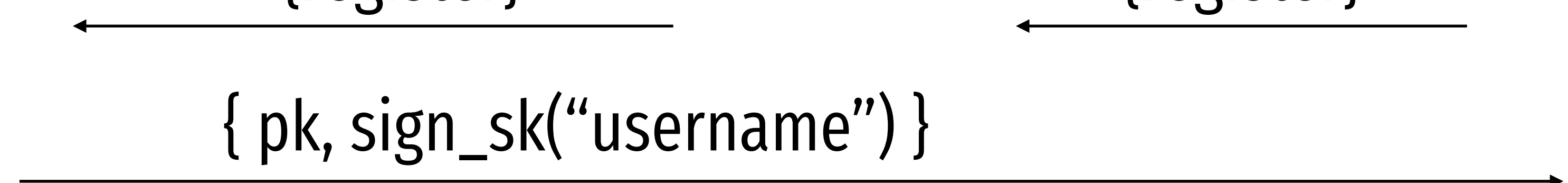
Website
(Relying
Party)

{register}

{register}

{ pk, sign_sk("username") }

User, pk



U2F can help prevent tracking

Init

Make a signing key
(sk,pk)
And link it with
appid, and create
A token "h"



Website
(Relying
Party)

{appid, register}

{appid, register}

{ h, pk, sign_sk("username") }

User, h, pk

U2F can help prevent tracking



Website
(Relying
Party)

Init

Make a signing key
(sk,pk)
And link it with
appid, and create
A token "h"

{appid, register}

{appid, register}

{ h, pk, sign_sk("username") }

User, h, pk

Login

Lookup sk using h
Sign challenge using sk

{login, h, ch, origin, tls_id}

{login, appid, challenge ch}

$s \leftarrow \text{Sign}_{sk}(ch, url, tls_{id})$

{ s, h }

Verify_{pk}(ch, url, tls_{id})
Check h

Sending request with appId: https://u2f.bin.coffee

```
{  
  "version": "U2F_V2",  
  "challenge": "uQnl3M4Rj3FZgs6WjyLaZAfwRh4"  
}
```

Got response:

```
{  
  "clientData": "eyJjaGFsbGVuZ2UiOiJlUW5sM000UmozRlpnczZXanlMYVpBZndSaDQiLCJvcmlnaW4iOiJodHRwczovL3UyZi5iaW4uY29mZmVlIiwidHlwIjoibmF2",  
  "errorCode": 0,  
  "registrationData": "BQRSuRLPv0p5udQ55vVhucf3N50q6...",  
  "version": "U2F_V2"  
}
```

Key Handle: 0r0Z0p0F0E0-0d0W0c0Q0b0X0i020C0w0-0E0v0h0t0T0T0P0_0-090_0a050P0e030u0b0z0l0K0Q0r000f0u030_0P020B0J0M0x0D050J0_0d0P0Q0e0j0

Certificate: 3082021c3082...

Attestation Cert

Subject: Yubico U2F EE Serial 14803321578

Issuer: Yubico U2F Root CA Serial 457200631

Validity (in millis): 1136332800000

Attestation Signature

R: 00b11e3efe5ae5ac7ca0e0d4fe2c5b5cf18a2531c0f4f70b11c30b72b5f946a9a3

S: 0f37ab2d4f93ebcdaed0a51b4b17fb93403db9873f0e9cce36f17b1502734bb2

[PASS] Signature buffer has no unnecessary bytes.: 71 == 71

[PASS] navigator.id.finishEnrollment == navigator.id.finishEnrollment

[PASS] uQnl3M4Rj3FZgs6WjyLaZAfwRh4 == uQnl3M4Rj3FZgs6WjyLaZAfwRh4

[PASS] https://u2f.bin.coffee == https://u2f.bin.coffee

[PASS] Verified certificate attestation signature

[PASS] Imported credential public key

Failures: 0 TODOs: 0

Future without passwords?

Authentication Protocols

Unix, PAM, and crypt

Network Information Service (NIS, aka Yellow Pages)

Needham-Schroeder and Kerberos

Status Check

- At this point, we have discussed:
 - How to securely store passwords
 - Techniques used by attackers to crack passwords
 - Biometrics and 2nd factors

Status Check

- At this point, we have discussed:
 - How to securely store passwords
 - Techniques used by attackers to crack passwords
 - Biometrics and 2nd factors
- Next topic: building authentication systems
 - Given a user and password, how does the system authenticate the user?
 - How can we perform efficient, secure authentication in a distributed system?

Building authentication systems

Example PAM Configuration

```
# cat /etc/pam.d/system-auth
#%PAM-1.0
```

```
auth required pam_unix.so try_first_pass
auth optional pam_permit.so
auth required pam_env.so
```

```
account required pam_unix.so
account optional pam_permit.so
account required pam_time.so
```

```
password required pam_unix.so try_first_pass nullok sha512 shadow
password optional pam_permit.so
```

```
session required pam_limits.so
session required pam_unix.so
session optional pam_permit.so
```

- Use SHA512 as the hash function
- Use /etc/shadow for storage

Unix Passwords

- Traditional method: *crypt*
 - 25 iterations of DES on a zeroed vector
 - First eight bytes of password used as key (additional bytes are ignored)
 - 12-bit salt
- Modern version of *crypt* are more extensible
 - Support for additional hash functions like MD5, SHA256, and SHA512
 - Key lengthening: defaults to 5000 iterations, up to $10^8 - 1$
 - Full password used
 - Up to 16 bytes of salt

Password Files

- Password hashes used to be in */etc/passwd*
 - World readable, contained usernames, password hashes, config information
 - Many programs read config info from the file...
 - But very few (only one?) need the password hashes

Password Files

- Password hashes used to be in */etc/passwd*
 - World readable, contained usernames, password hashes, config information
 - Many programs read config info from the file...
 - But very few (only one?) need the password hashes
- Turns out, world-readable hashes are **Bad Idea**
- Hashes now located in */etc/shadow*
 - Also includes account metadata like expiration
 - Only visible to root

Password Storage on Linux

/etc/passwd

username:x:UID:GID:full_name:home_directory:shell

cbw:x:1001:1000:Christo Wilson:/home/cbw/#!/bin/bash

amislove:1002:2000:Alan Mislove:/home/amislove/#!/bin/sh

/etc/shadow

username:password:last:may:must:warn:expire:disable:reserved

cbw:\$1\$0nSd5ewF\$0df/3G7iSV49nsbAa/5gSg:9479:0:10000:::

amislove:\$1\$I3RxU5F1\$:8172:0:10000:::

Password Storage on Linux

/etc/passwd

username:x:UID:GID:full_name:home_directory:shell

cbw:x:1001:1000:Christo Wilson:/home/cbw/#!/bin/bash

n Mislove:/home/amislove/#!/bin/sh

\$<algo>\$<salt>\$<hash>

Algo: 1 = MD5, 5 = SHA256, 6 = SHA512

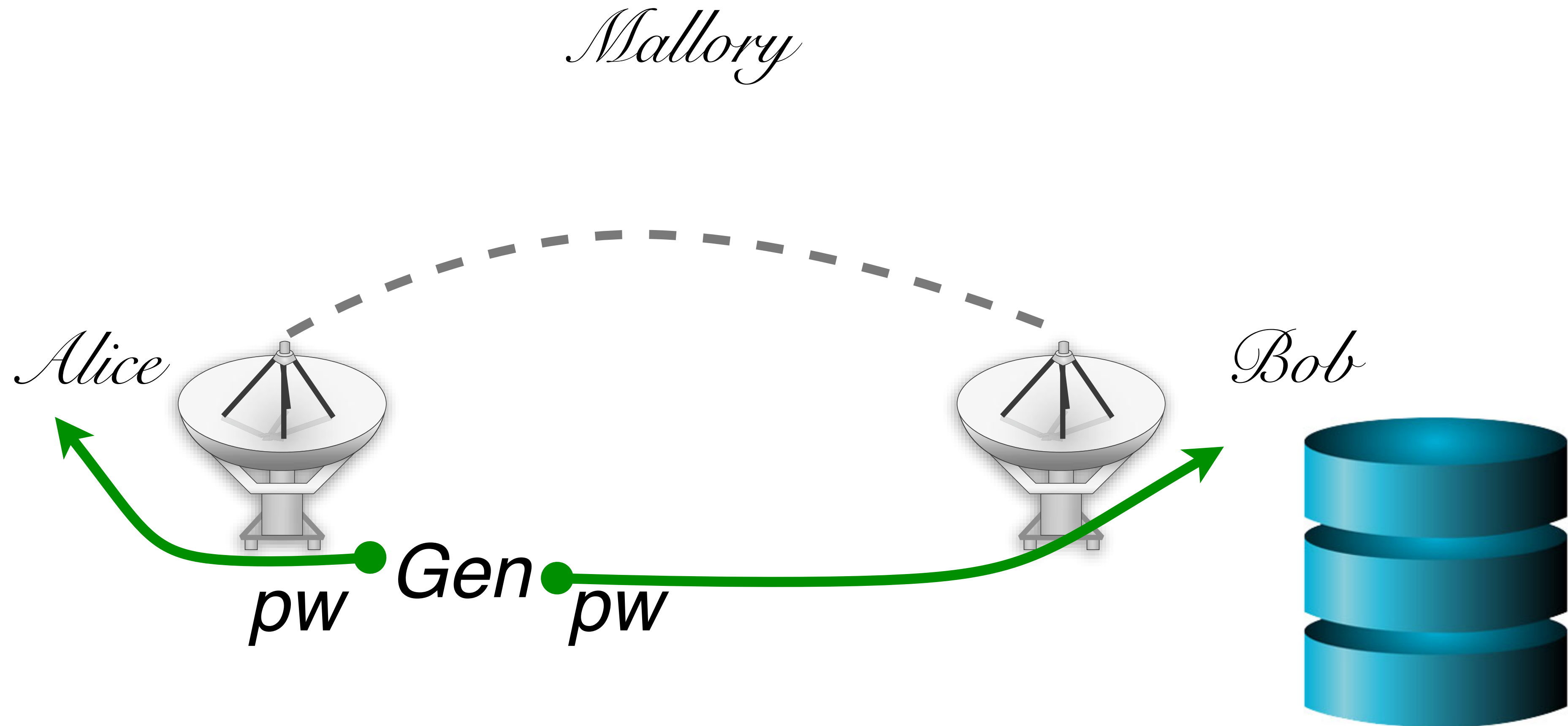
/etc/shadow

username:password:last:may:must:warn:expire:disable:reserved

cbw:\$1\$0nSd5ewF\$0df/3G7iSV49nsbAa/5gSg:9479:0:10000:::

amislove:\$1\$I3RxU5F1\$:8172:0:10000:::

Password Security game

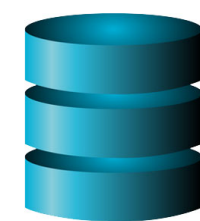
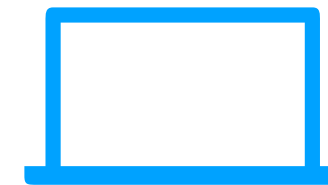


More realistic picture of the world

Alice
pw



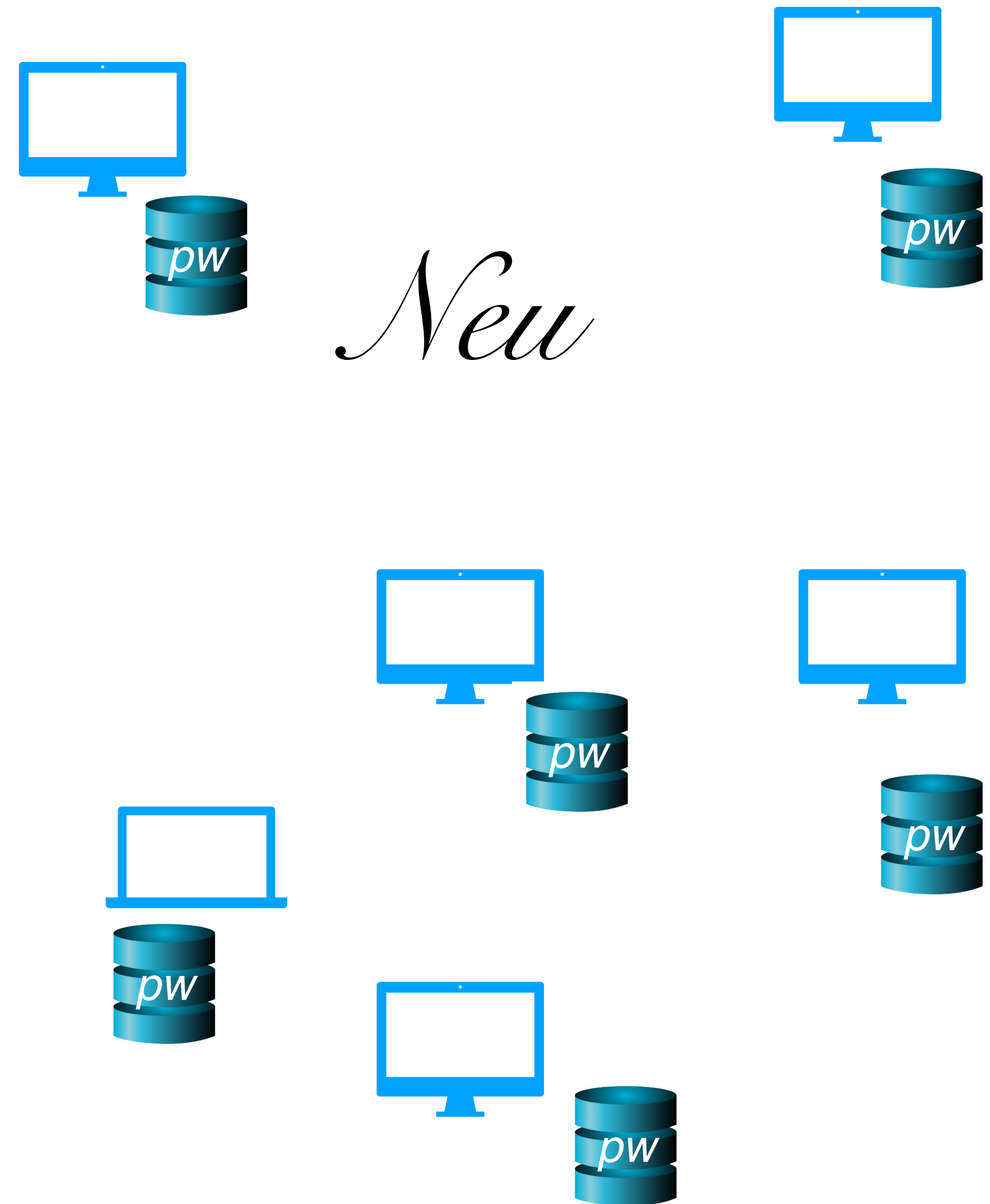
Neu



More realistic picture of the world

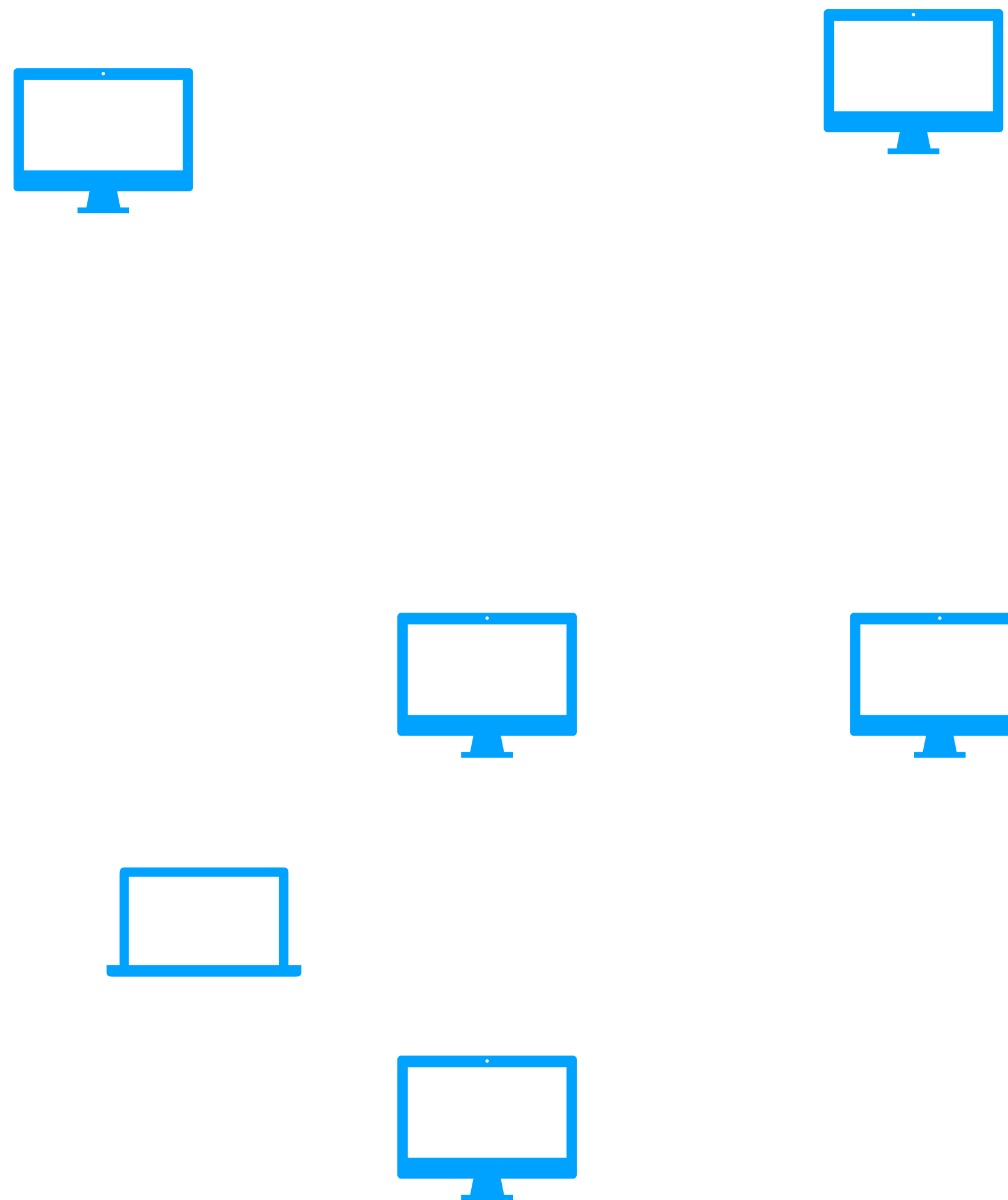
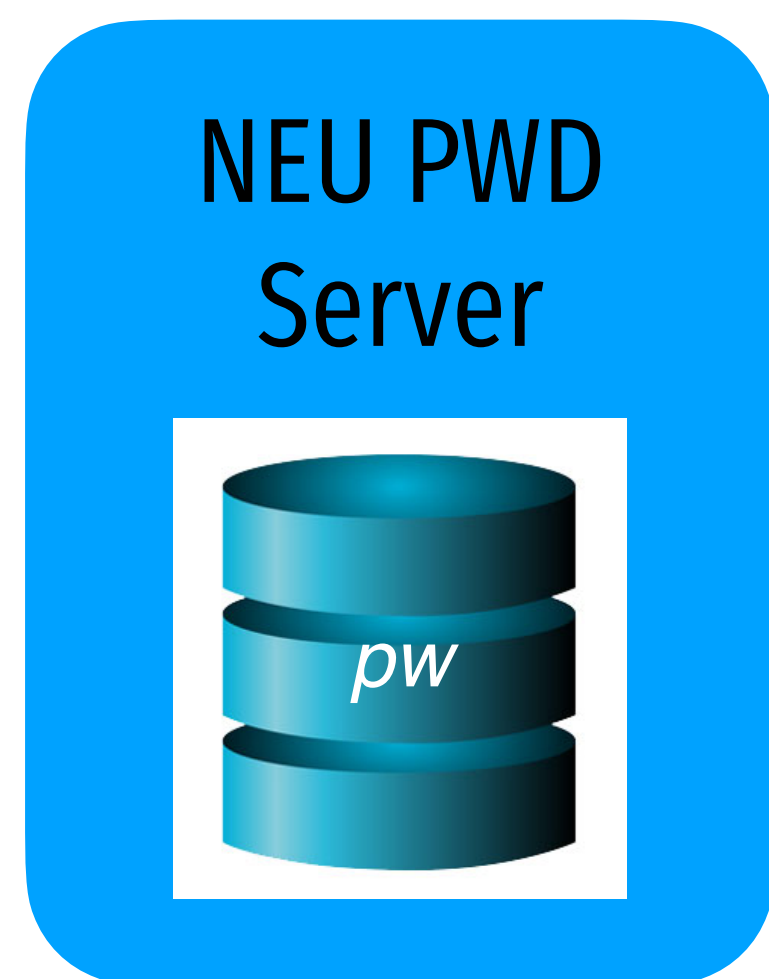
What are the problems with this solution?

Alice
pw



The problem of distributed authentication

Alice
pw

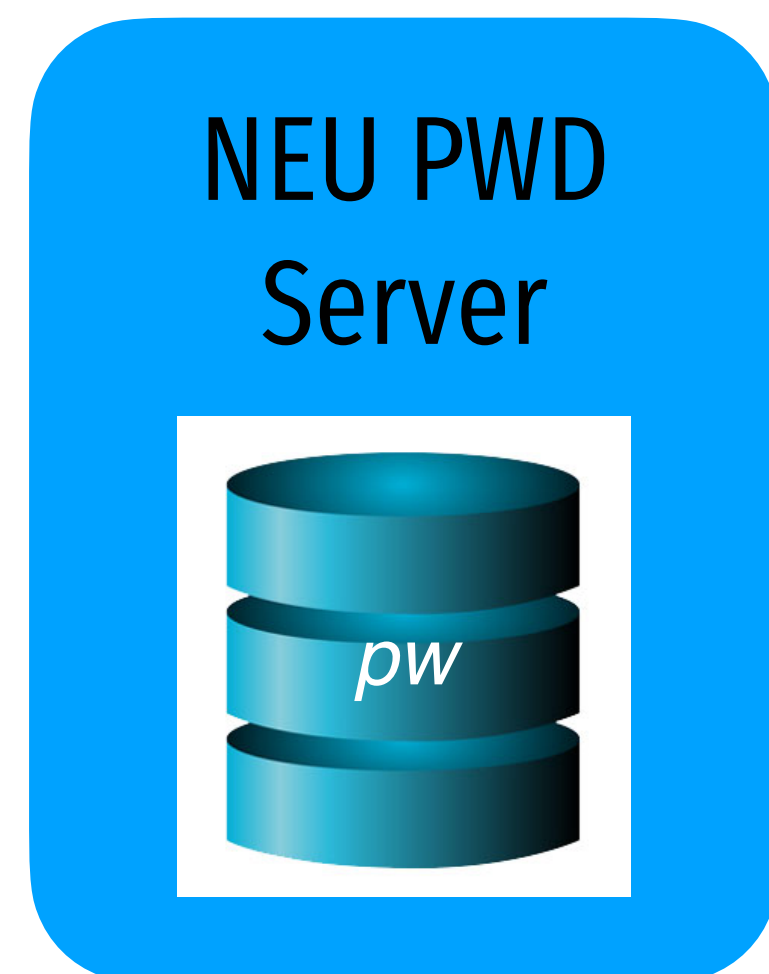
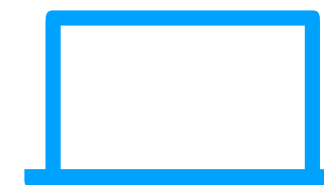


Distributed authentication: Attacker model

What can attacker do?

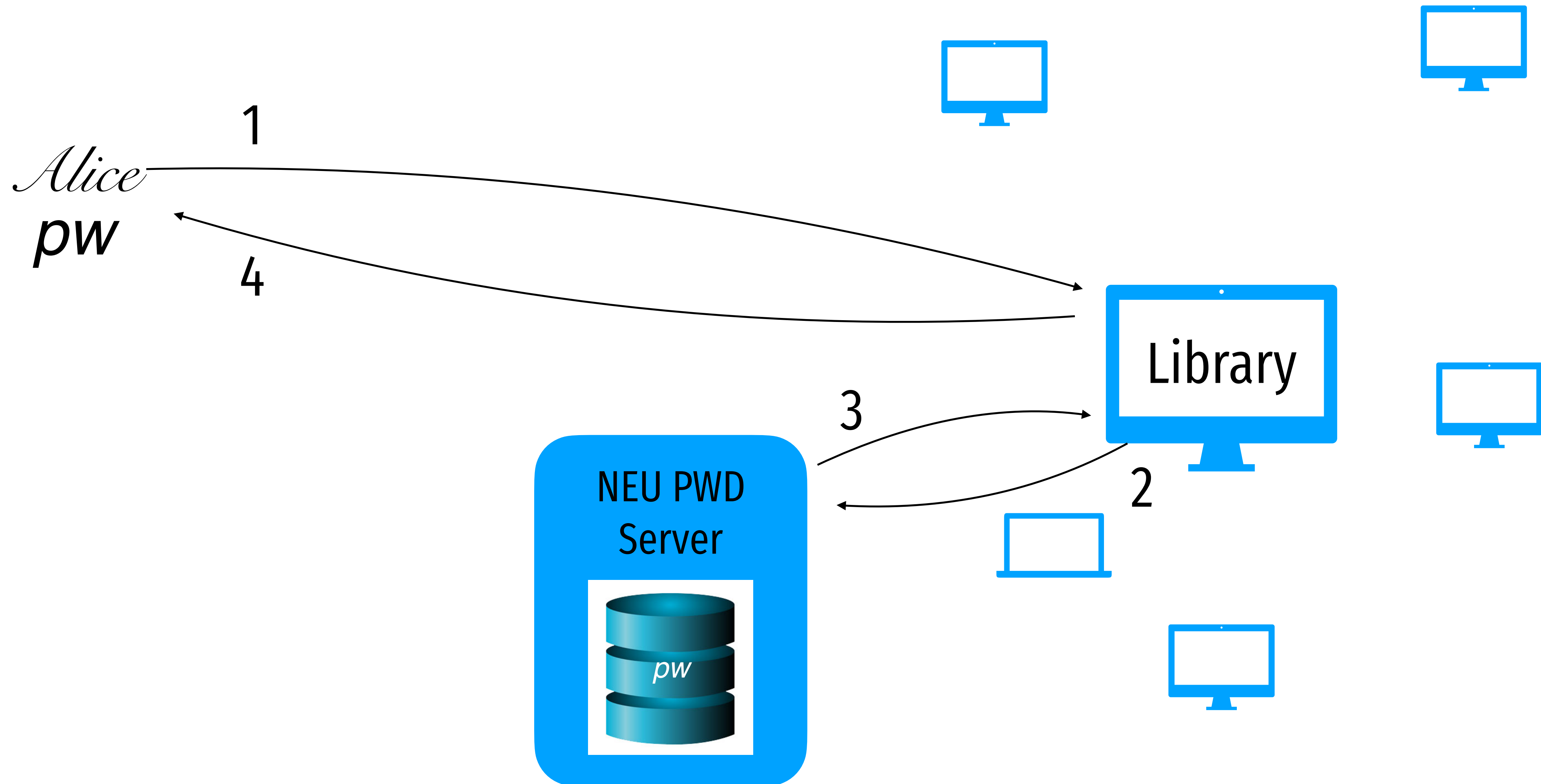


Alice
pw



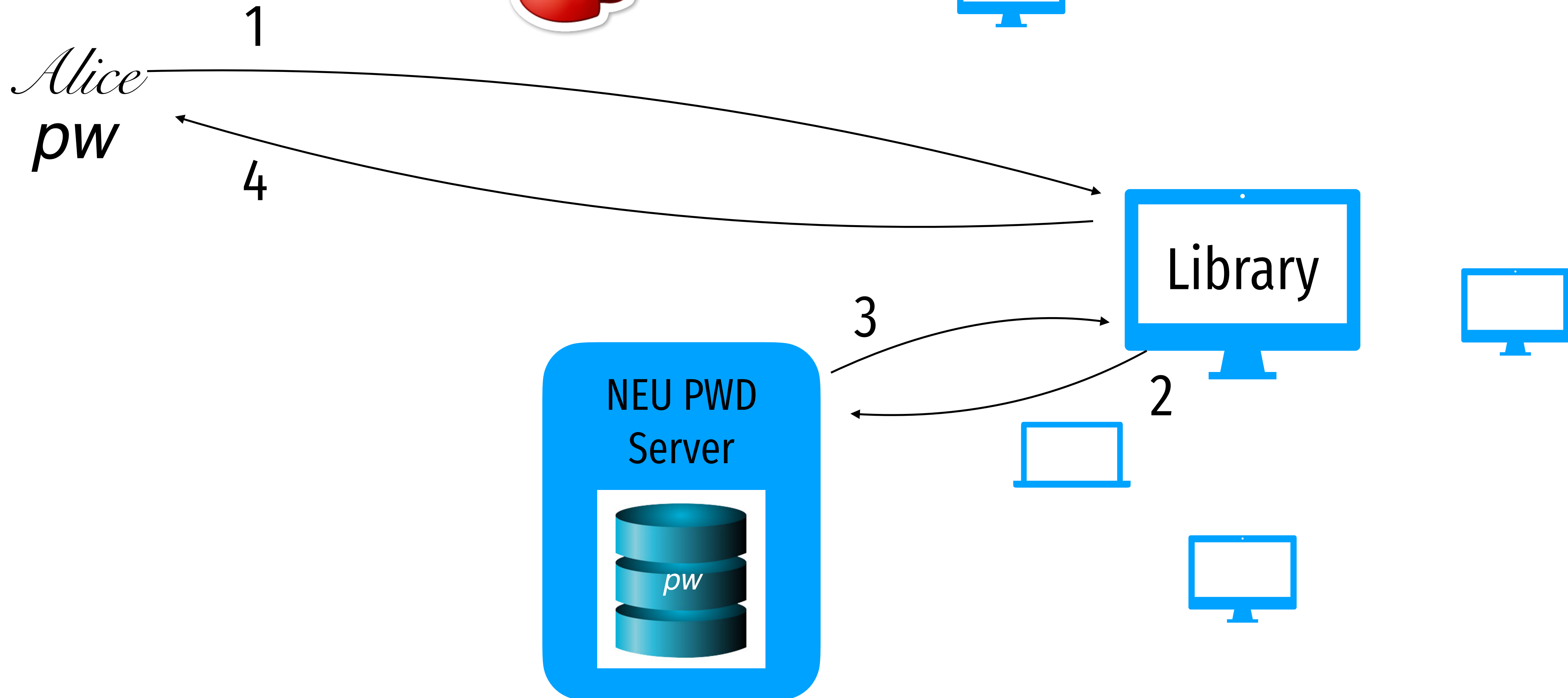
Distributed authentication: Bad Solution

What can attacker do?



Distributed authentication: Bad Solution

What can attacker do?



Needham-Schroeder Protocol

- Let Alice A and Bob B be two parties that trust server S
- K_{AS} and K_{BS} are shared secrets between $[A, S]$ and $[B, S]$
- K_{AB} is a negotiated session key between $[A, B]$
- N_i and N_j are random **nonces** generated by A and B

1) $A \rightarrow S: A, B, N_i$

2) $S \rightarrow A: \{N_i, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

3) $A \rightarrow B: \{K_{AB}, A\}_{K_{BS}}$

4) $B \rightarrow A: \{N_j\}_{K_{AB}}$

5) $A \rightarrow B: \{N_j - 1\}_{K_{AB}}$

Needham-Schroeder Protocol

- Let Alice A and Bob B be two parties that trust server S
- K_{AS} and K_{BS} are shared secrets between $[A, S]$ and $[B, S]$
- K_{AB} is a negotiated session key between $[A, B]$
- N_i and N_j are random **nonces** generated by A and B

1) $A \rightarrow S: A, B, N_i$

2) $S \rightarrow A: \{N_i, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

3) $A \rightarrow B: \{K_{AB}, A\}_{K_{BS}}$

4) $B \rightarrow A: \{N_j\}_{K_{AB}}$

5) $A \rightarrow B: \{N_j - 1\}_{K_{AB}}$

Challenge nonce forces A to acknowledge they have K_{AB}

Needham-Schroeder Protocol

- Let Alice A and Bob B be two parties that trust server S
- K_{AS} and K_{BS} are shared secrets between $[A, S]$ and $[B, S]$
- K_{AB} is a negotiated session key between $[A, B]$
- N_i and N_j are random **nonces** generated by A and B

1) $A \rightarrow S: A, B, N_i$

K_{AS} is not sent in the clear, authenticates S and A

2) $S \rightarrow A: \{N_i, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

3) $A \rightarrow B: \{K_{AB}, A\}_{K_{BS}}$

4) $B \rightarrow A: \{N_j\}_{K_{AB}}$

5) $A \rightarrow B: \{N_j - 1\}_{K_{AB}}$

Challenge nonce forces A to acknowledge they have K_{AB}

Needham-Schroeder Protocol

- Let Alice A and Bob B be two parties that trust server S
- K_{AS} and K_{BS} are shared secrets between $[A, S]$ and $[B, S]$
- K_{AB} is a negotiated session key between $[A, B]$
- N_i and N_j are random **nonces** generated by A and B

1) $A \rightarrow S: A, B, N_i$

K_{AS} is not sent in the clear, authenticates S and A

2) $S \rightarrow A: \{N_i, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

3) $A \rightarrow B: \{K_{AB}, A\}_{K_{BS}}$

K_{BS} is not sent in the clear, authenticates B

4) $B \rightarrow A: \{N_j\}_{K_{AB}}$

5) $A \rightarrow B: \{N_j - 1\}_{K_{AB}}$

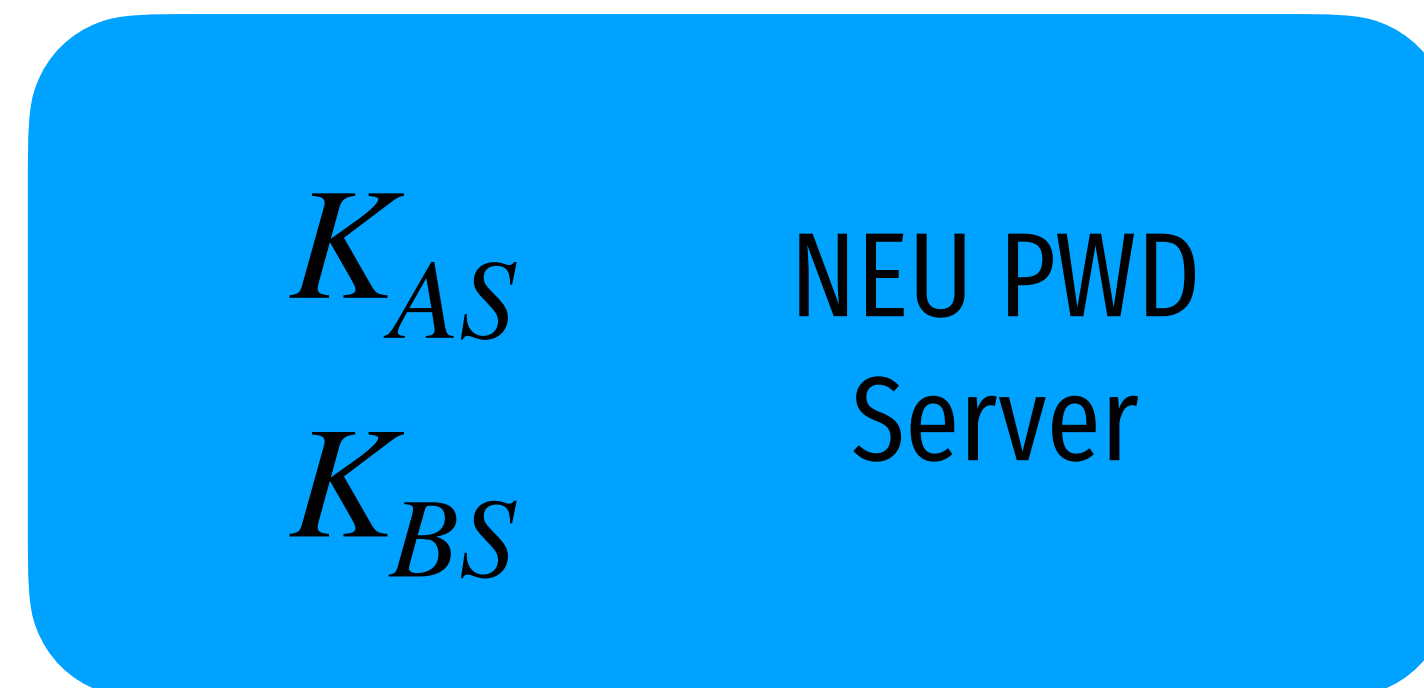
Challenge nonce forces A to acknowledge they have K_{AB}

Notorious Needham-Schroeder Protocol

Goal: K_{AB}

K_{AS} *Alice*

Bob K_{BS}



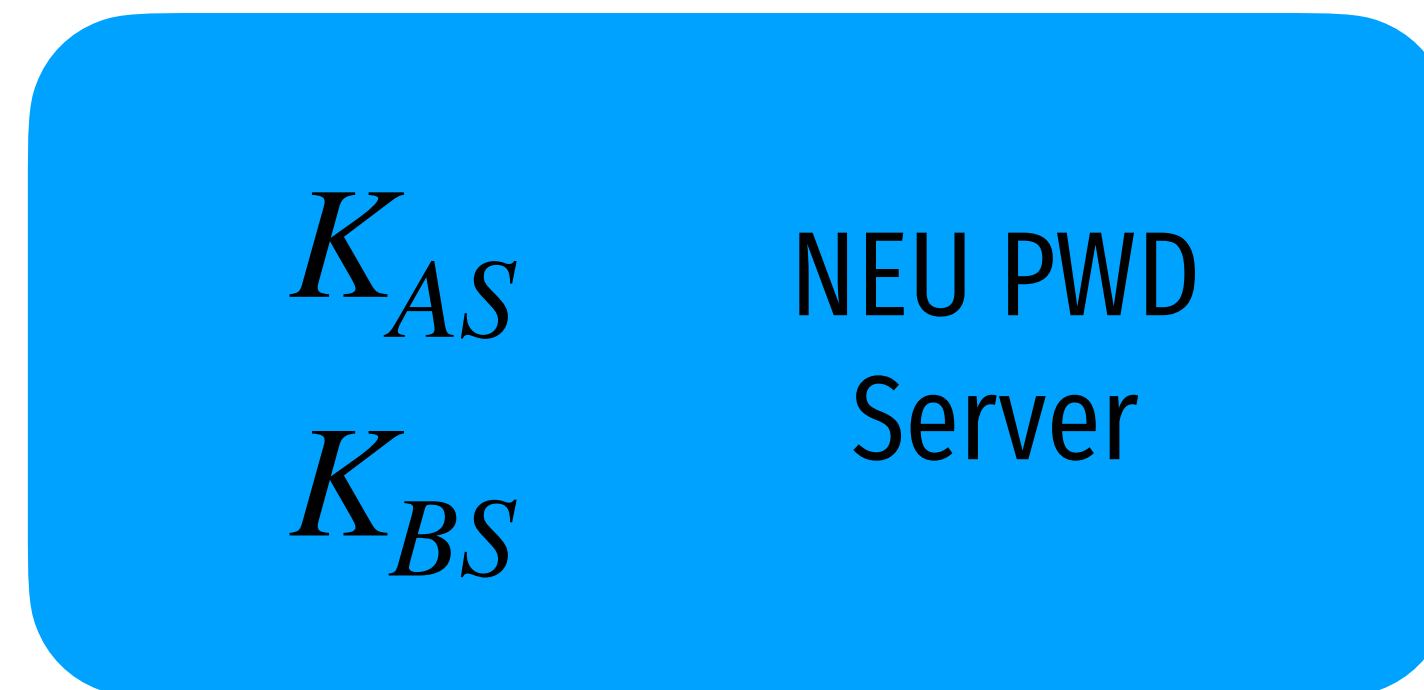
Notorious Needham-Schroeder Protocol

Goal: K_{AB}

K_{AS} *Alice*

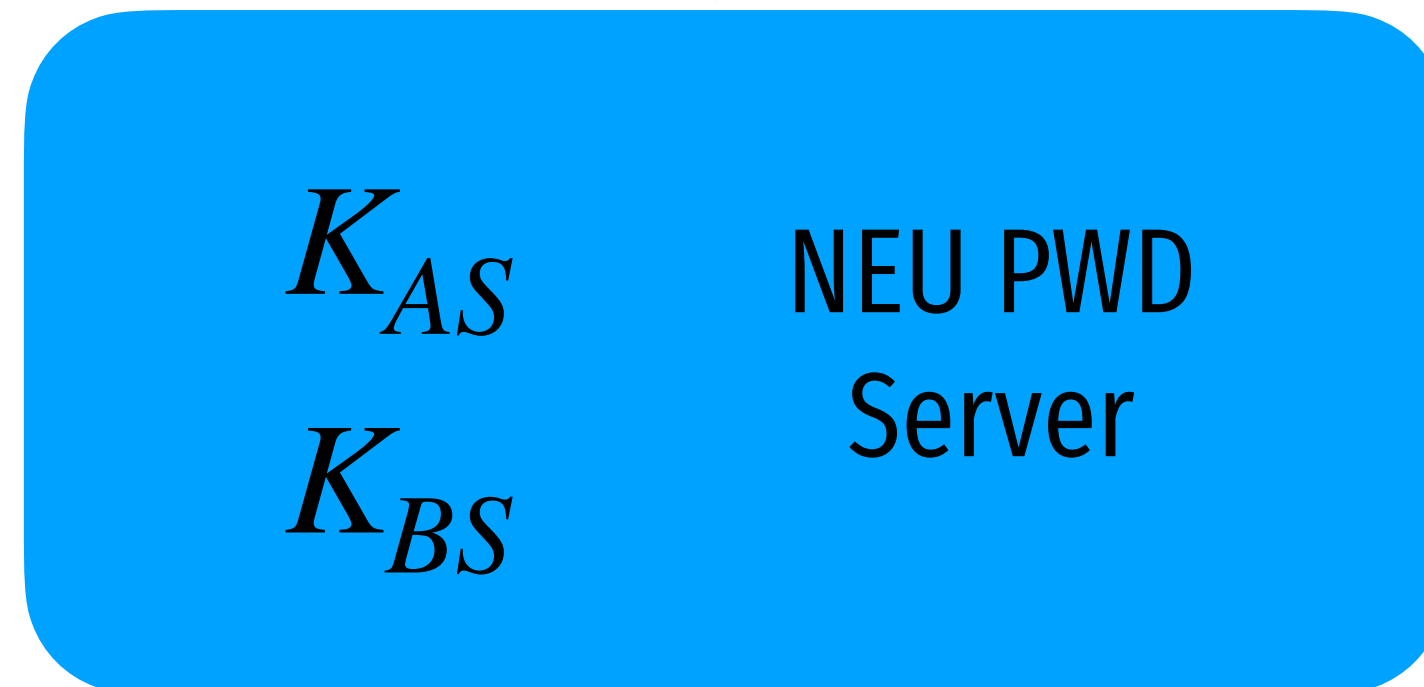
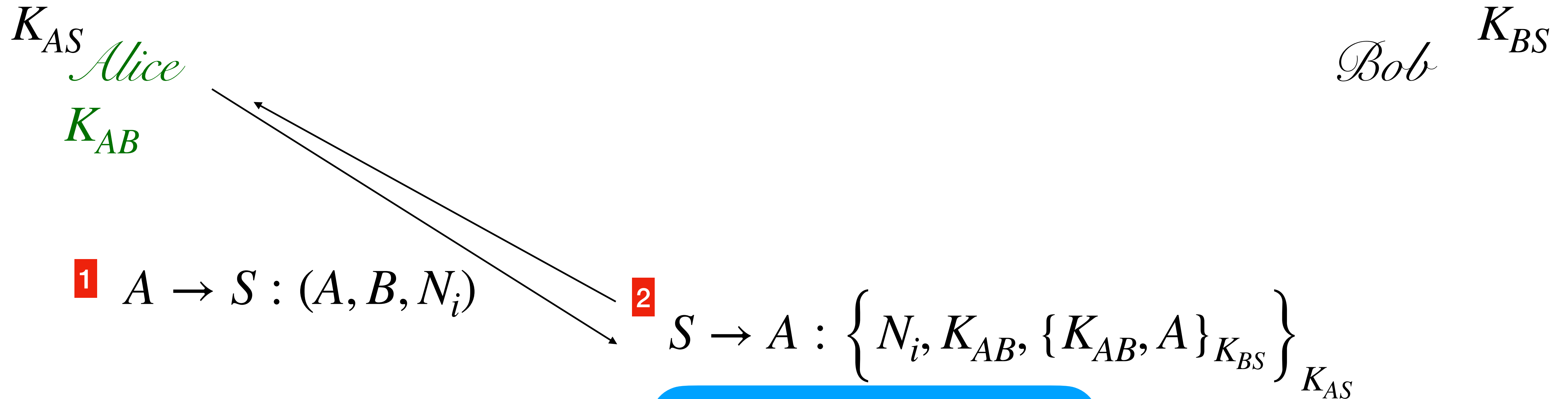
Bob K_{BS}

1 $A \rightarrow S : (A, B, N_i)$



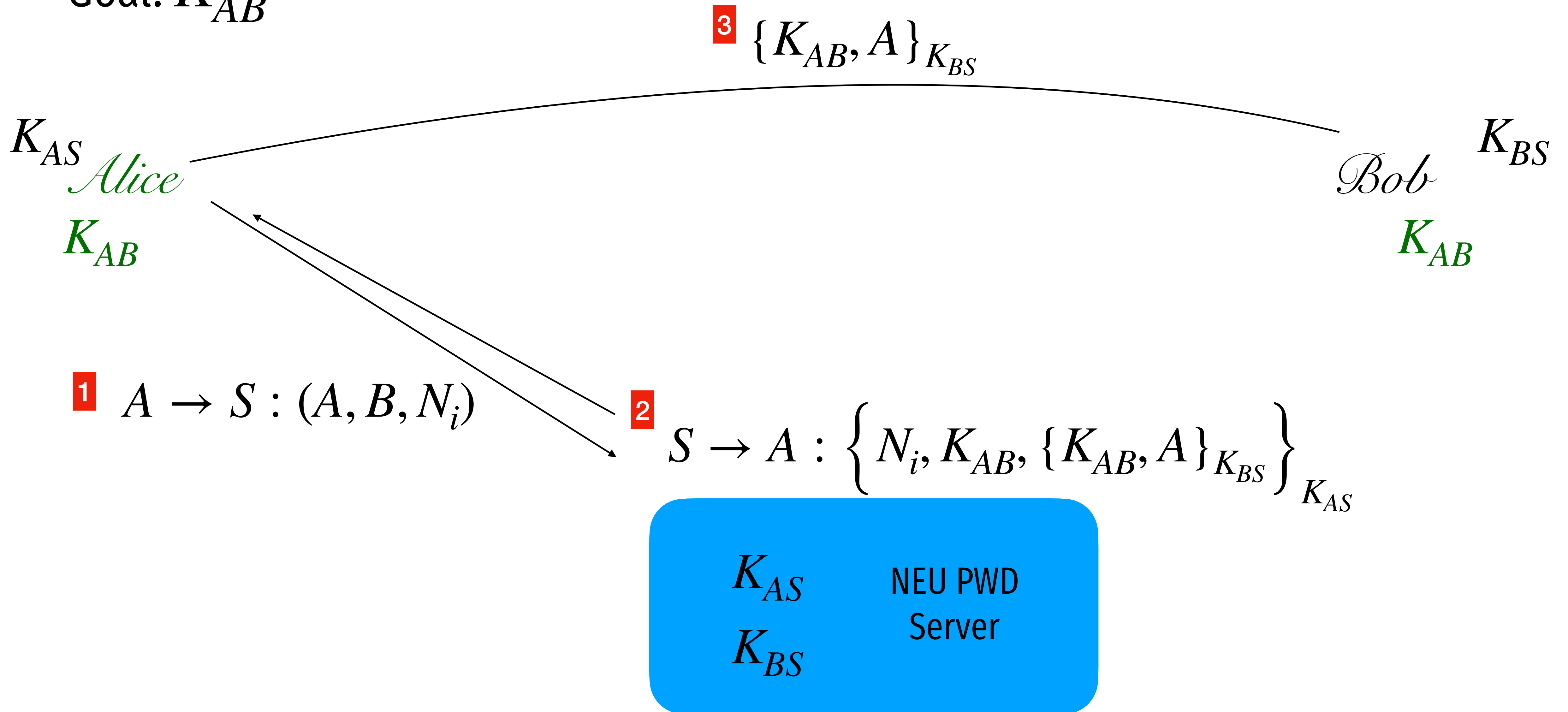
Notorious Needham-Schroeder Protocol

Goal: K_{AB}



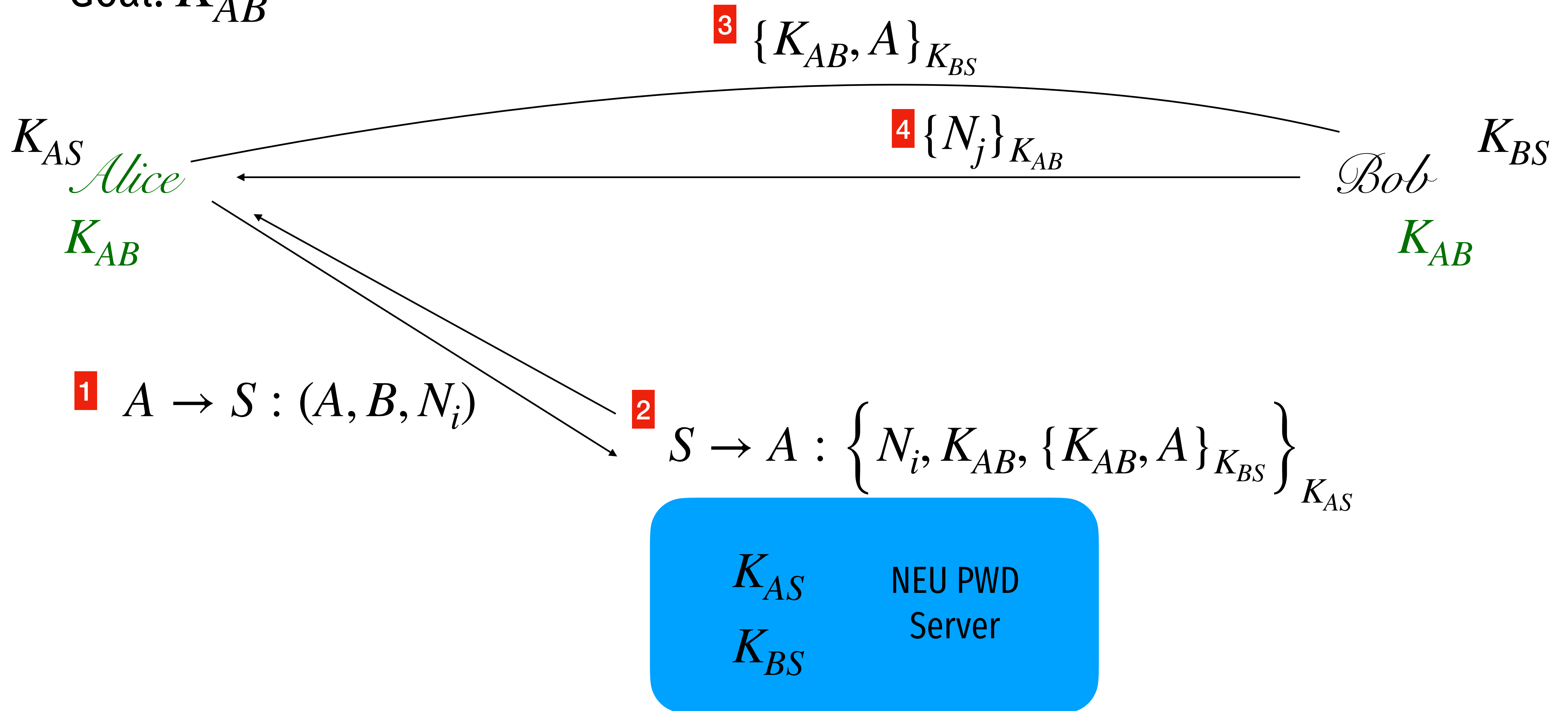
Notorious Needham-Schroeder Protocol

Goal: K_{AB}



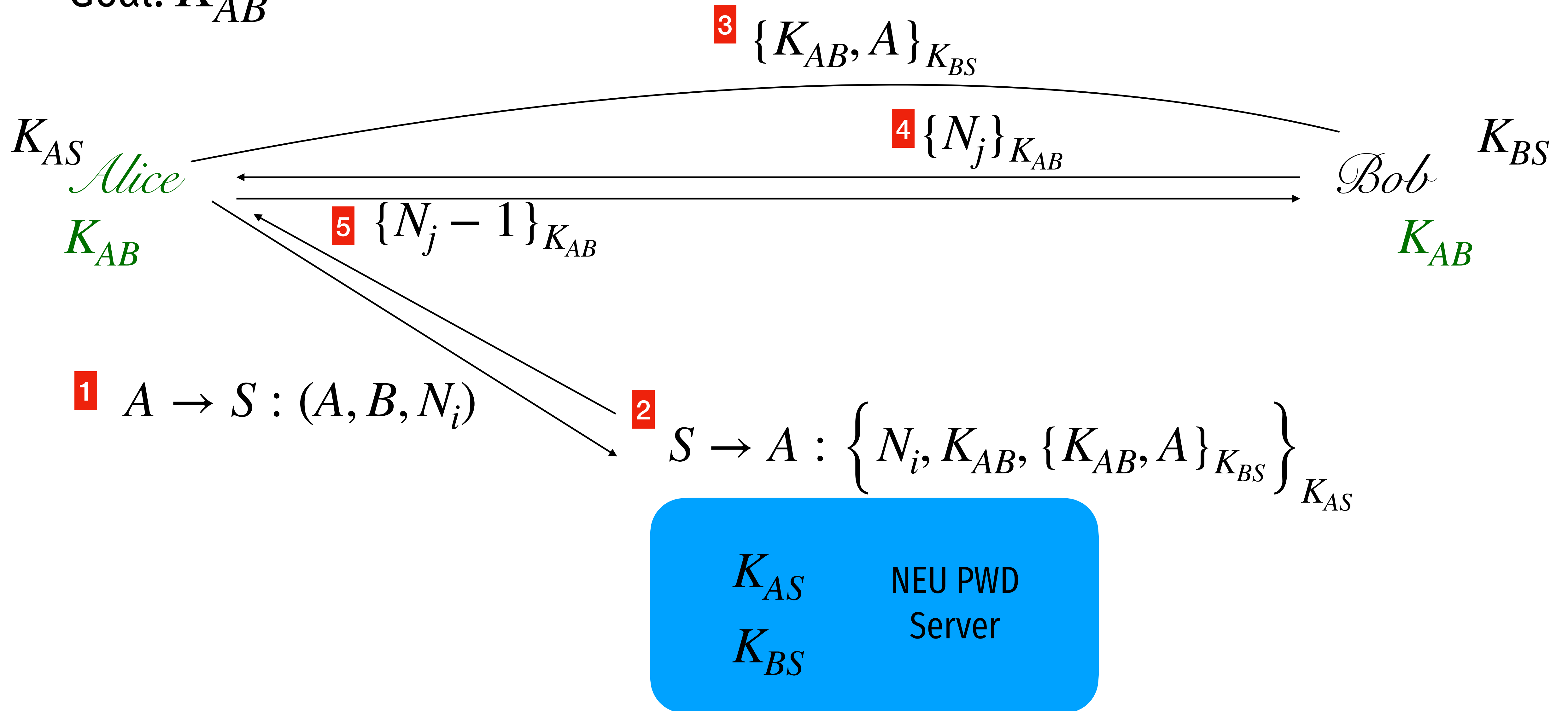
Notorious Needham-Schroeder Protocol

Goal: K_{AB}



Notorious Needham-Schroeder Protocol

Goal: K_{AB}



Notorious Needham-Schroeder Protocol

Goal: K_{AB}

Suppose attacker tries to impersonate Alice

K_{AS} Alice



Bob K_{BS}

1 $A \rightarrow S : (A, B, N_i)$

2 $S \rightarrow A : \left\{ N_i, K_{AB}, \{ K_{AB}, A \}_{K_{BS}} \right\}_{K_{AS}}$

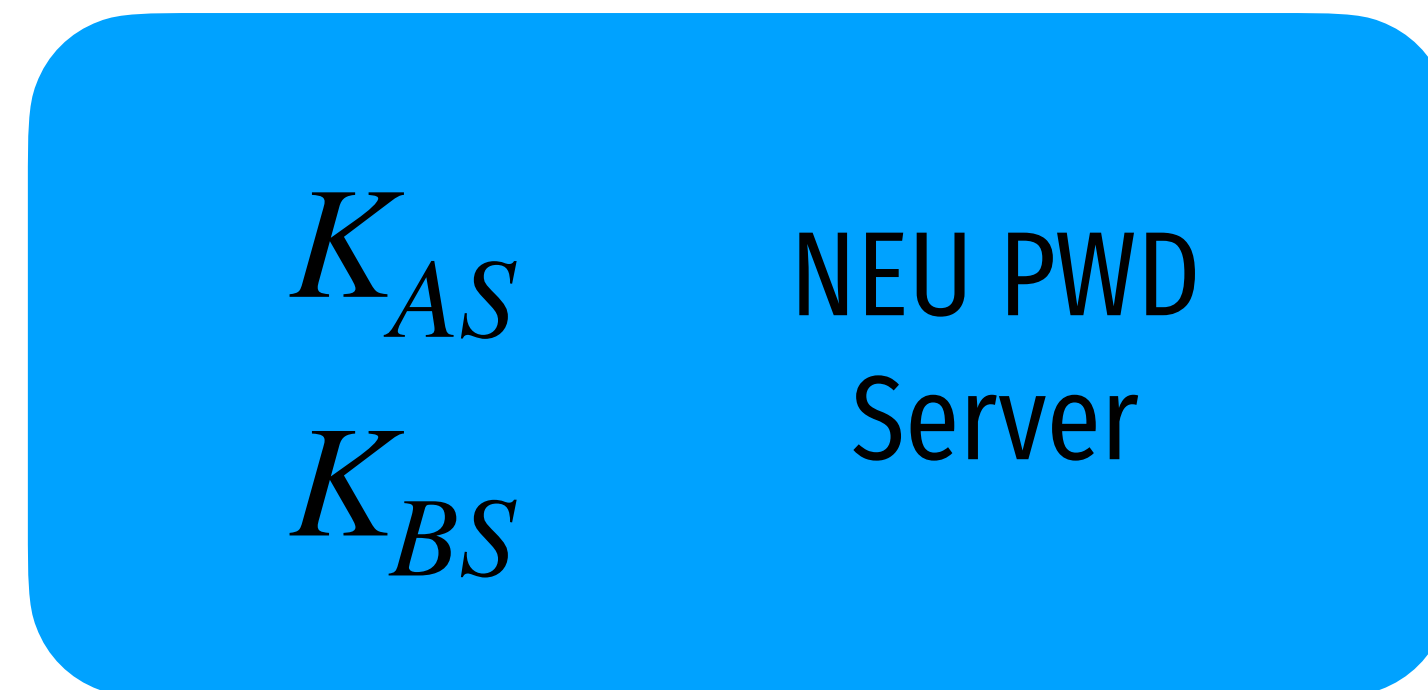
K_{AS} NEU PWD
 K_{BS} Server

Notorious Needham-Schroeder Protocol

Goal: K_{AB}

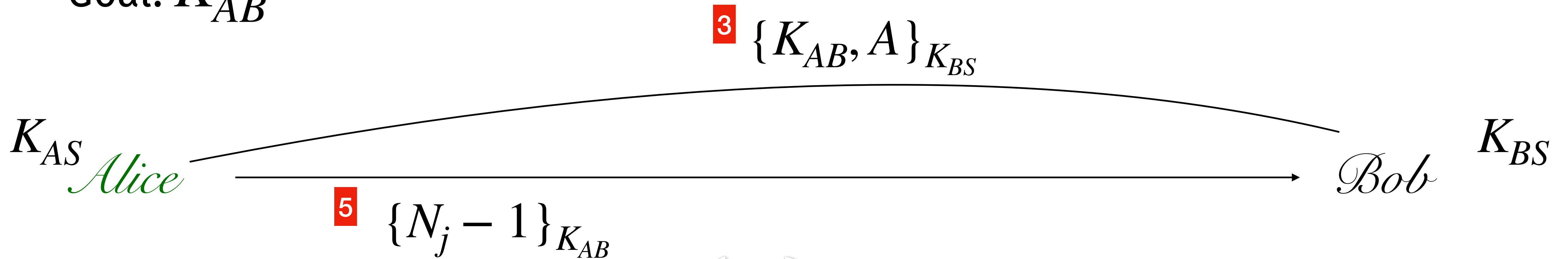


Suppose
attacker tries to
impersonate
Alice to Bob



Needham-Schroeder Replay Attack

Goal: K_{AB}



Protocol runs once. Attacker observes.

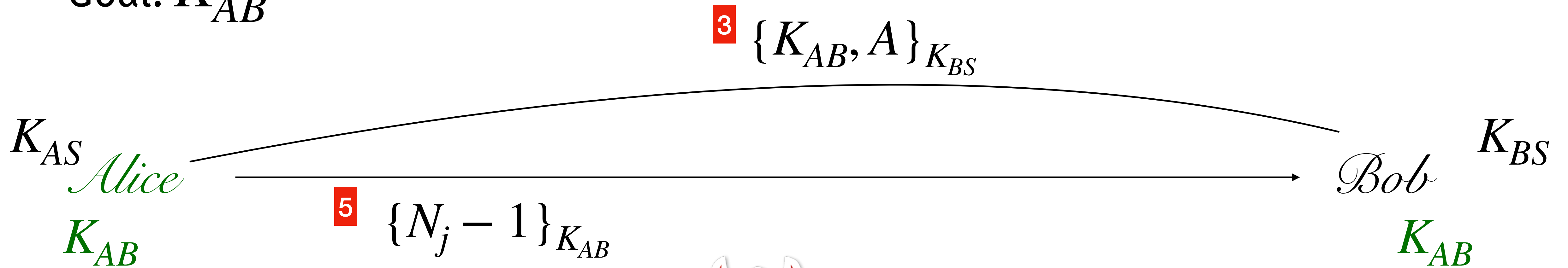


$\{K_{AB}, A\}_{K_{BS}}$
 $\{N_j - 1\}_{K_{AB}}$

K_{AS} NEU PWD
 K_{BS} Server

Needham-Schroeder Replay Attack

Goal: K_{AB}



Protocol runs once. Attacker observes.

K_{AS}
 K_{BS}
NEU PWD
Server

Needham-Schroeder Replay Attack

Goal: K_{AB}

K_{AS} Alice
 K_{AB}

Bob K_{BS}
 K_{AB}



Protocol runs once. Attacker observes.

$\{K_{AB}, A\}_{K_{BS}}$
 $\{N_j - 1\}_{K_{AB}}$

K_{AS}

NEU PWD

K_{BS}

Server

Needham-Schroeder Replay Attack

Goal: K_{AB}

K_{AS} Alice
 K_{AB}

Bob K_{BS}
 K_{AB}

Protocol runs once. Attacker observes.
Attacker breaks into Alice and steals old K_{AB} .



K_{AB}

$\{K_{AB}, A\}_{K_{BS}}$
 $\{N_j - 1\}_{K_{AB}}$

K_{AS}

NEU PWD
Server

K_{BS}

Needham-Schroeder Replay Attack

Goal: K_{AB}

K'_{AS}
Alice
 K_{AB}

Bob K_{BS}
 K_{AB}

Protocol runs once. Attacker observes.
Attacker breaks into Alice and steals old K_{AB} .
Alice updates K_{AS} .



K_{AB} $\{K_{AB}, A\}_{K_{BS}}$
 $\{N_j - 1\}_{K_{AB}}$

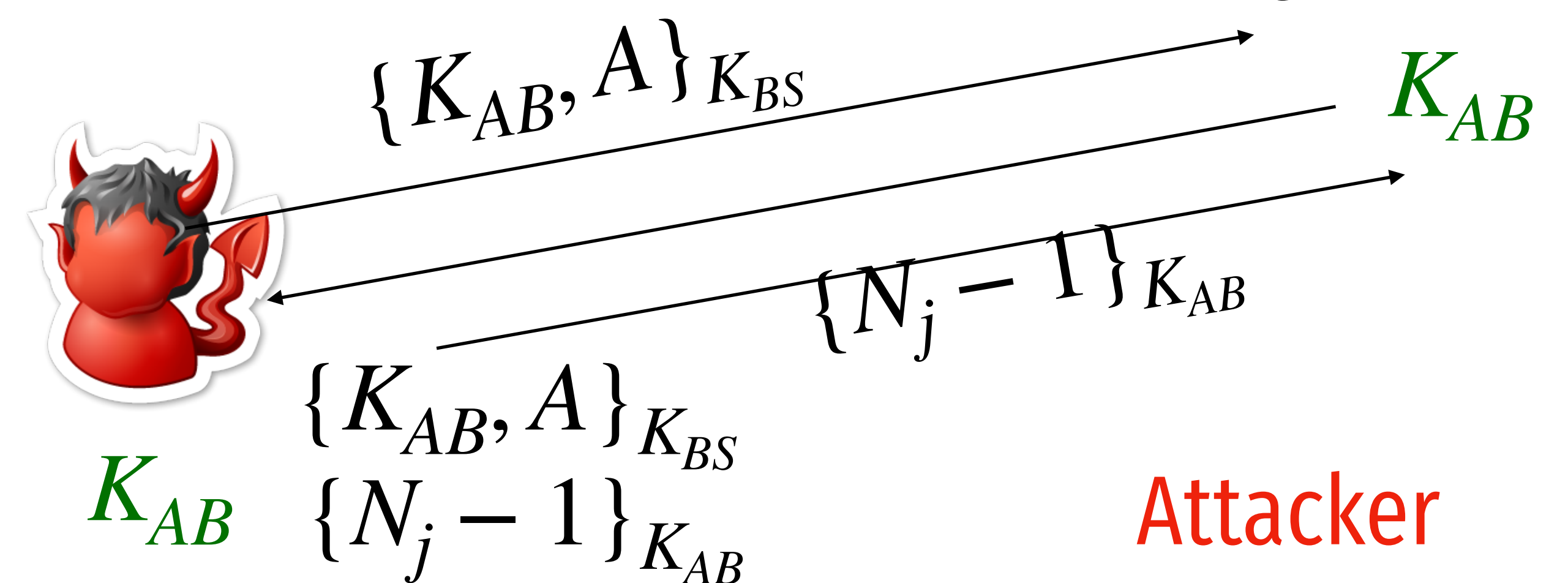
K_{AS} NEU PWD Server
 K_{BS}

Needham-Schroeder Replay Attack

Goal: K_{AB}

K'_{AS}
Alice
 K_{AB}

Bob K_{BS}



Protocol runs once. Attacker observes.
Attacker breaks into Alice and steals old K_{AB} .

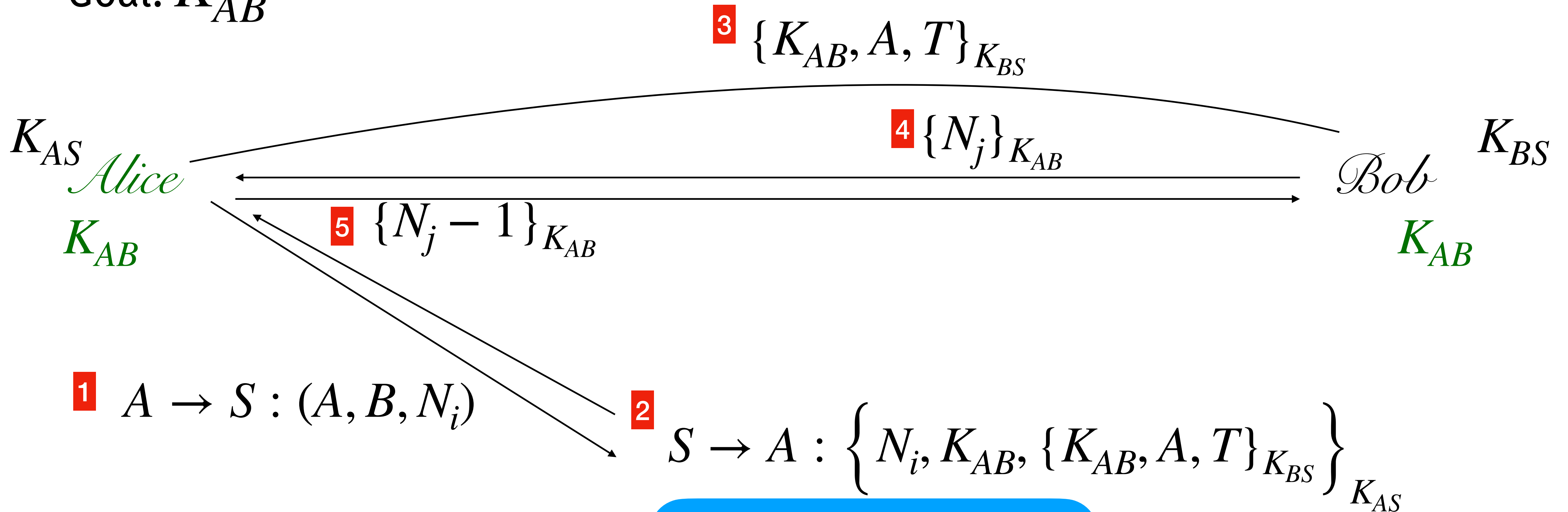
Alice updates K_{AS} .

K_{AS} NEU PWD Server
 K_{BS}

**Attacker
Replays
Message
To BOB!**

Fixed Needham-Schroeder Protocol

Goal: K_{AB}



K_{AS} NEU PWD
 K_{BS} Server

“Single Sign on”

Sign up with your identity provider

You'll use this service to log in to your network

 Sign up with Google

 Sign up with Microsoft

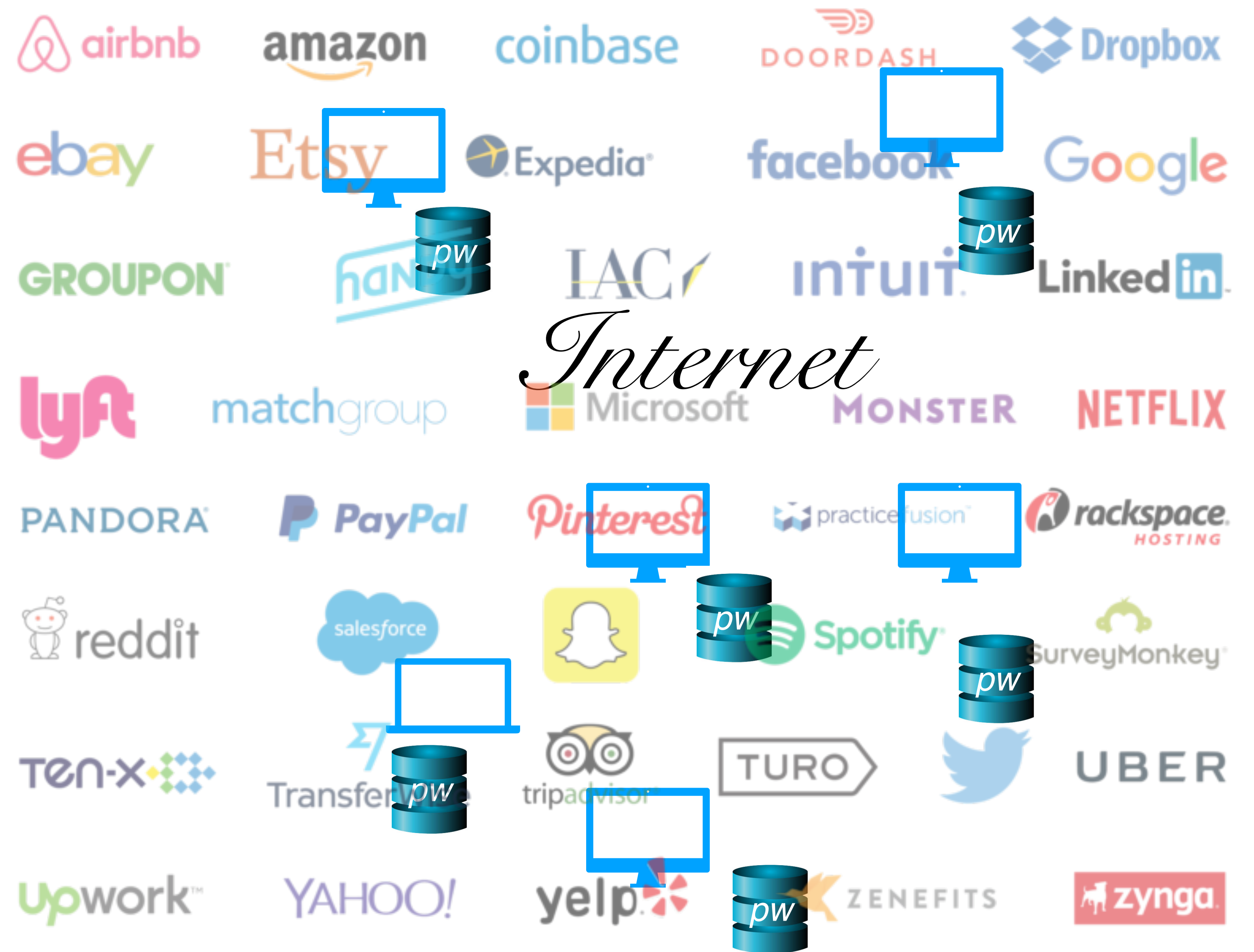
OR

|Enter your email...



Sign up with Email

Same problem as before

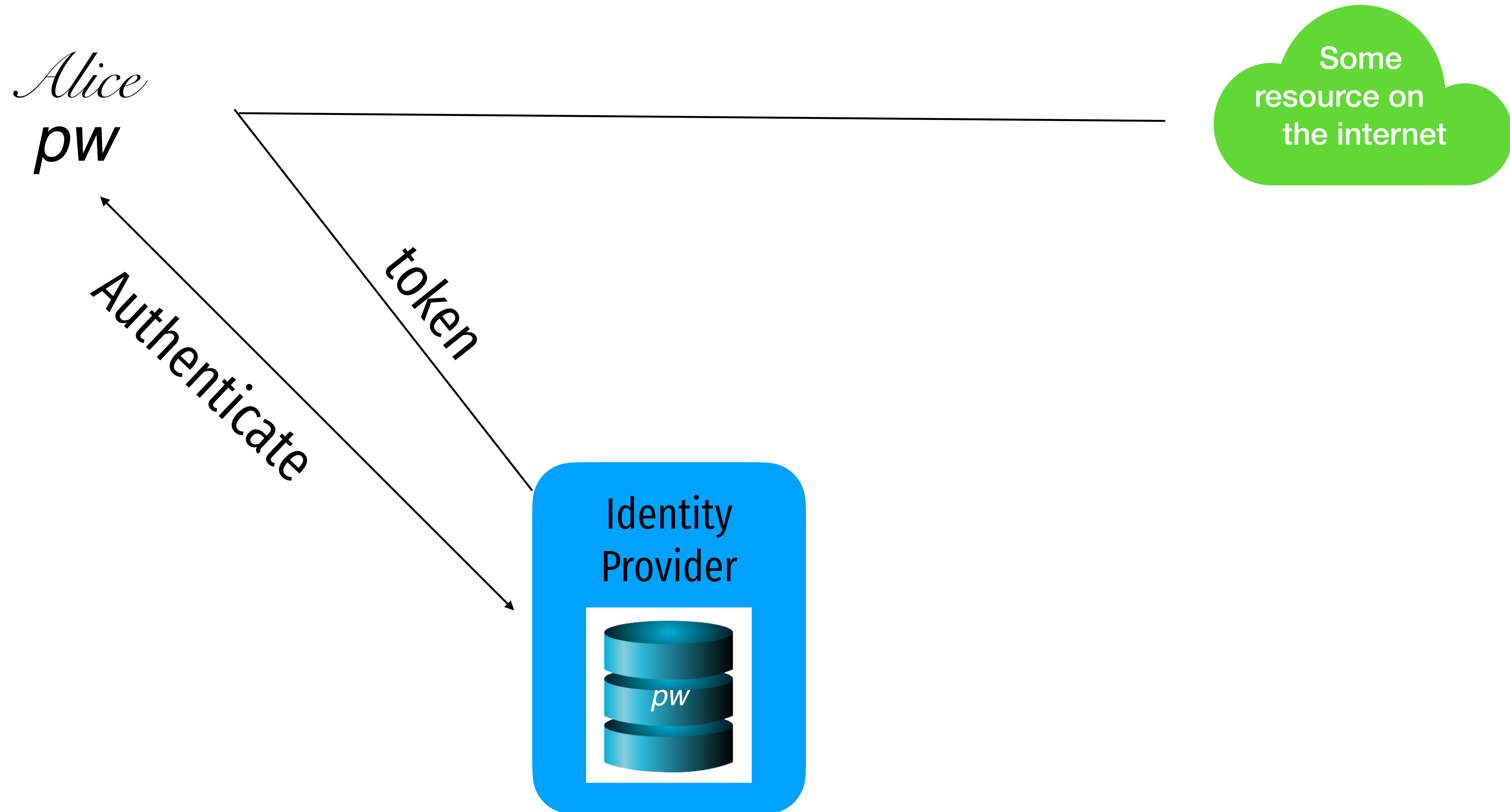


Alice
pw

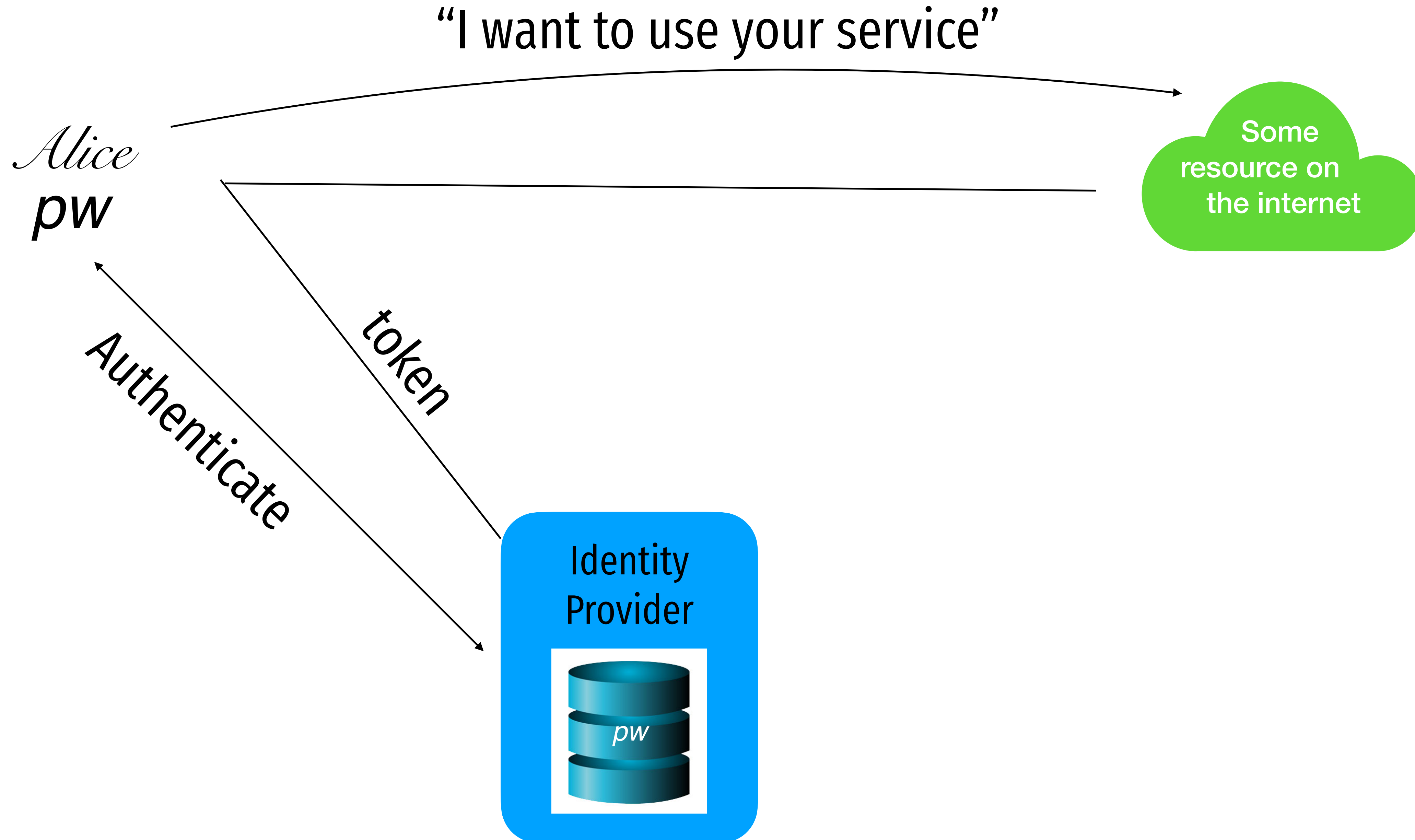
Kerberos

- Created as part of MIT Project Athena
 - Based on Needham-Schroeder
- Provides mutual authentication over untrusted networks
 - **Tickets** as assertions of authenticity, authorization
 - Forms basis of Active Directory authentication
- Principals
 - Client
 - Server
 - Key distribution center (KDC)
 - Authentication server (AS)
 - Ticket granting server (TGS)

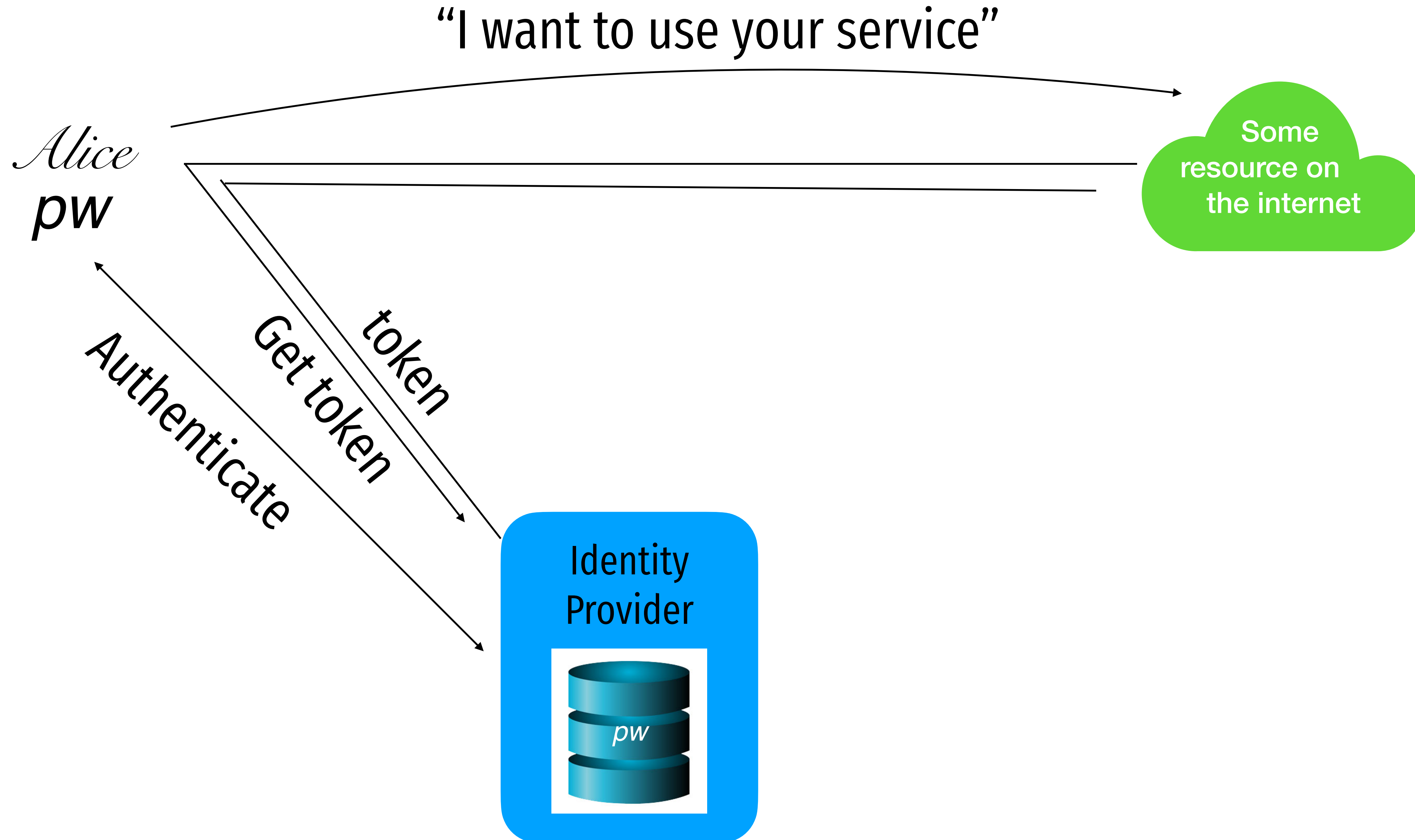
Oauth



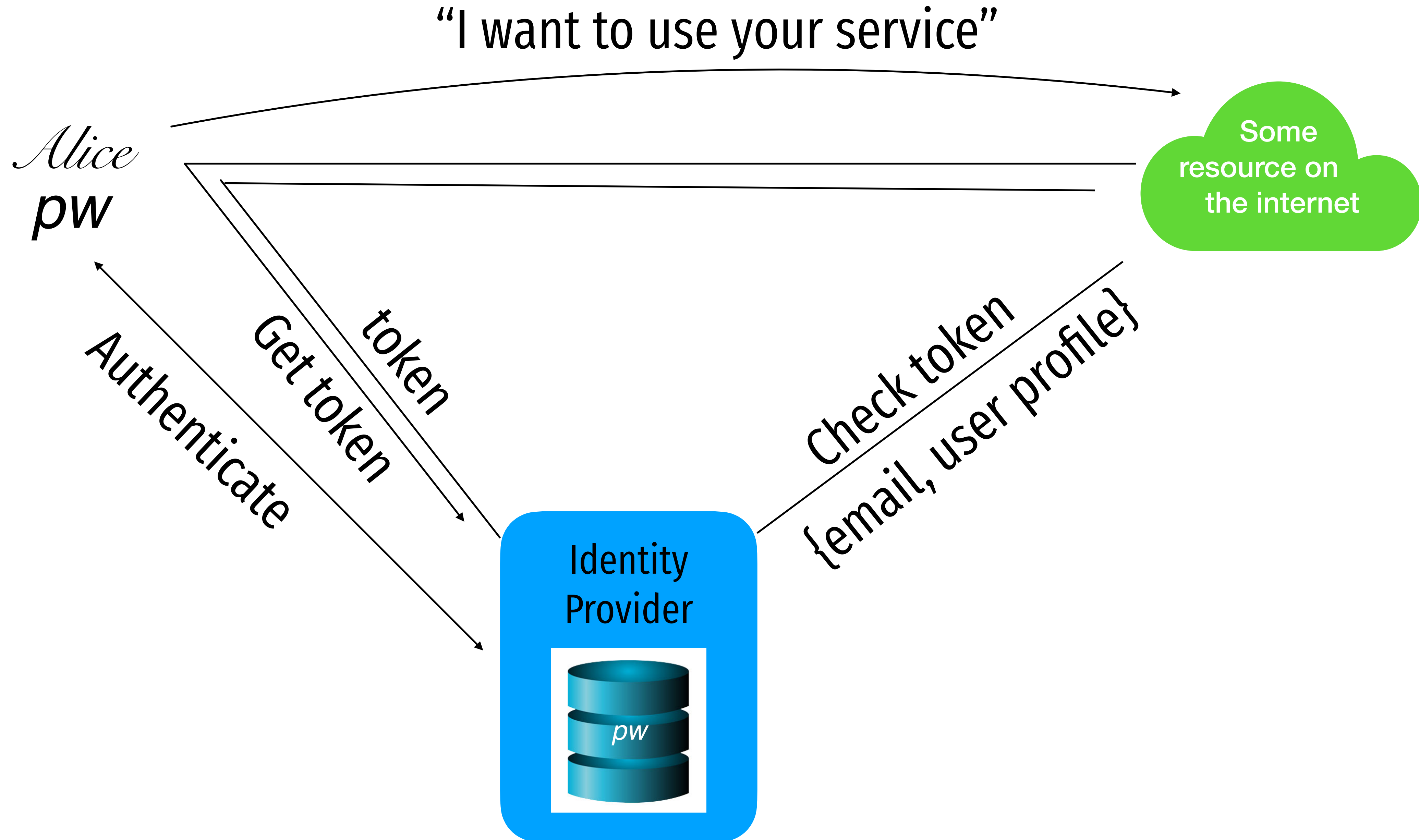
Oauth



Oauth



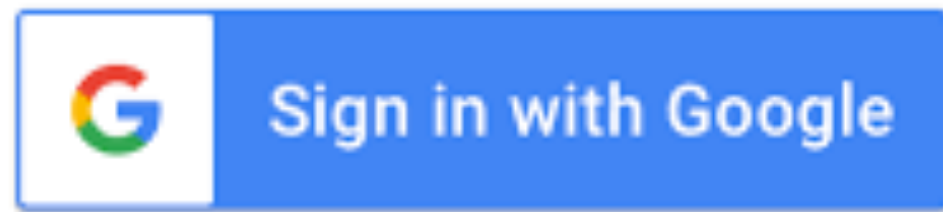
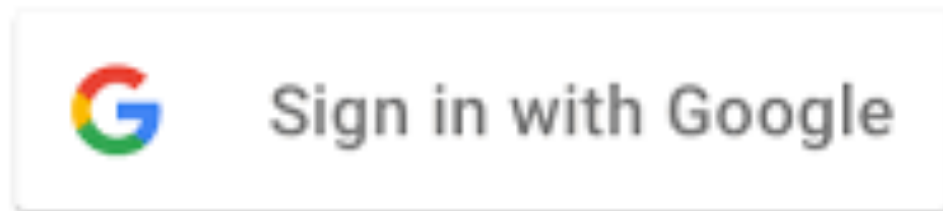
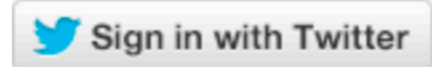
Oauth



Attacks against “Login with…” services

Log in with Twitter

Use Log in with Twitter, also known as Sign in with Twitter, to place a button on your site or application which allows Twitter users to enjoy the benefits of a registered user account in as little as one click. This works on websites, iOS, mobile, and desktop applications.



Use Sign in with Apple on your Apple device

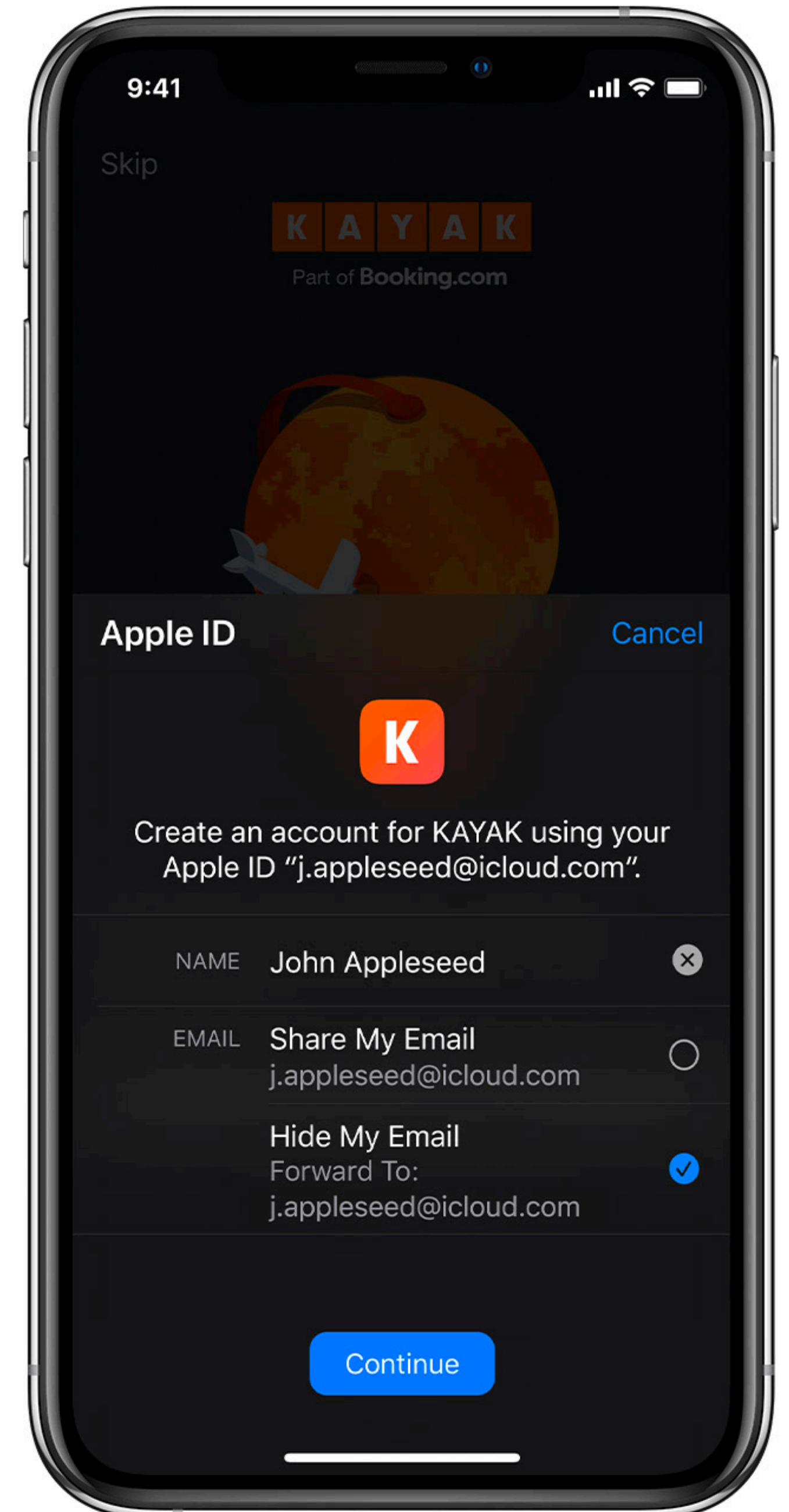
Using Sign in with Apple is quick and easy on any Apple device with the latest software. Make sure you're [signed in with your Apple ID](#) on your device.

1. Tap the Sign in with Apple button on the participating app or website.

If the app or site has not requested any information to set up your account, check that your Apple ID is correct and go to Step 4.

If you're asked to provide your name and email address, Sign in with Apple automatically fills in the information from your Apple ID. You can edit your name if you like and choose Share My Email or [Hide My Email](#).

Tap Continue and confirm with a quick Face ID, Touch ID, or device passcode to sign in. If you don't have Face ID, Touch ID, or a passcode set up, enter your Apple ID password.



Sources

1. Many slides courtesy of Wil Robertson: <https://wkr.io>
 2. Honeywords, Ari Juels and Ron Rivest: <http://www.arijuels.com/wp-content/uploads/2013/09/JR13.pdf>
- For more on generating secure passwords, and understanding people's mental models of passwords, see the excellent work of Blas Ur: <http://www.blaseur.com/pubs.htm>