2550 Intro to cybersecurity

L11: Passwords

abhi shelat

Thanks Christo for slides!

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

(434 535
2244)

Example PAM Configuration

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

auth required pam_unix.so try_first_pa
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:$1$l3RxU5F1$: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>

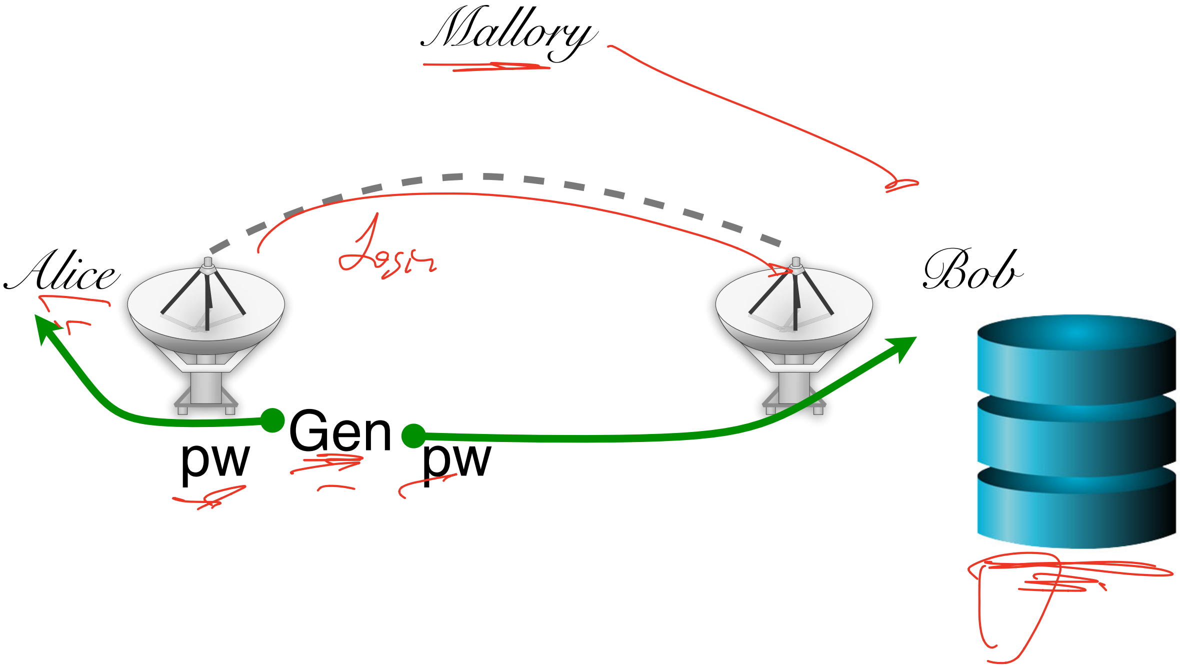
Algo: 1 = MD5, 5 = SHA256, 6 = SHA512

/etc/shadow

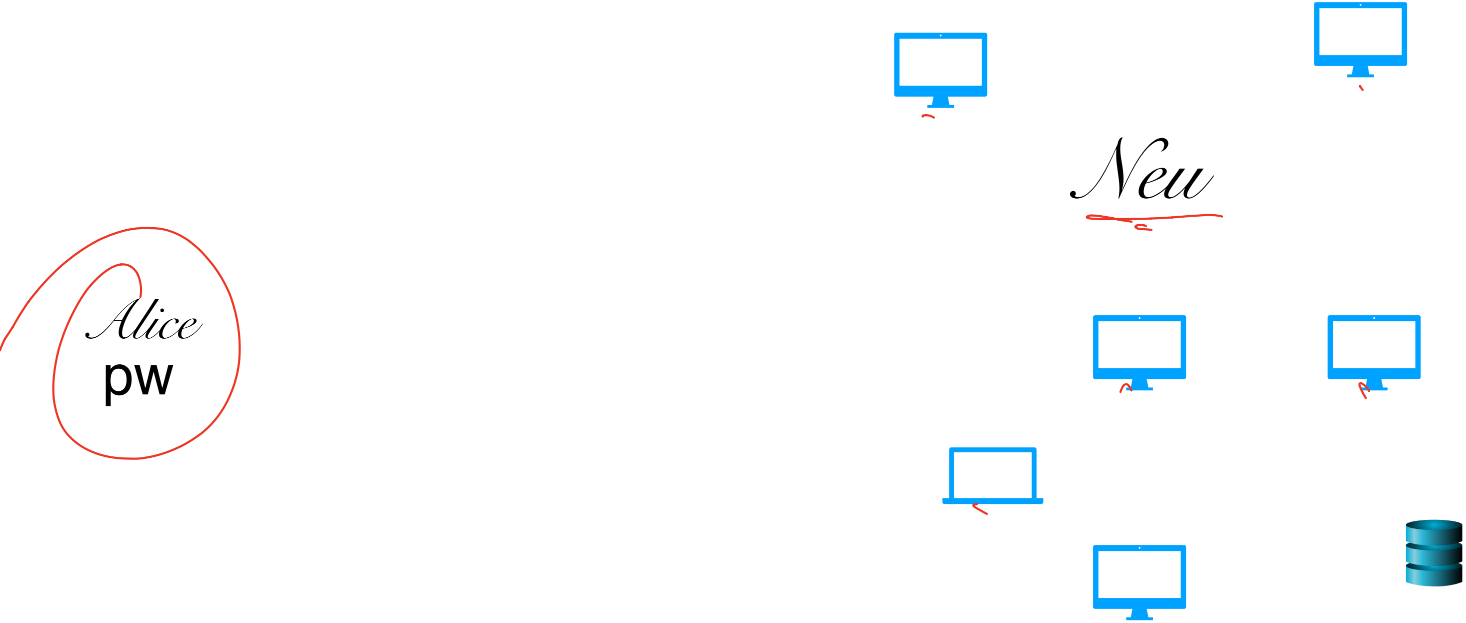
ername:password:last:may:must:warn:expire:disable:reserved

```
cbw:$1$0nSd5ewF$0df/3G7iSV49nsbAa/5gSg:9479:0:10000::::
amislove:$1$l3RxU5F1$: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?

① pwd for each machine ?;?

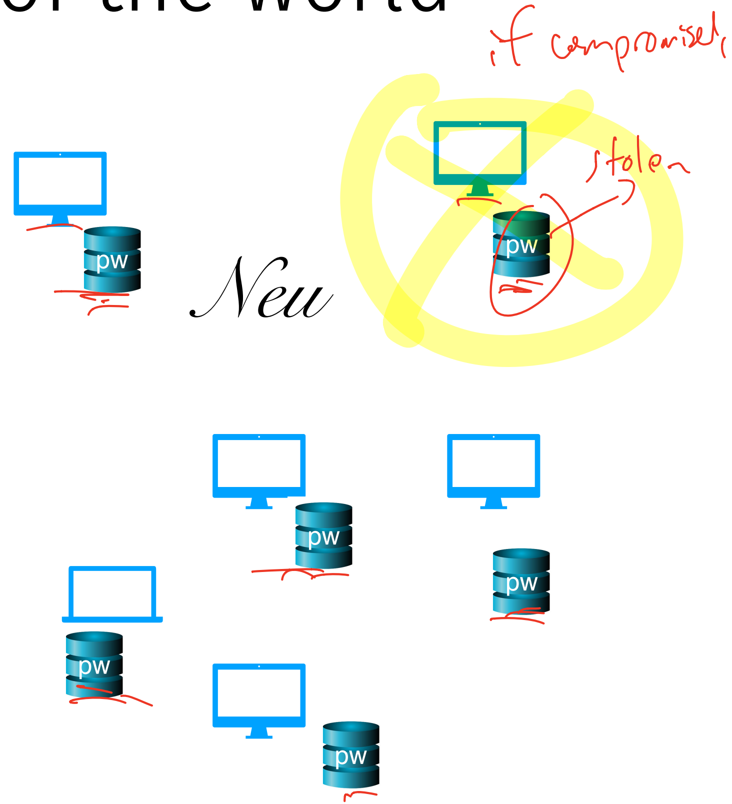
usability

Alice

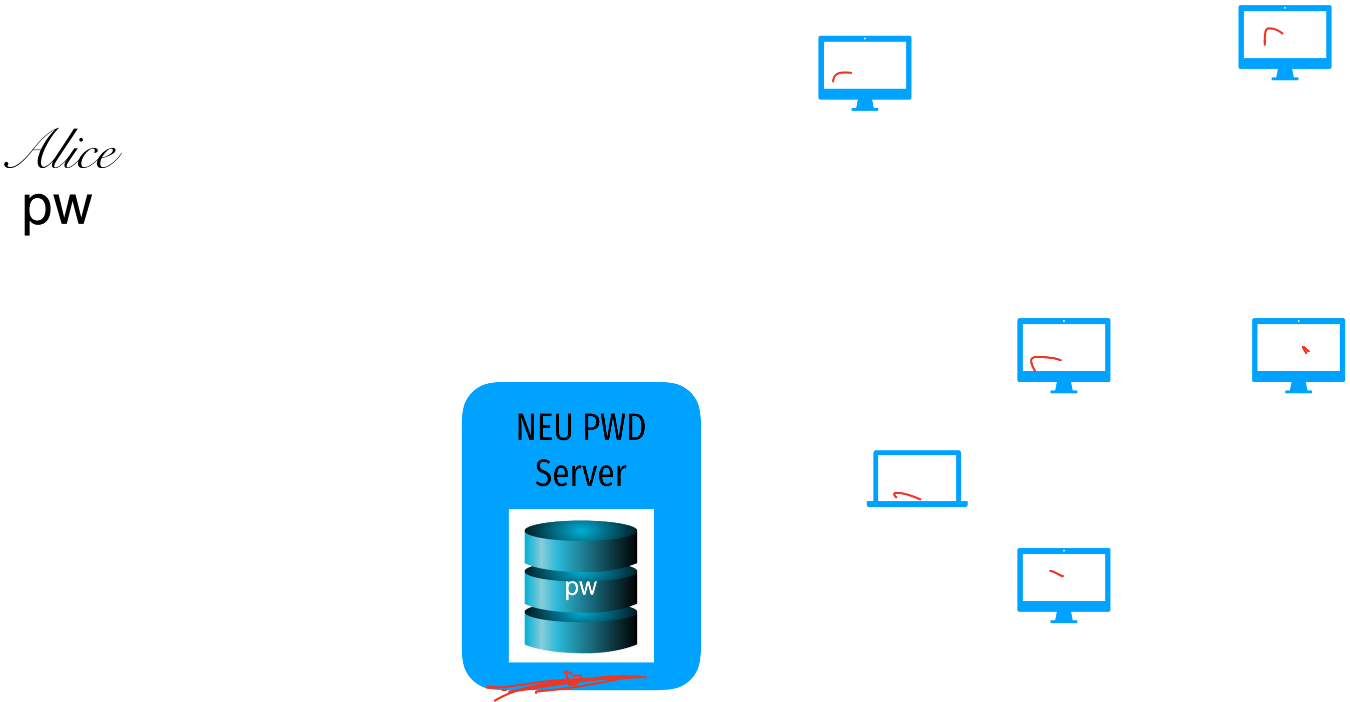
~~pw~~

② same pwd everywhere

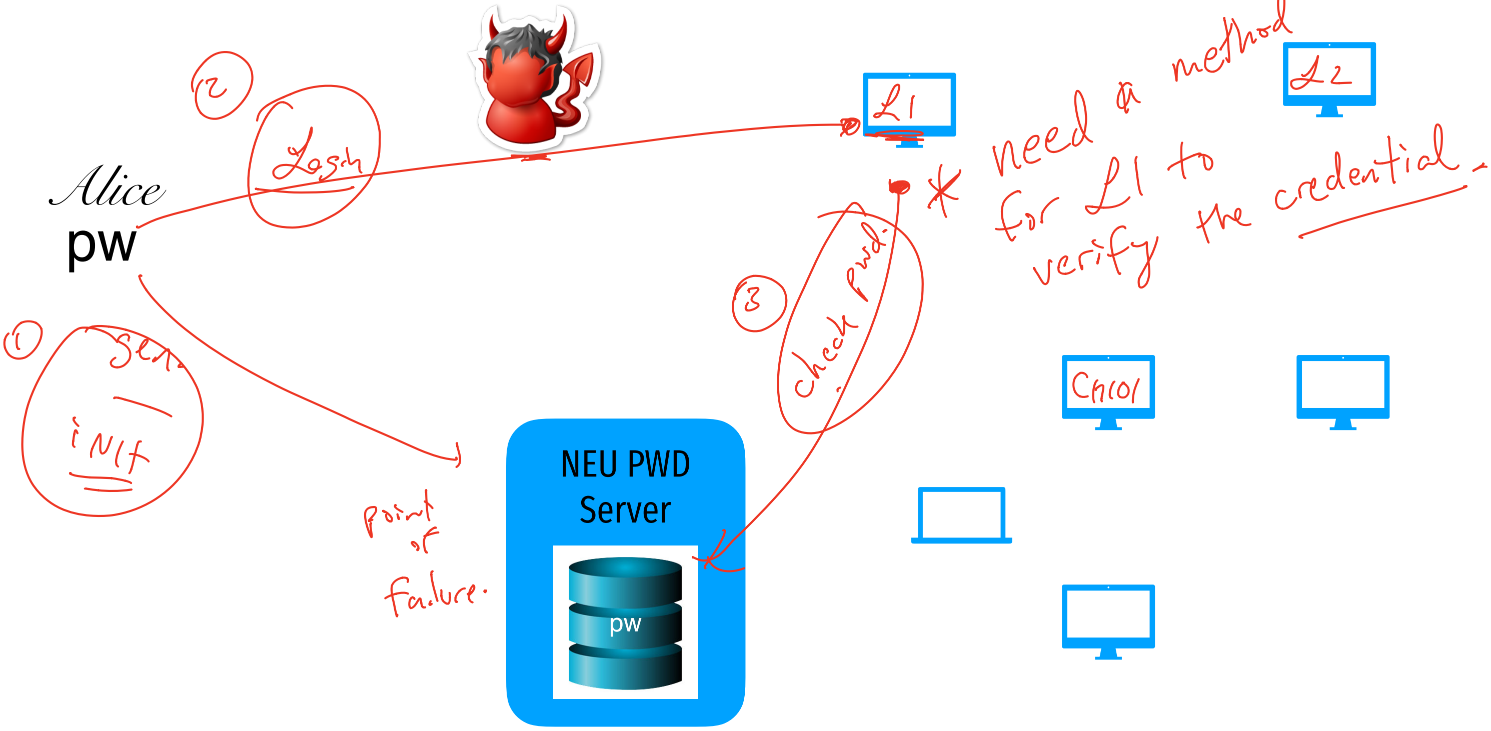
security problem.



The problem of distributed authentication



Distributed authentication: Attacker model



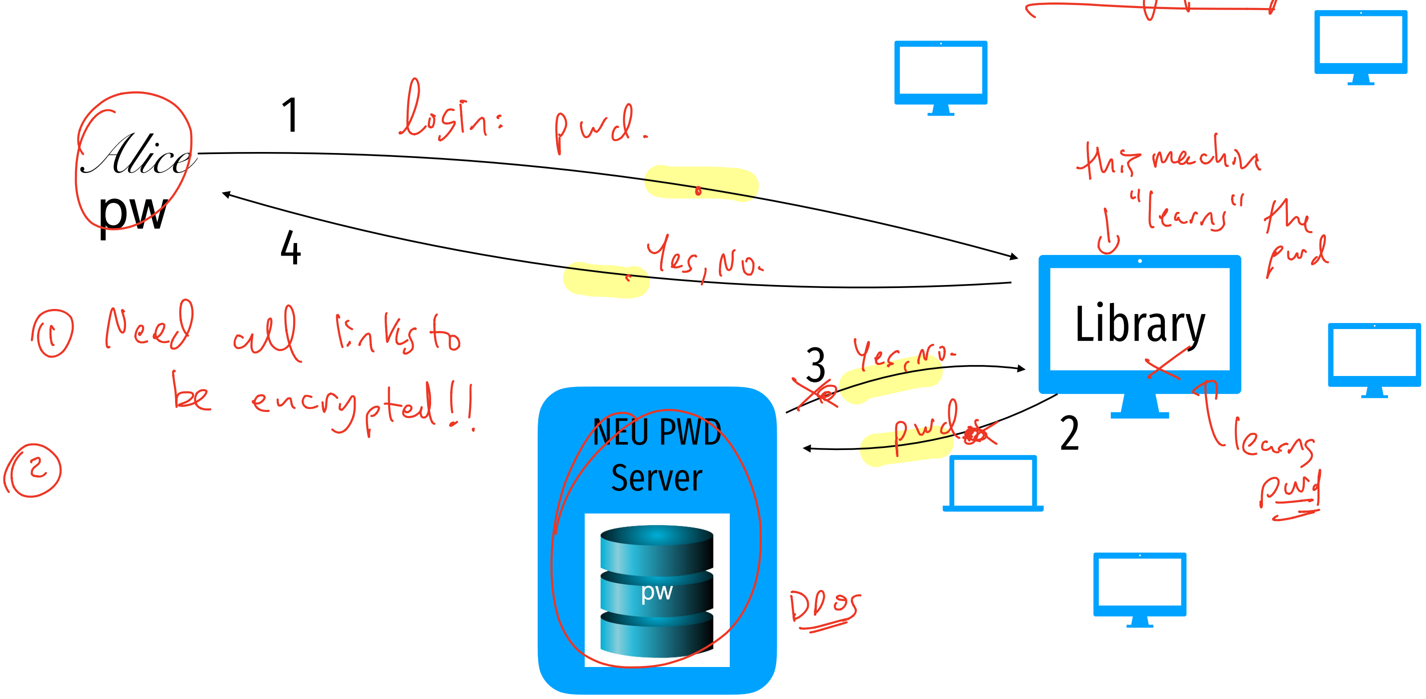
What can attacker do?

* need a method for L1 to verify the credential.

Distributed authentication: Bad Solution

What can attacker do?

Relying party



login: pwd.

Yes, No.

this machine "learns" the pwd

Yes, No.

pwd

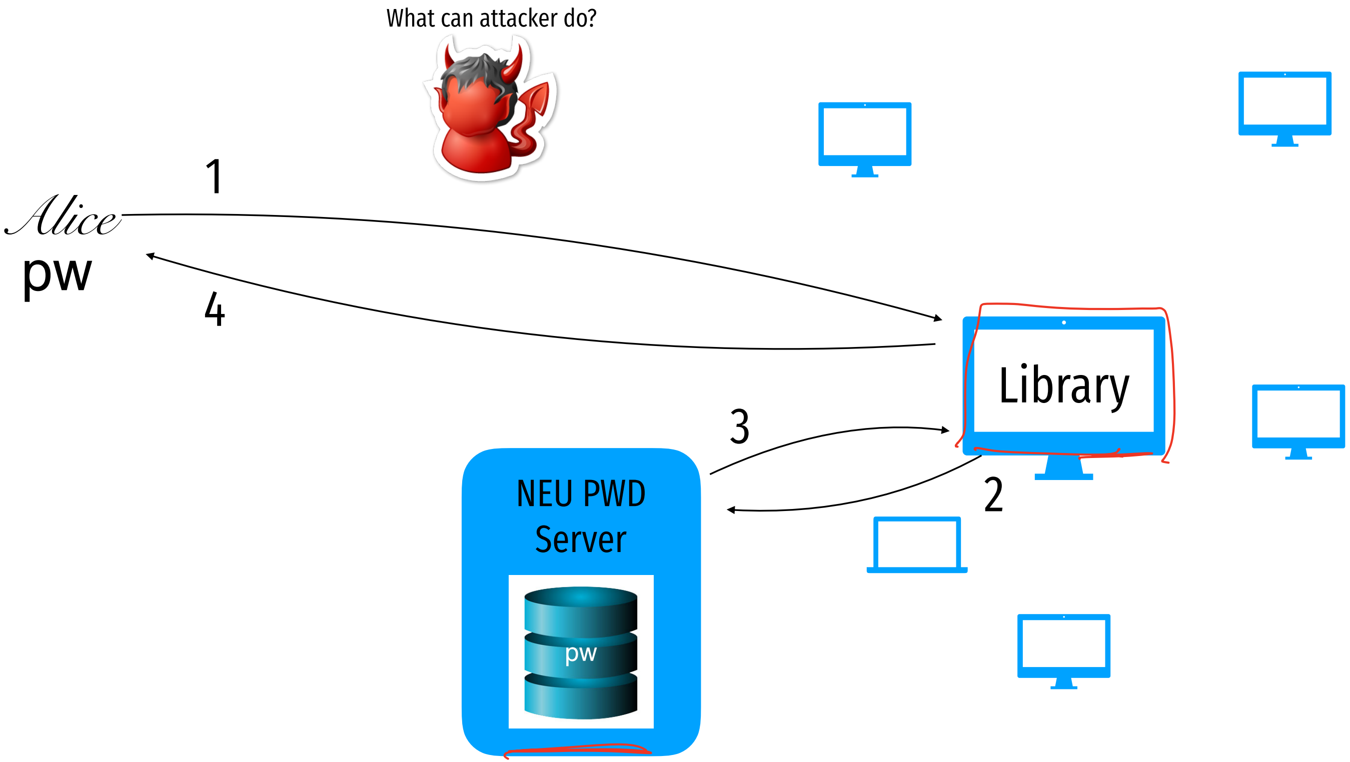
learns pwd

DDos

① Need all links to be encrypted!!

②

Distributed authentication: Bad Solution



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) $A \rightarrow S: A, B, N_i$
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) $A \rightarrow B: \{N_j - 1\}_{K_{AB}}$

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

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

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

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

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

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

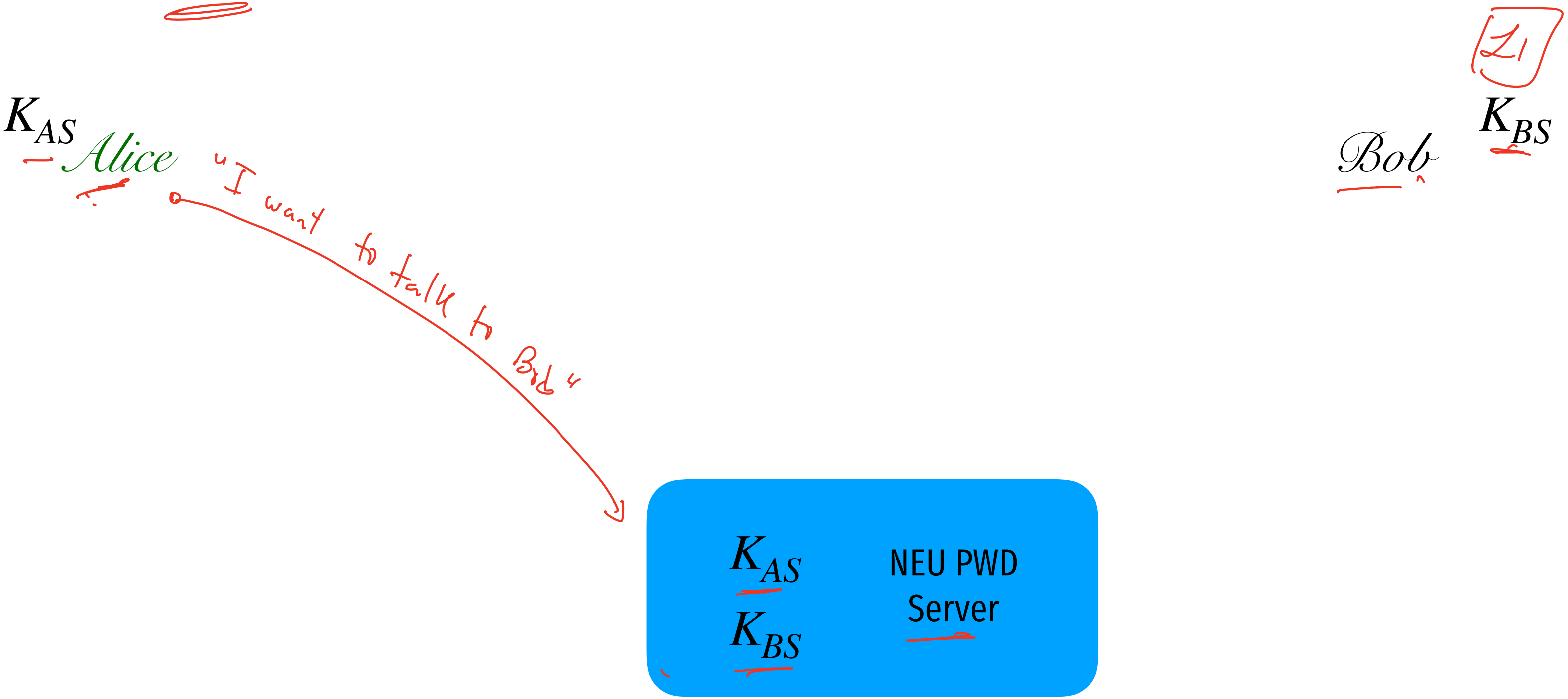
4) $B \rightarrow A: \{N_j\}_{K_{AB}}$

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

"I want to talk to Bob"

L1

K_{BS} Bob

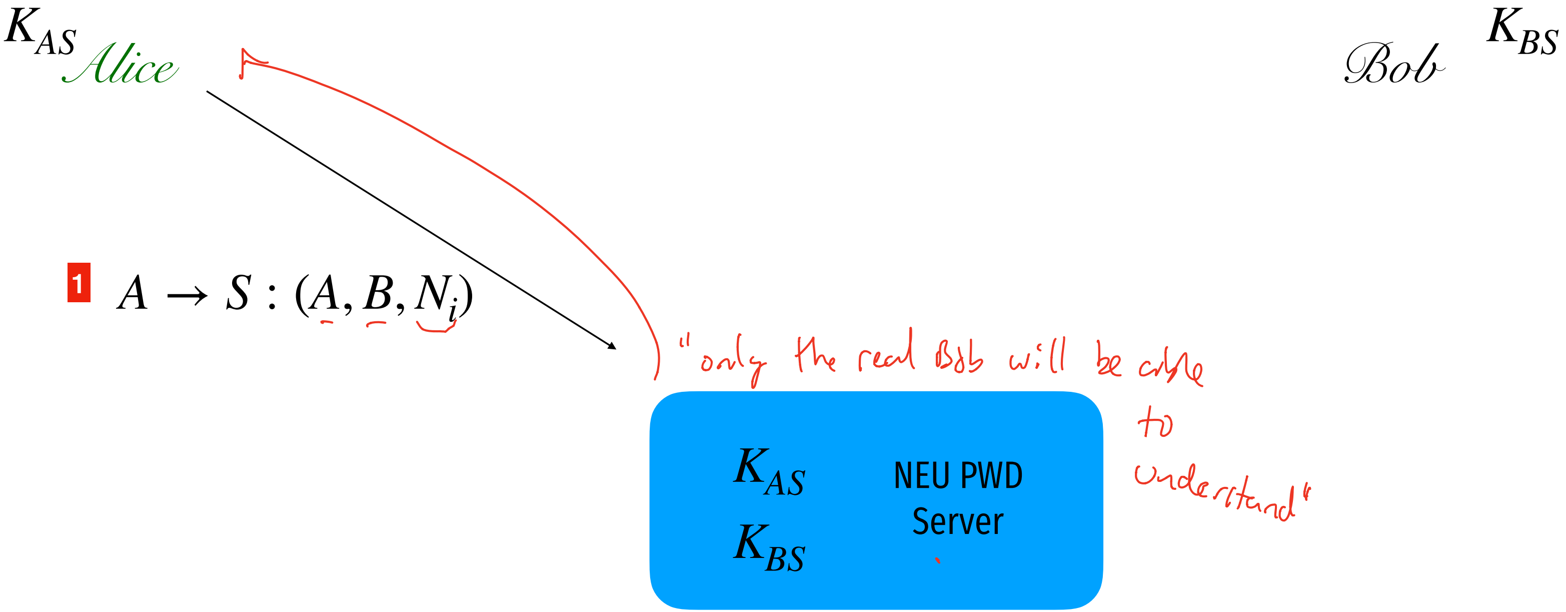
K_{AS}

K_{BS}

NEU PWD Server

Notorious Needham-Schroeder Protocol

Goal: K_{AB}

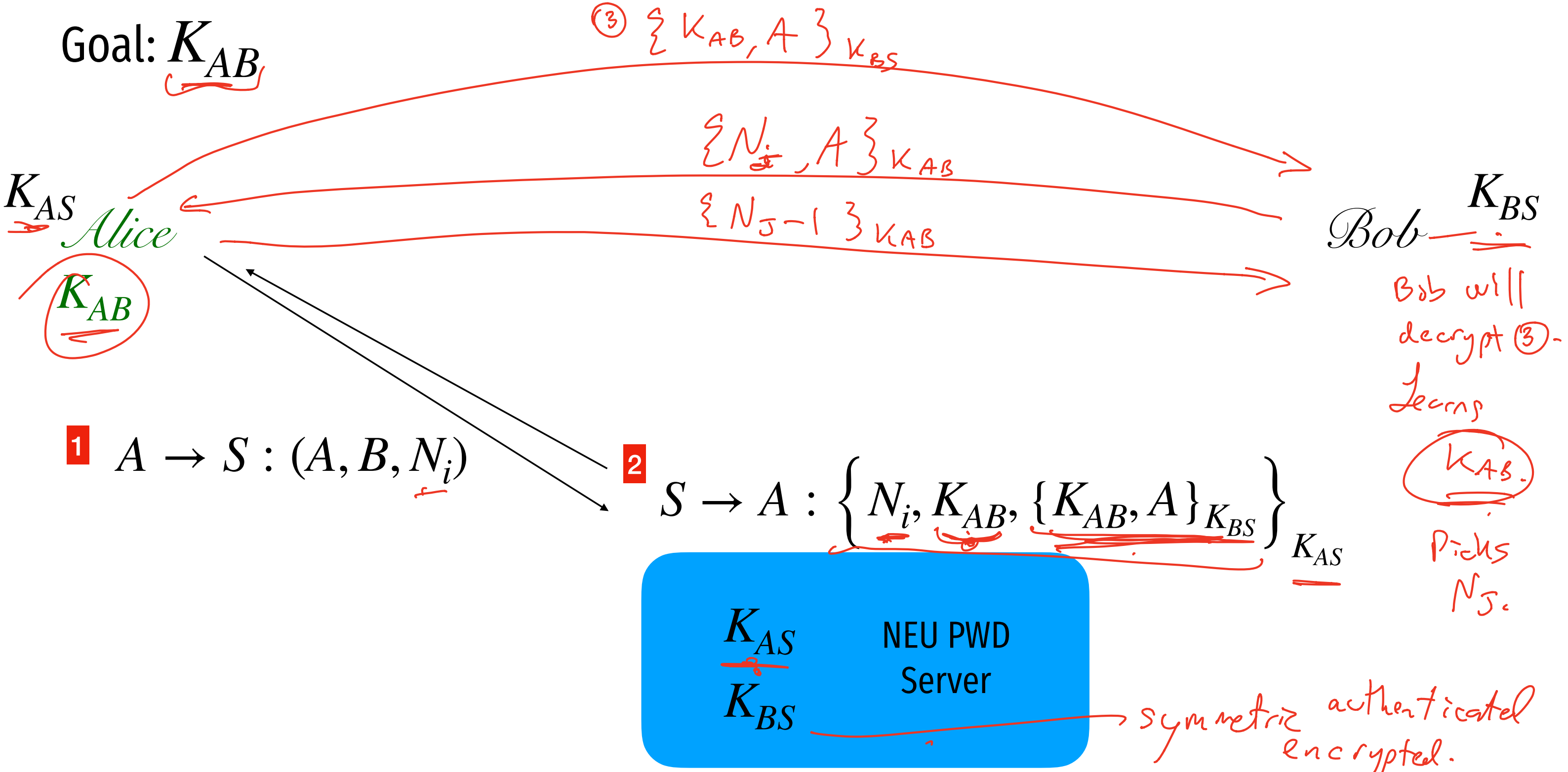


1 $A \rightarrow S : (\underline{A}, \underline{B}, \underline{N_i})$

"only the real Bob will be able to understand"

Notorious Needham-Schroeder Protocol

Goal: K_{AB}



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

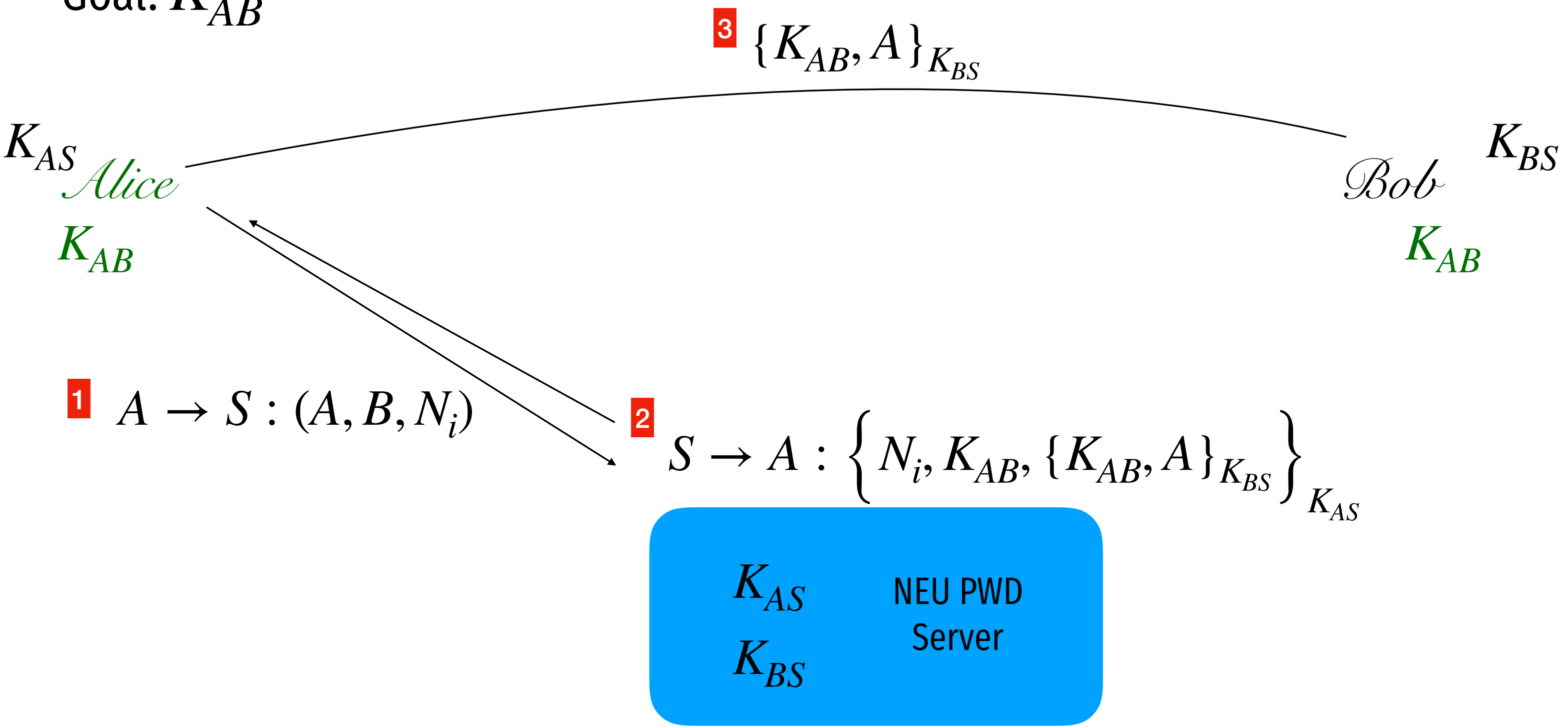
2 $S \to A : \left\{N_i, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\right\}_{K_{AS}}$

Bob will decrypt ③. learns K_{AB}. Picks N_J.

symmetric authenticated encrypted.

Notorious Needham-Schroeder Protocol

Goal: K_{AB}



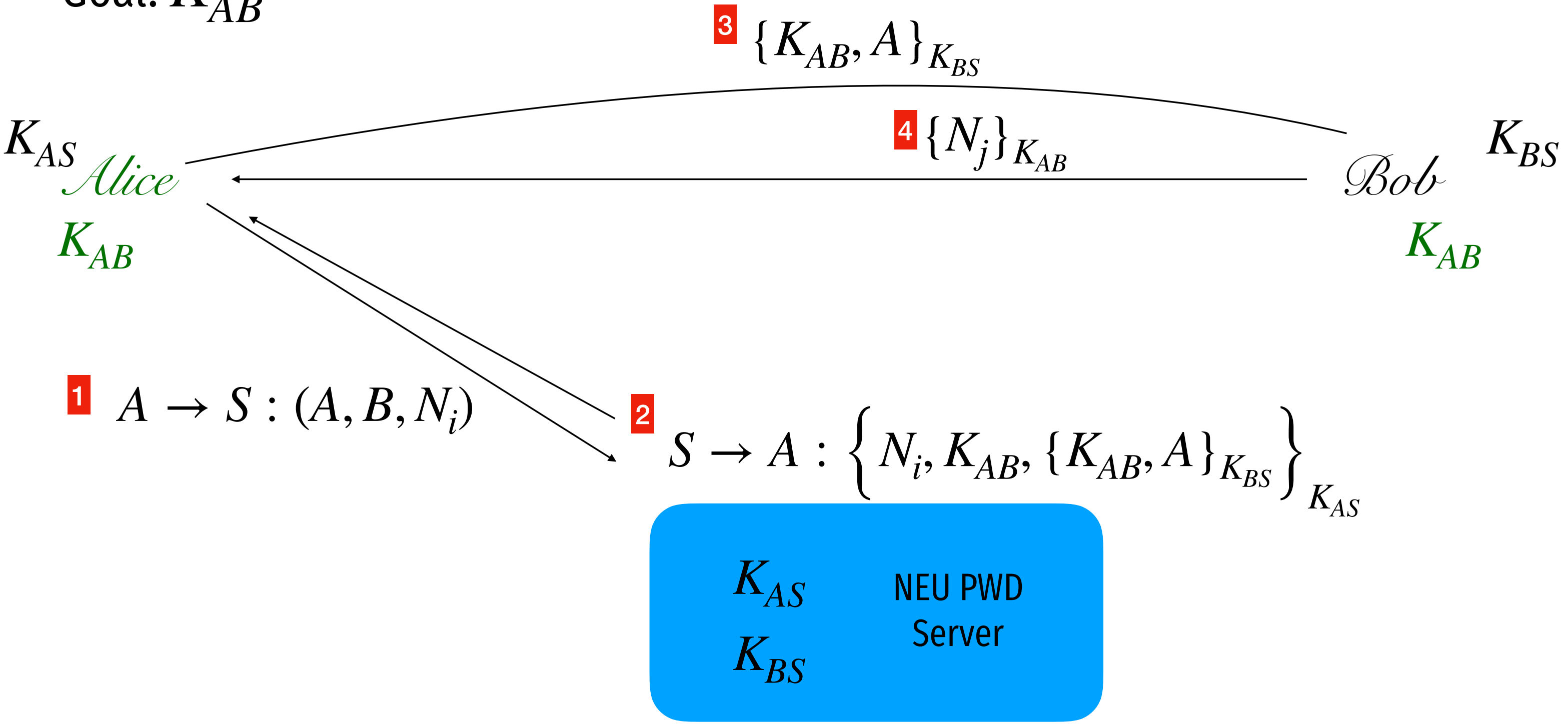
3 $\{K_{AB}, A\}_{K_{BS}}$

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

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

Notorious Needham-Schroeder Protocol

Goal: K_{AB}



3 $\{K_{AB}, A\}_{K_{BS}}$

K_{AS} Alice K_{AB}

4 $\{N_j\}_{K_{AB}}$

Bob K_{BS} K_{AB}

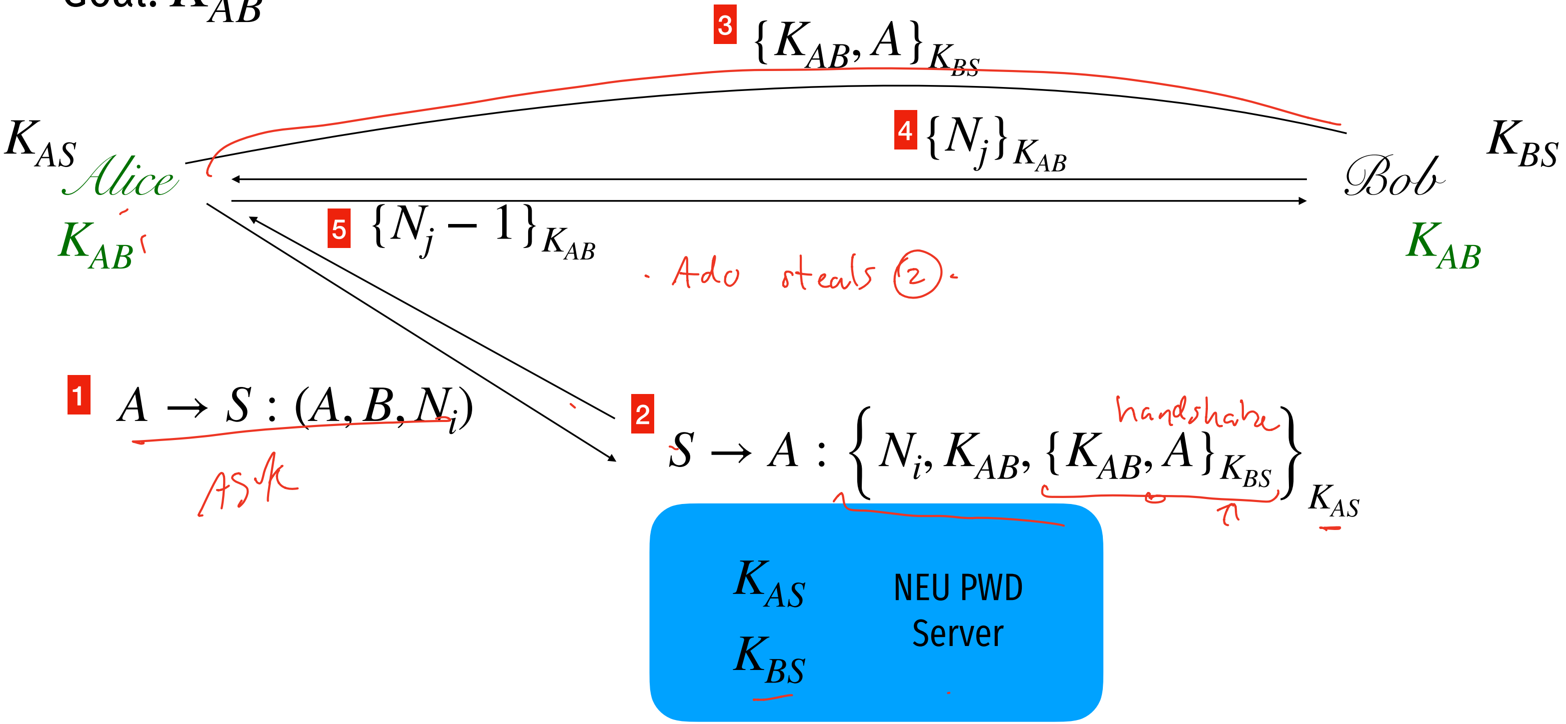
1 $A \rightarrow S : (A, B, N_i)$

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

K_{AS} K_{BS} NEU PWD Server

Notorious Needham-Schroeder Protocol

Goal: K_{AB}



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

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

3 $\{K_{AB}, A\}_{K_{BS}}$

4 $\{N_j\}_{K_{AB}}$

5 $\{N_j - 1\}_{K_{AB}}$

Notorious Needham-Schroeder Protocol

Goal: K_{AB}

Suppose attacker tries to impersonate Alice

K_{AS} Alice

Attackr.

Bob K_{BS}

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

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

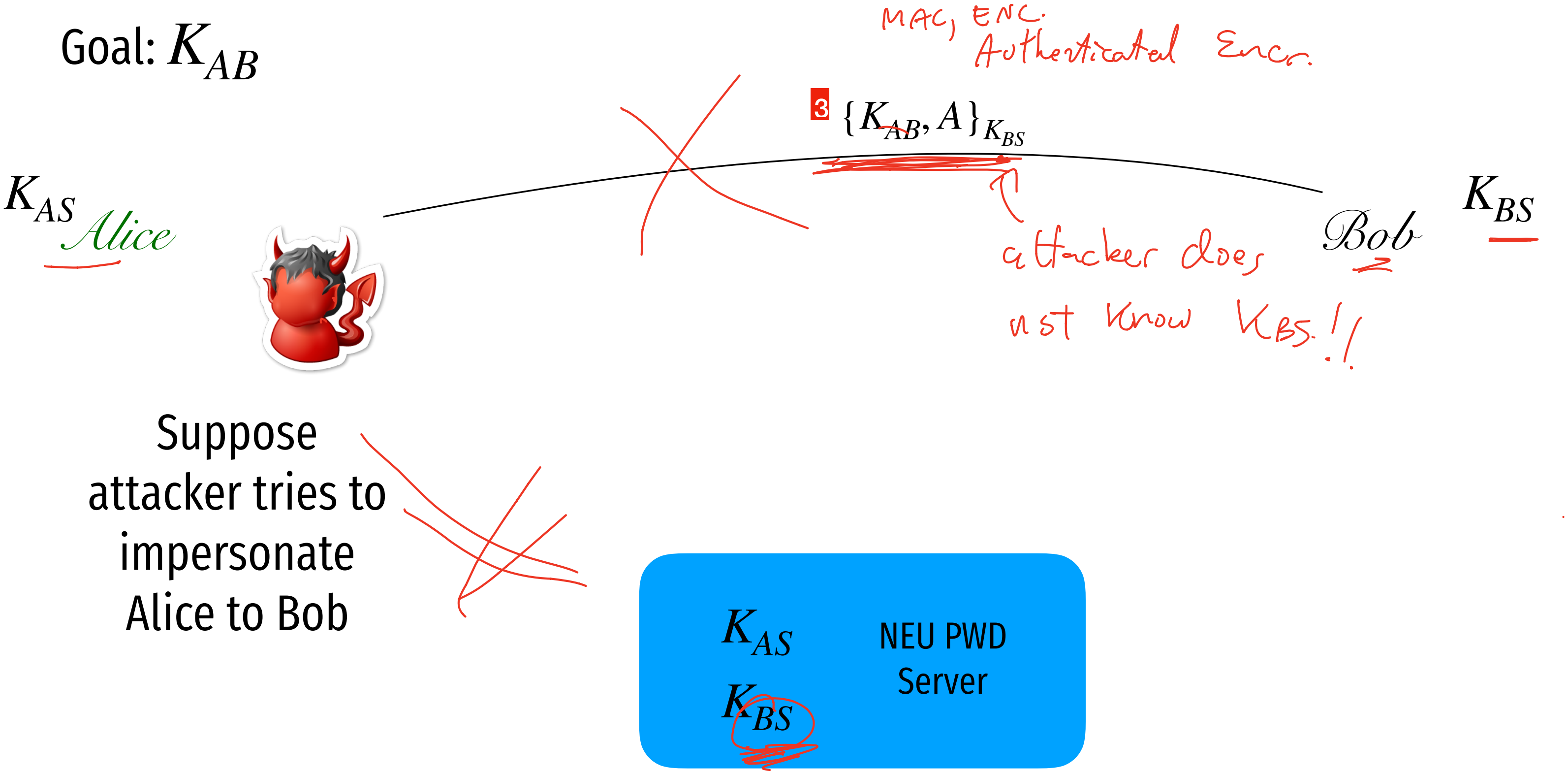
K_{AS}

K_{BS}

NEU PWD Server

Notorious Needham-Schroeder Protocol

Goal: K_{AB}



MAC, ENC.
Authenticated Encr.

3 $\{K_{AB}, A\}_{K_{BS}}$

attacker does
nst know K_{BS}.!!

K_{AS} Alice

Bob K_{BS}

Suppose attacker tries to impersonate Alice to Bob

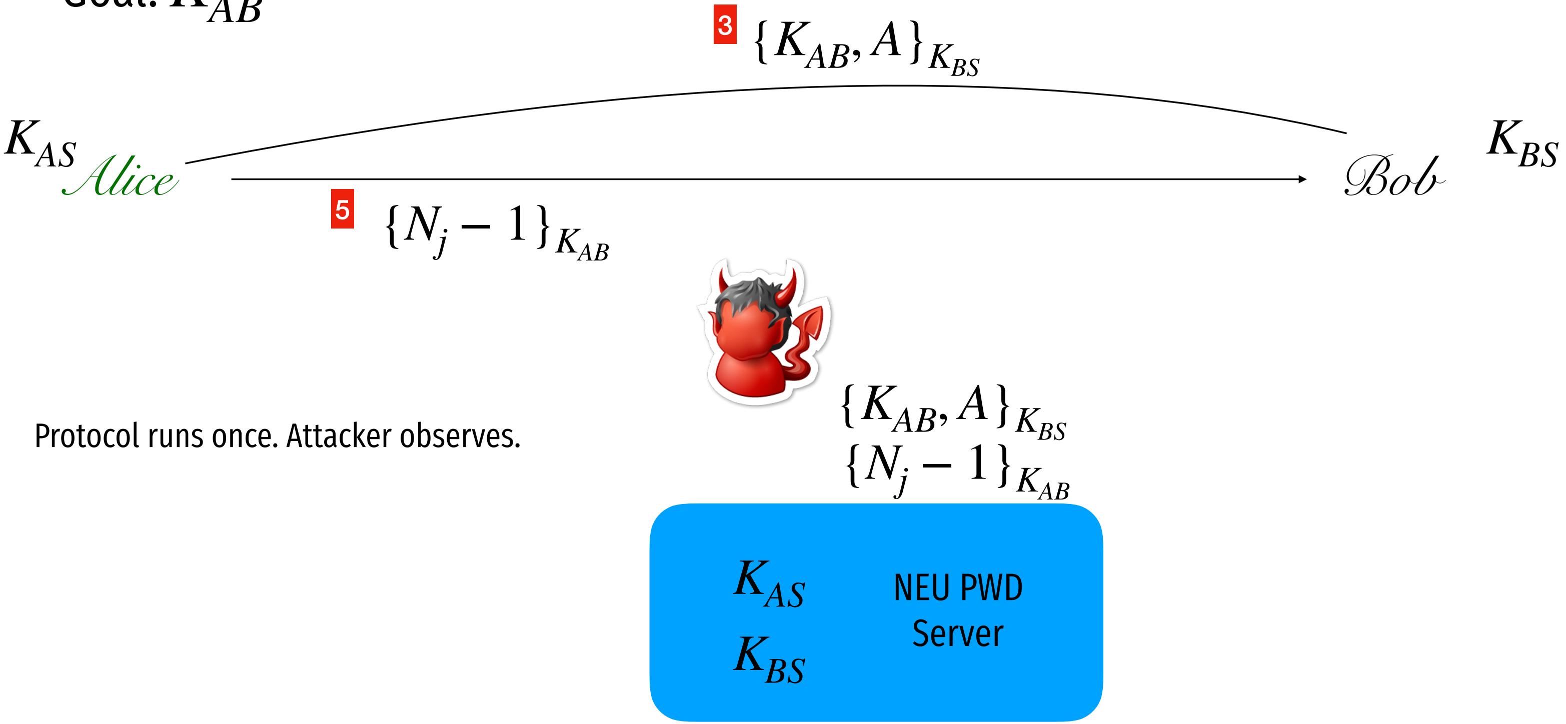
K_{AS}

K_{BS}

NEU PWD Server

Needham-Schroeder Replay Attack

Goal: K_{AB}



3 $\{K_{AB}, A\}_{K_{BS}}$

K_{AS} Alice

Bob K_{BS}

5 $\{N_j - 1\}_{K_{AB}}$

$\{K_{AB}, A\}_{K_{BS}}$

$\{N_j - 1\}_{K_{AB}}$

K_{AS}

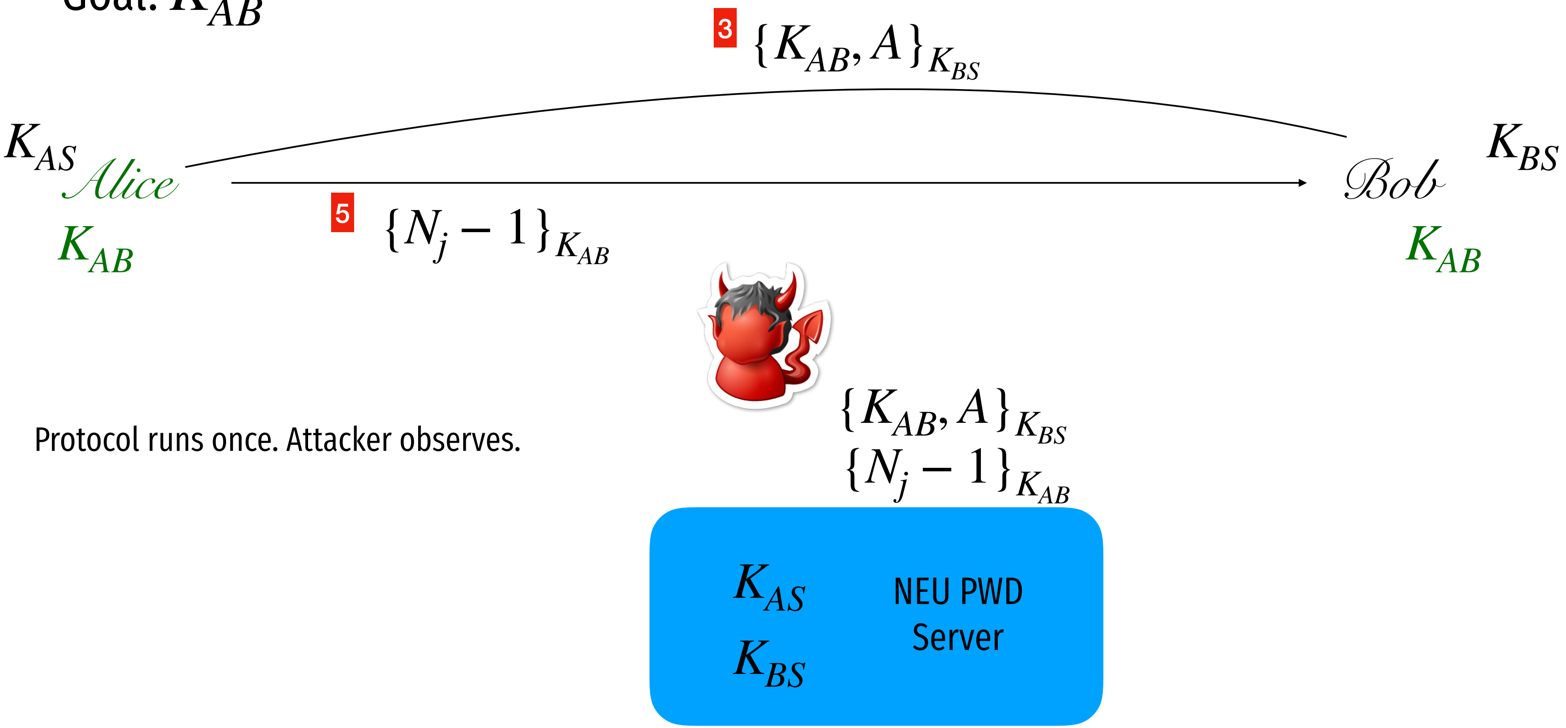
K_{BS}

NEU PWD Server

Protocol runs once. Attacker observes.

Needham-Schroeder Replay Attack

Goal: K_{AB}



3 $\{K_{AB}, A\}_{K_{BS}}$

K_{AS} Alice K_{AB}

5 $\{N_j - 1\}_{K_{AB}}$

Bob K_{BS} K_{AB}

$\{K_{AB}, A\}_{K_{BS}}$

$\{N_j - 1\}_{K_{AB}}$

K_{AS} K_{BS} NEU PWD Server

Protocol runs once. Attacker observes.

Needham-Schroeder Replay Attack

Goal: K_{AB}

K_{AS} *Alice*

K_{AB}

Bob K_{BS}

K_{AB}

$\{K_{AB}, A\}_{K_{BS}}$

$\{N_j - 1\}_{K_{AB}}$

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.
Attacker breaks into Alice and steals old K_AB.

K_{AB}

$\{K_{AB}, A\}_{K_{BS}}$

$\{N_j - 1\}_{K_{AB}}$

K_{AS}

K_{BS}

NEU PWD
Server

Needham-Schroeder Replay Attack

Goal: K_{AB}

K'_{AS}

Alice

K_{AB}

K_{BS}

Bob

K_{AB}

$\{K_{AB}, A\}_{K_{BS}}$

K_{AB} $\{N_j - 1\}_{K_{AB}}$

Protocol runs once. Attacker observes.

Attacker breaks into Alice and steals old K_AB.

Alice updates K_AS.

K_{AS}

K_{BS}

NEU PWD Server

Needham-Schroeder Replay Attack

Goal: K_{AB}

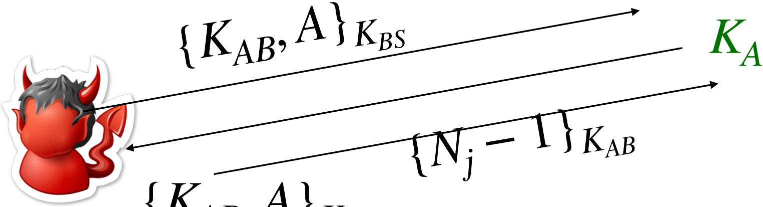
K'_{AS}

Alice

K_{AB}

Bob K_{BS}

K_{AB}



$\{K_{AB}, A\}_{K_{BS}}$

$\{N_j - 1\}_{K_{AB}}$

K_{AB} $\{K_{AB}, A\}_{K_{BS}}$

$\{N_j - 1\}_{K_{AB}}$

K_{AS}

K_{BS}

NEU PWD
Server

Protocol runs once. Attacker observes.

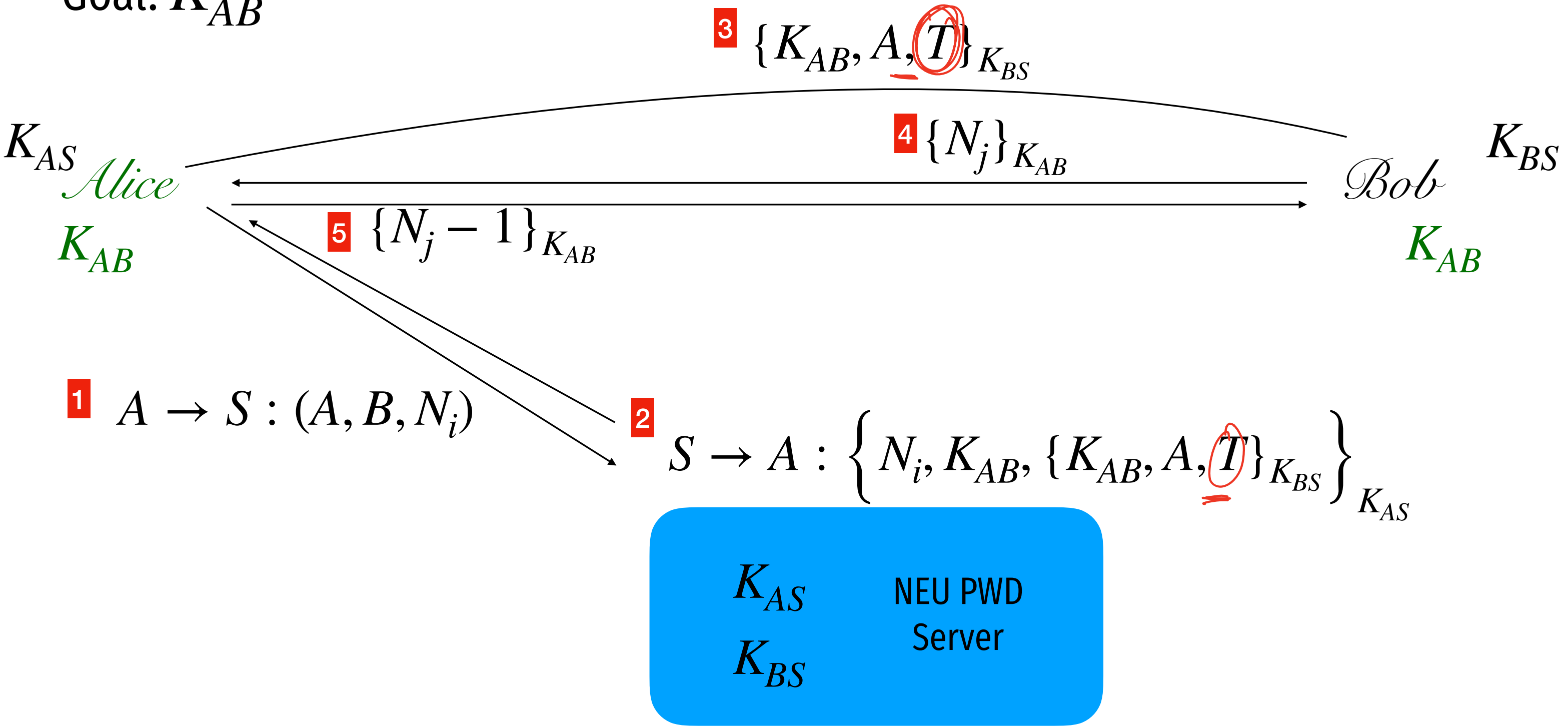
Attacker breaks into Alice and steals old K_AB.

Alice updates K_AS.

Attacker
Replays
Message
To BOB!

Fixed Needham-Schroeder Protocol

Goal: K_{AB}



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

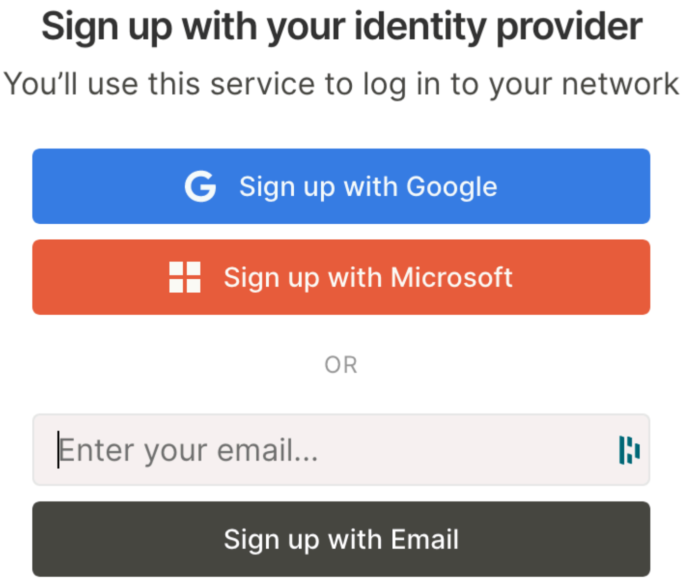
2 $S \rightarrow A : \left\{ N_i, K_{AB}, \{K_{AB}, A, T\}_{K_{BS}} \right\}_{K_{AS}}$

3 $\{K_{AB}, A, T\}_{K_{BS}}$

4 $\{N_j\}_{K_{AB}}$

5 $\{N_j - 1\}_{K_{AB}}$

“Single Sign on”



Sign up with your identity provider

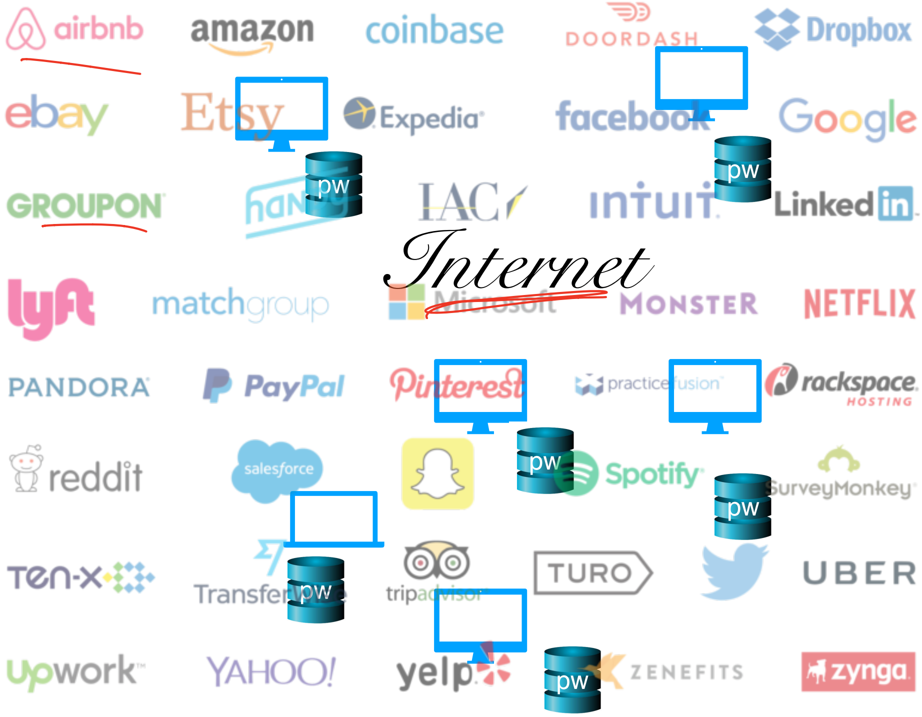
You’ll use this service to log in to your network

Same problem as before

Alice

pw

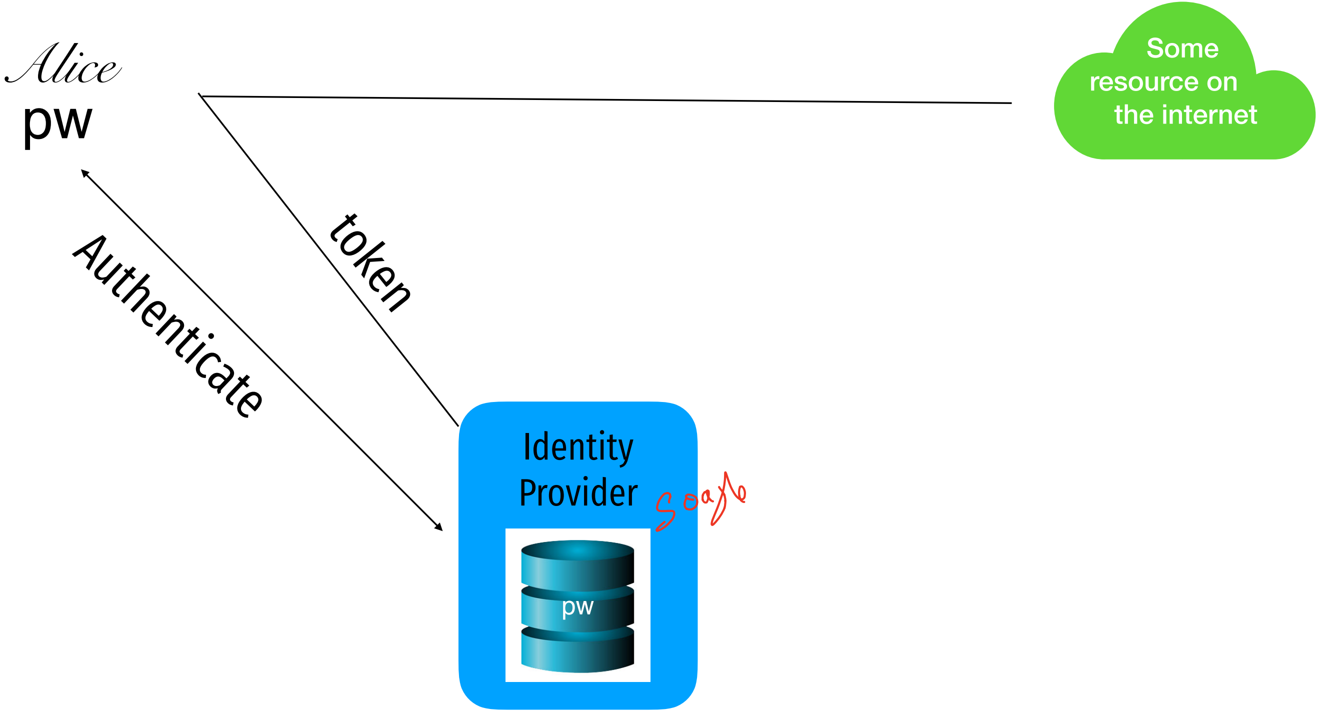
pwd
server.
google



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



Alice
pw

Authenticate

token

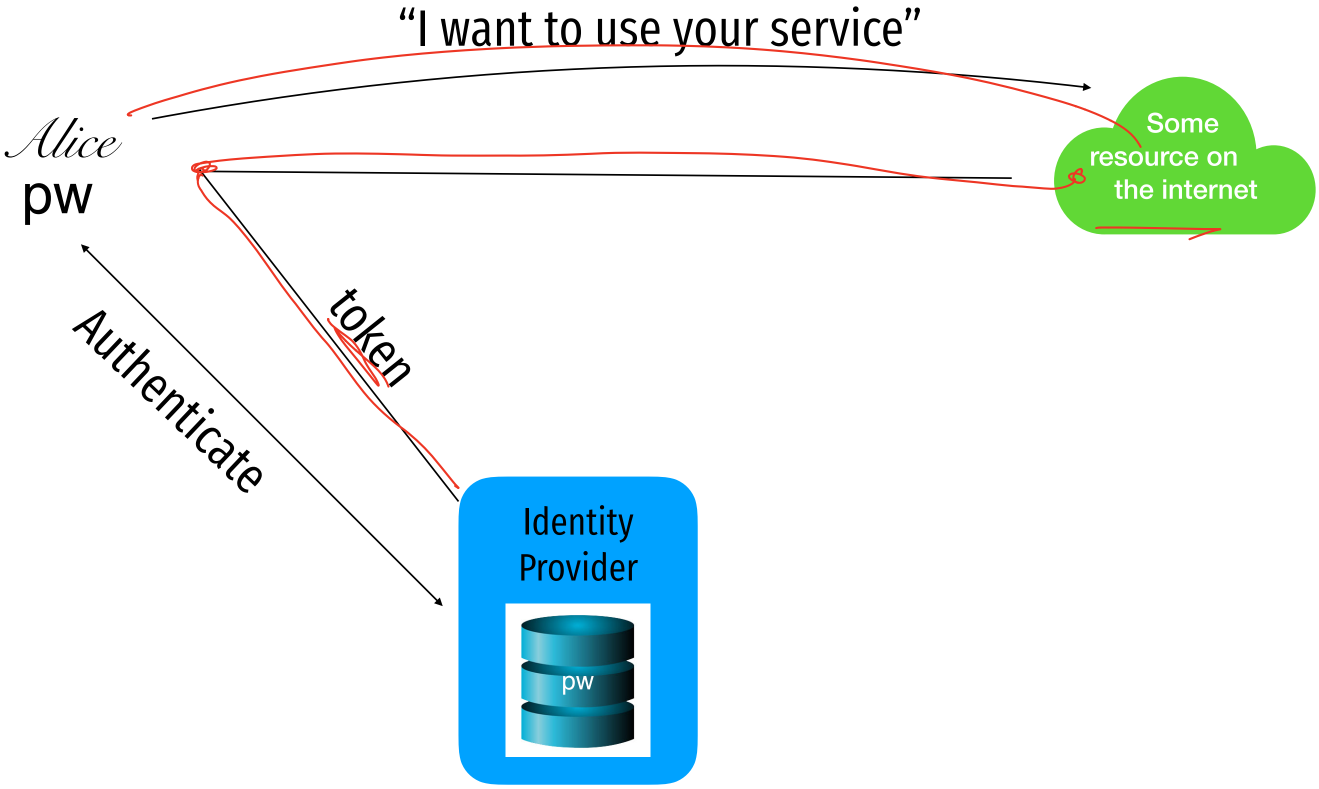
Identity Provider

pw

Soafe

Some resource on the internet

Oauth



“I want to use your service”

Alice

pw

Authenticate

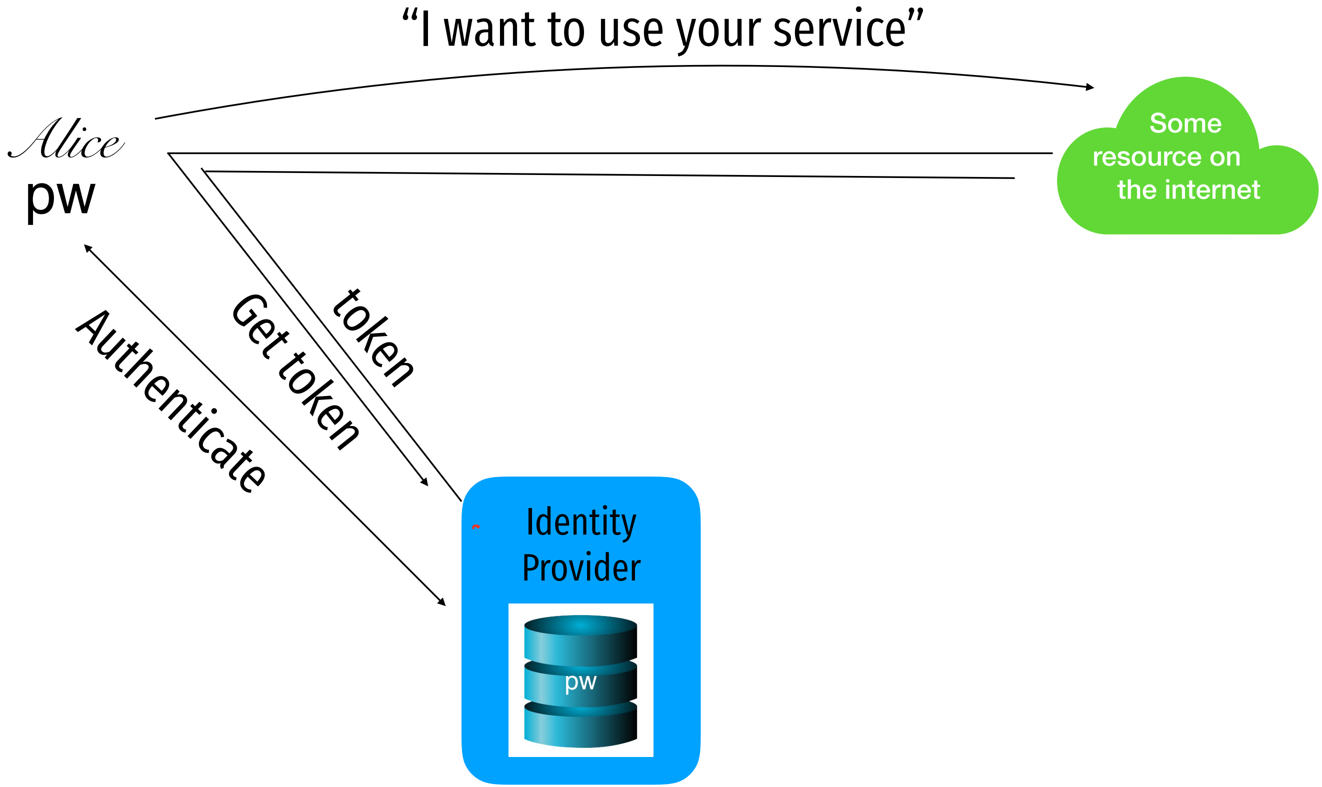
token

Some resource on the internet

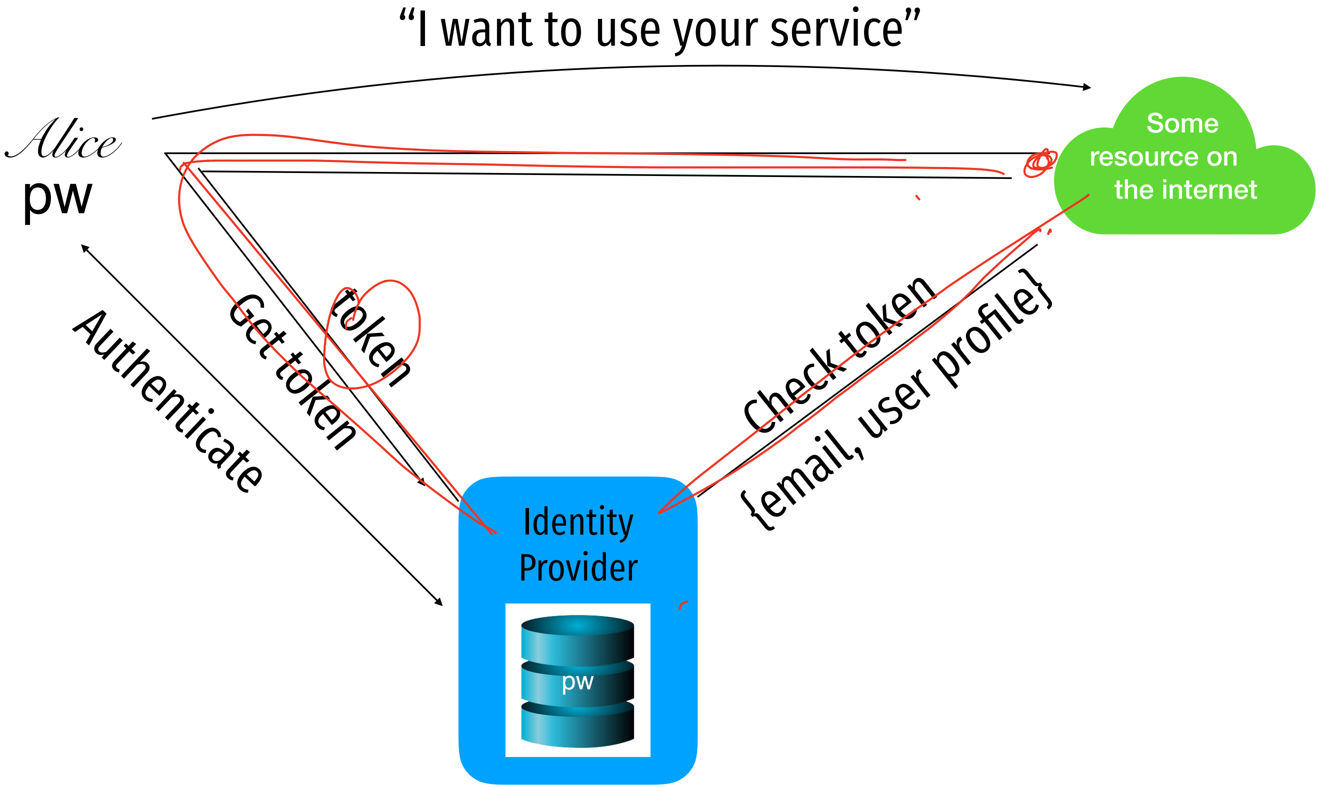
Identity Provider

pw

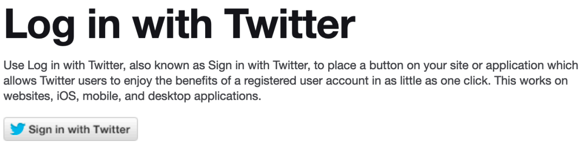
Oauth


"I want to use your service"
Alice
pw
Some resource on the internet
Authenticate
Get token
token
Identity Provider
pw

Oauth

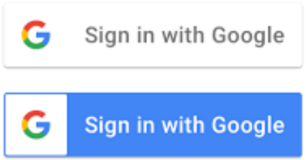

"I want to use your service"
Alice
pw
Some resource on the internet
Authenticate
Get token
token
Check token
{email, user profile}
Identity Provider
pw

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~~

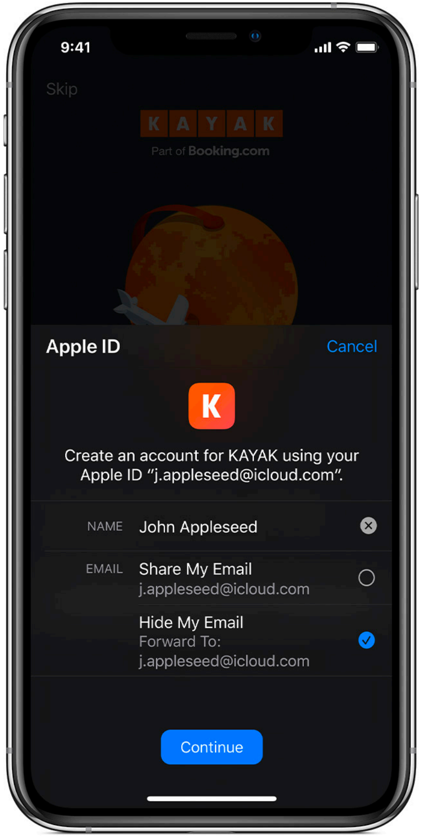
Using Sign in with Apple is quick and easy on any Apple device with the latest software. Make sure you're signed in with your Apple ID on your device.

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.



Create an account for KAYAK using your Apple ID "j.appleseed@icloud.com".

NAME John Appleseed

EMAIL
○ Share My Email
j.appleseed@icloud.com

✓ Hide My Email
Forward To:
j.appleseed@icloud.com

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