CSCI-3403: Cyber Security
Spring 2020

Biljith Thadichi

Department of Computer Science

University of Colorado Boulder

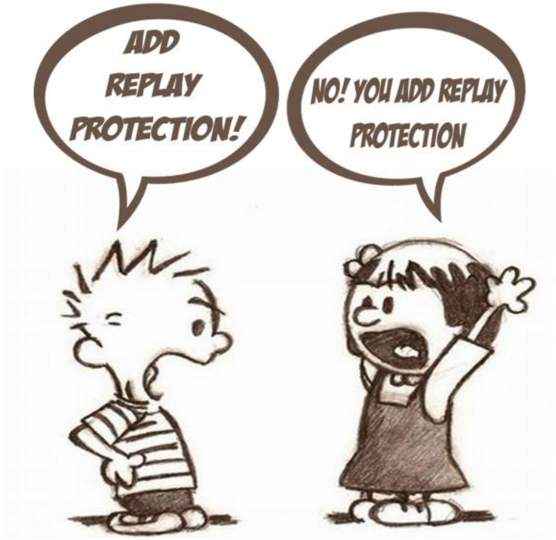
Mid Term FCQs

Week 5

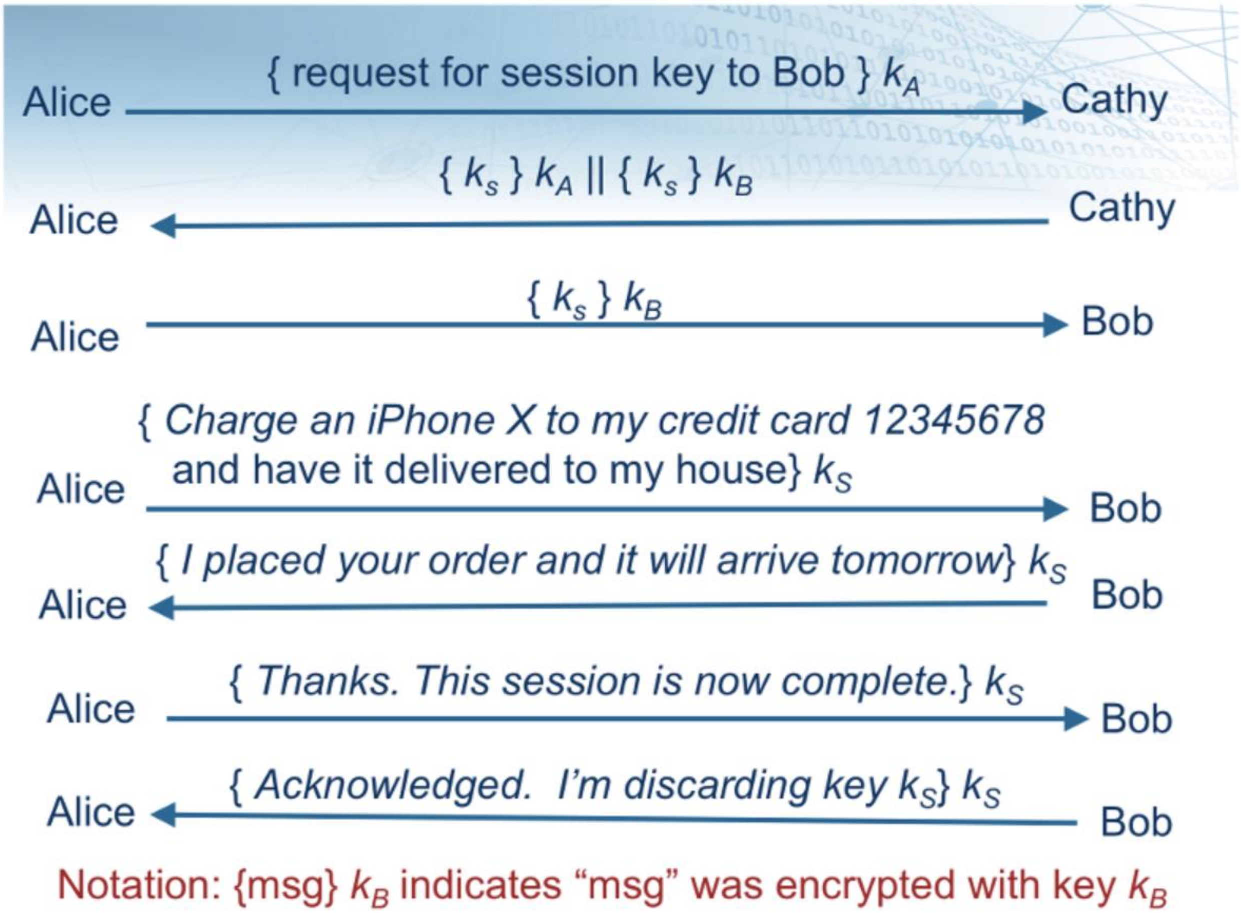
> Replay Attacks

> Salting

> Project 2



Case 1



Alice → Cathy: { request for session key to Bob } k_A

Alice ← Cathy: { k_s } k_A || { k_s } k_B

Alice → Bob: { k_s } k_B

Alice → Bob: { *Charge an iPhone X to my credit card 12345678* and have it delivered to my house} k_S

Alice ← Bob: { *I placed your order and it will arrive tomorrow*} k_S

Alice → Bob: { *Thanks. This session is now complete.*} k_S

Alice ← Bob: { *Acknowledged. I'm discarding key* k_S} k_S

Notation: {msg} k_B indicates "msg" was encrypted with key k_B

Quiz 1

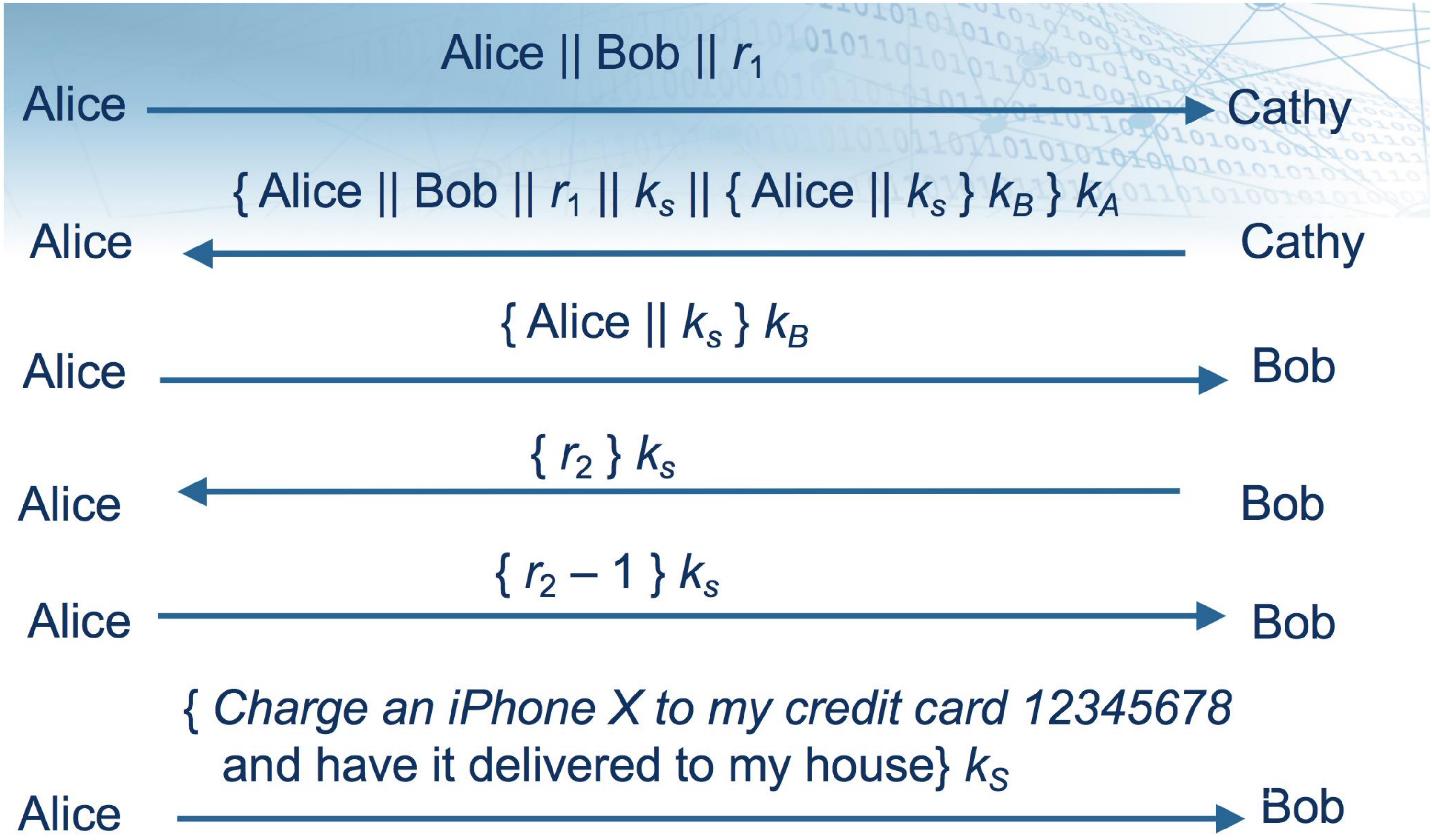
In a replay attack, the adversary must necessarily know how to decrypt the message?

○ True

○ False

Can Eve launch a successful replay attack? As part of the replay attack, will Eve learn Alice's credit card number

Case 2: Needham Schroeder Protocol



Alice || Bob || r_1

Alice → Cathy

{ Alice || Bob || r_1 || k_s || { Alice || k_s } k_B } k_A

Alice ← Cathy

{ Alice || k_s } k_B

Alice → Bob

{ r_2 } k_s

Alice ← Bob

{ $r_2 – 1$ } k_s

Alice → Bob

{ *Charge an iPhone X to my credit card 12345678* and have it delivered to my house} k_S

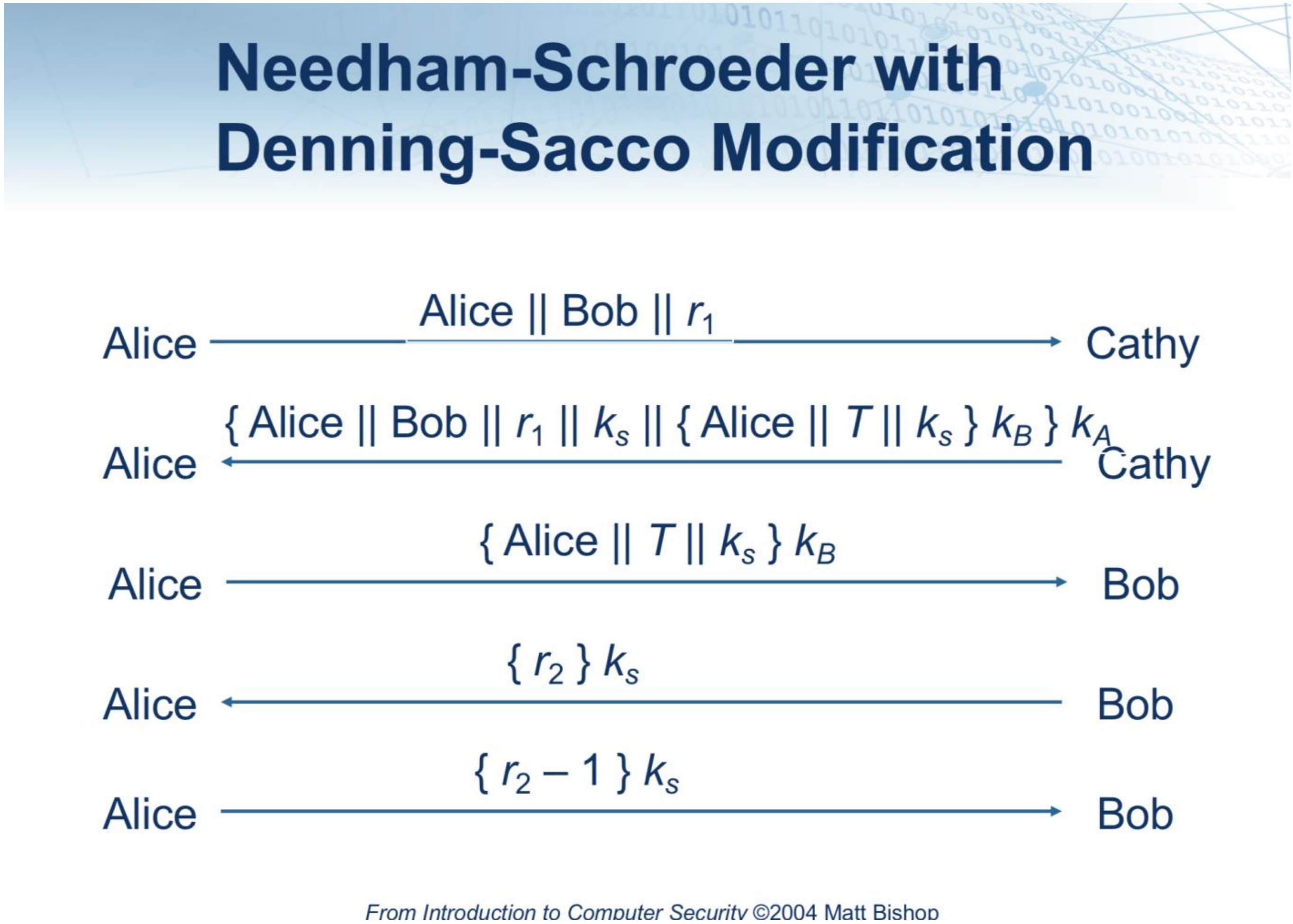
Alice → Bob

Quiz 2

Alice instead uses the key exchange shown prior, can Eve launch a successful replay attack?

If Alice uses the key exchange shown in Slide 2 and Eve has obtained session key Ks, can Eve launch a successful replay attack?

Case 3



Needham-Schroeder with Denning-Sacco Modification

$$\text{Alice} \xrightarrow{\text{Alice} \,||\, \text{Bob} \,||\, r_1} \text{Cathy}$$

$$\text{Alice} \xleftarrow{\{\, \text{Alice} \,||\, \text{Bob} \,||\, r_1 \,||\, k_s \,||\, \{\, \text{Alice} \,||\, T \,||\, k_s \,\}\, k_B \,\}\, k_A} \text{Cathy}$$

$$\text{Alice} \xrightarrow{\{\, \text{Alice} \,||\, T \,||\, k_s \,\}\, k_B} \text{Bob}$$

$$\text{Alice} \xleftarrow{\{\, r_2 \,\}\, k_s} \text{Bob}$$

$$\text{Alice} \xrightarrow{\{\, r_2 - 1 \,\}\, k_s} \text{Bob}$$

From Introduction to Computer Security ©2004 Matt Bishop

Quiz 2

If Alice uses the key exchange shown in Slide 3 and Eve has obtained session key Ks, can Eve launch a successful replay attack?

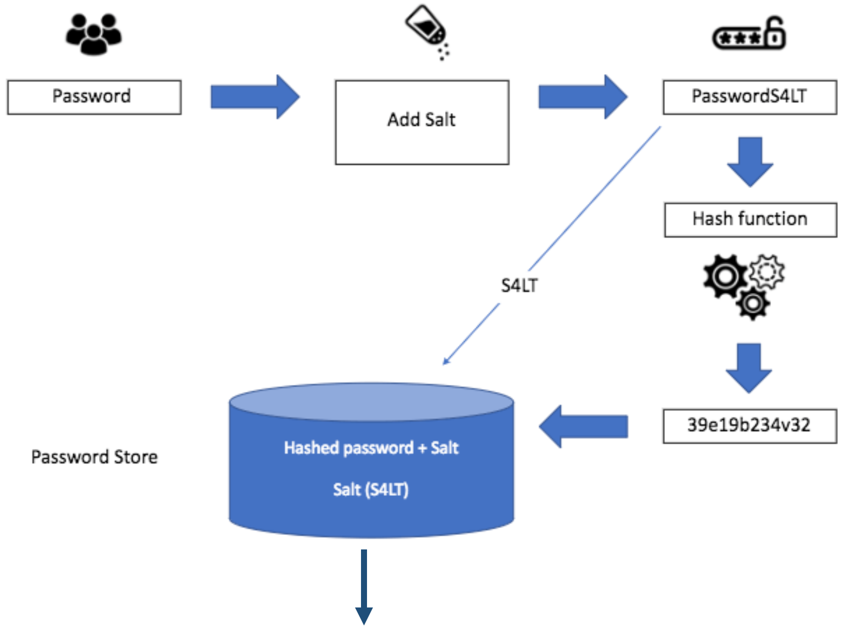
Week 5

> Replay Attacks

> Salting

> Project 2

Salted Password Scheme



User ID	Salt (12 Random bits)	Encrypted Password
Alice	01101000111	H(salt || password)
Bobs	01110111001	H(salt || password)

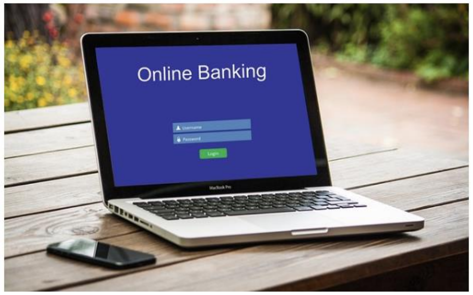
Effect of Salts

	👤	👤	👤	👤
Password	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz	p4s5w3rdz
Salt	–	–	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	lvn49sa	z32i6t0

Online vs Offline Attacks

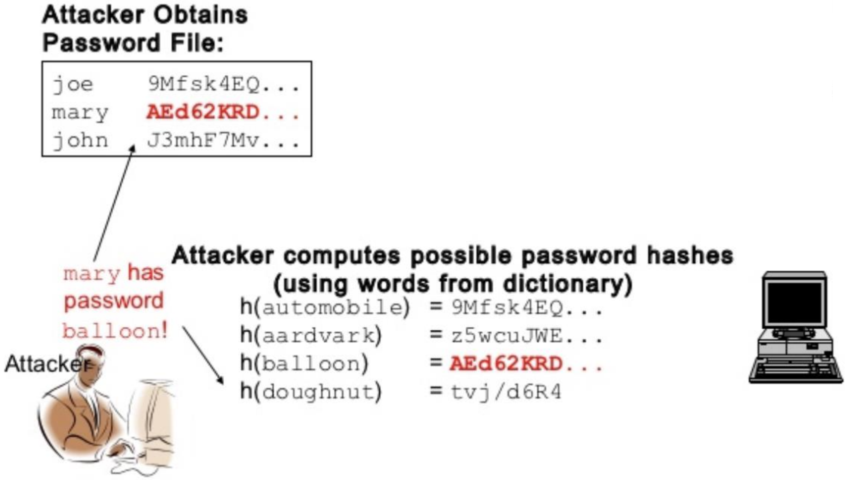
Online Attacks

- Trying a large number of password combinations on the login portal in the hope of getting the right password.
- Limited by speed of the network
- Limited by account lockouts



Offline Dictionary attacks

- A dictionary attack is based on trying all the strings in a pre-arranged listing, typically derived from a list of words such as in a dictionary



Attacker Obtains Password File:

```
joe    9Mfsk4EQ...
mary   AEd62KRD...
john   J3mhF7Mv...
```

mary has password balloon!

Attacker

Attacker computes possible password hashes (using words from dictionary)

```
h(automobile)  = 9Mfsk4EQ...
h(aardvark)    = z5wcuJWE...
h(balloon)     = AEd62KRD...
h(doughnut)    = tvj/d6R4
```

Quiz 1

How much harder does the addition of a salt make it for an attacker who compromises the password file to learn Alice's password?

Assume: salt = 12 bits long, # people $> 2^{12}$

○ Not much

○ Twice as hard

○ 2^{12} times as hard

Quiz 1

How much harder does the addition of a salt make it for an attacker who compromises the password file to learn Alice’s password?

○ Not much

○ Twice as hard

○ 2^{12} times as hard

Quiz 2

How much harder does the addition of a salt (12 random bits) make it for an attacker to carry out an offline dictionary attack?

○ Not much

○ Twice as hard

○ 2^{12} times as hard

Quiz 2

How much harder does the addition of a salt make it for an attacker to carry out an offline dictionary attack?

○ Not much

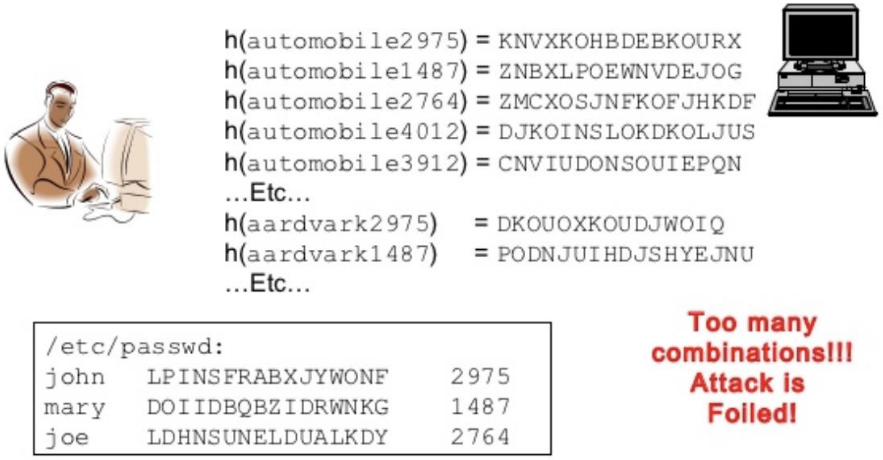
○ Twice as hard

○ 2^{12} times as hard

Salting – Good News!

- Dictionary attack against an arbitrary user is harder
  - **Before salts:** Hash word and compare it with password file
  - **After salts:** Hash words and combos of possible salts
- N word dictionary, k bit salts
  - Attacker must hash $n * 2^k$ strings vs n strings (no salt)

- Offline Dictionary attack foiled!



```
h(automobile2975) = KNVXKOHBDEBKOURX
h(automobile1487) = ZNBXLPOEWNVDEJOG
h(automobile2764) = ZMCXOSJNFKOFJHKDF
h(automobile4012) = DJKOINSLOKDKOLJUS
h(automobile3912) = CNVIUDONSOUIEPQN
...Etc...
h(aardvark2975)   = DKOUOXKOUDJWOIQ
h(aardvark1487)   = PODNJUIHDJSHYEJNU
...Etc...
```

```
/etc/passwd:
john    LPINSFRABXJYWONF     2975
mary    DOIIDBQBZIDRWNKG     1487
joe     LDHNSUNELDUALKDY     2764
```

Too many combinations!!! Attack is Foiled!

Salting – Bad News!

- Ineffective against chosen victim attack
  - Attacker wants to compromise particular account
  - Just hash dictionary words with the victim’s salt
- Attacker’s job becomes harder, not impossible
  - Easy for attacker to compute $2^k * n$ hashes?
  - Then offline dictionary attack is still a threat

Week 5

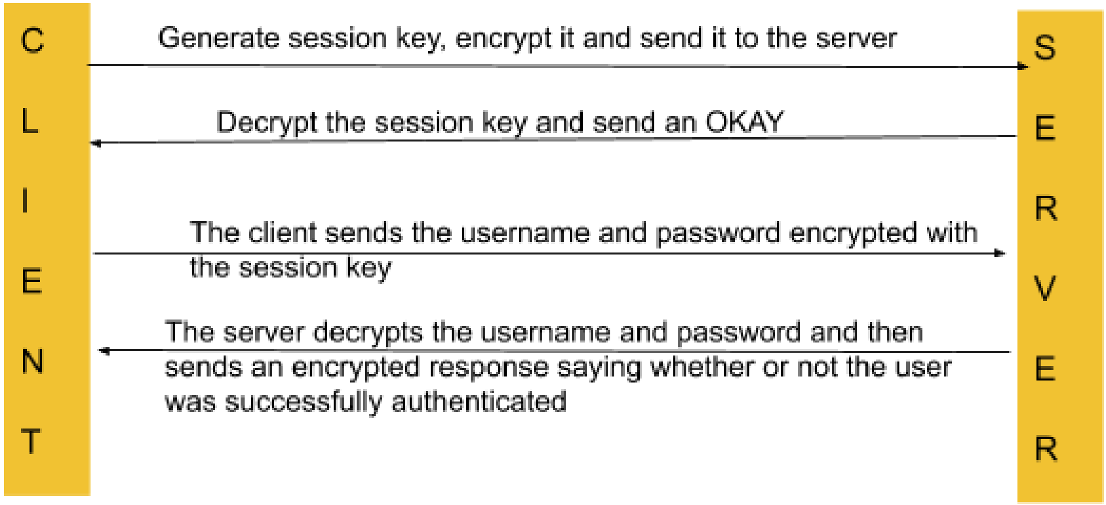
> Replay Attacks

> Salting

> Project 2

Project

- **Goal:** Learn how almost everything **secure** on the internet works, including HTTPS and SSH
- The communication should be guaranteed to be **confidential** and it should have complete **integrity**



Generate session key, encrypt it and send it to the server

Decrypt the session key and send an OKAY

The client sends the username and password encrypted with the session key

The server decrypts the username and password and then sends an encrypted response saying whether or not the user was successfully authenticated

Session keys

- The session key should not be guessable
- What is a good source of a "random" session key?

  `/dev/urandom`

- How do you access /dev/urandom

  Hint: See python's os module

Session Keys

Can we send this session key out in the open?

Of course not!

How can we make sure that nobody other than the server can read this session key?

Public Key Encryption!

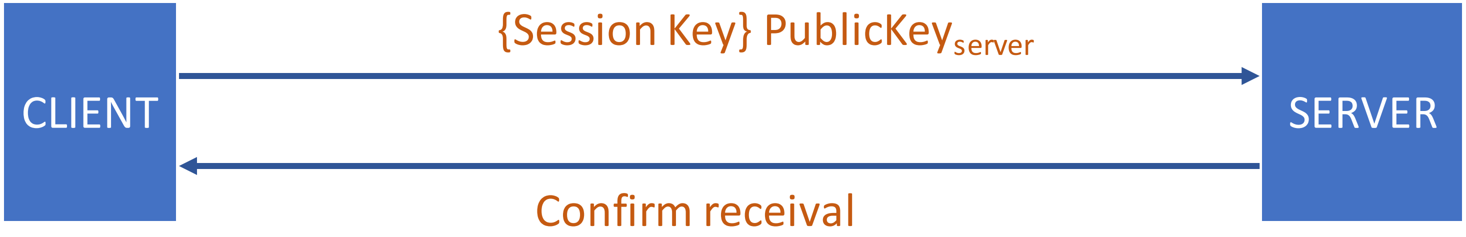
Generating Public – Private Key Pairs

Whose public key should we use to encrypt the message?

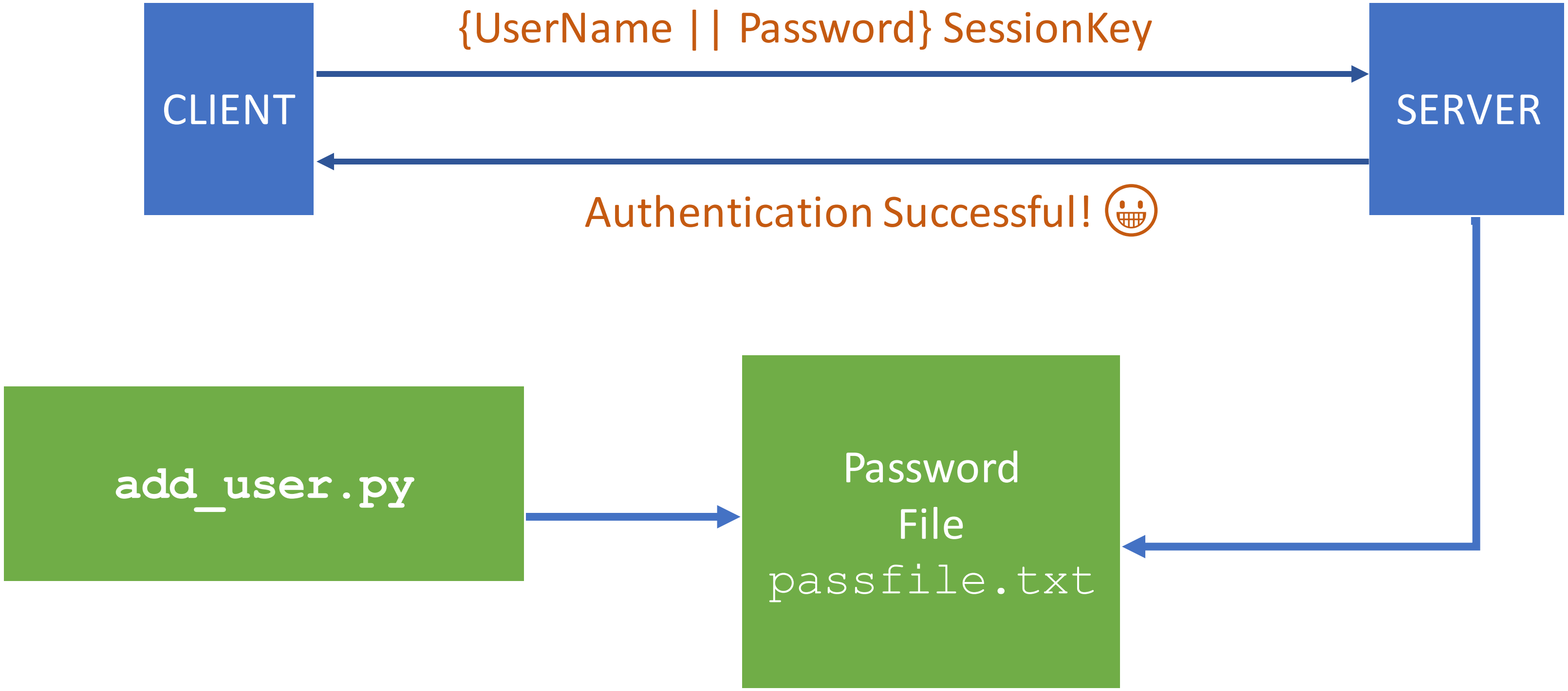
- **Server's!**

How to generate public key private key pairs?

- **`ssh-keygen`**



Send Username Password



CLIENT

{UserName || Password} SessionKey

SERVER

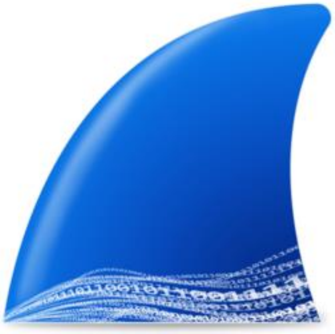
Authentication Successful! 😁

`add_user.py`

Password File `passfile.txt`

Verify your message is encrypted

WIRESHARK



Mid Term FCQs

Your feedback is important!!!

Please and Thank you!