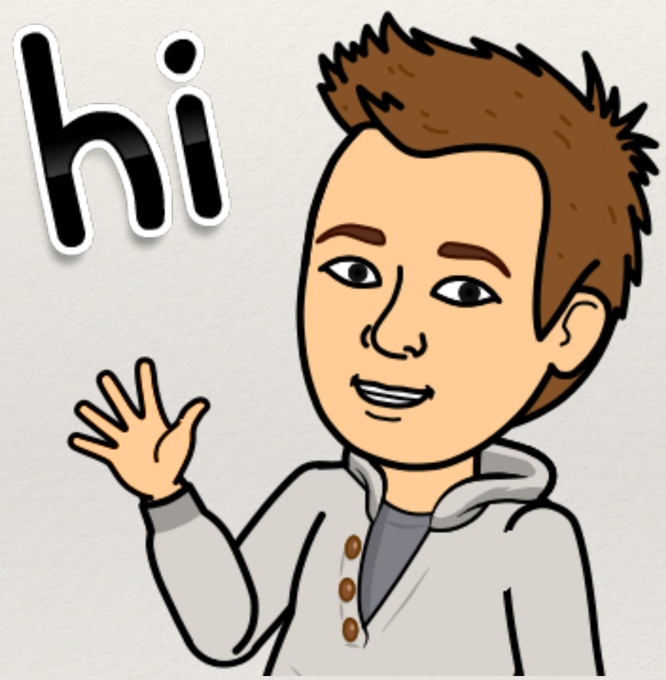
SPA Implementation

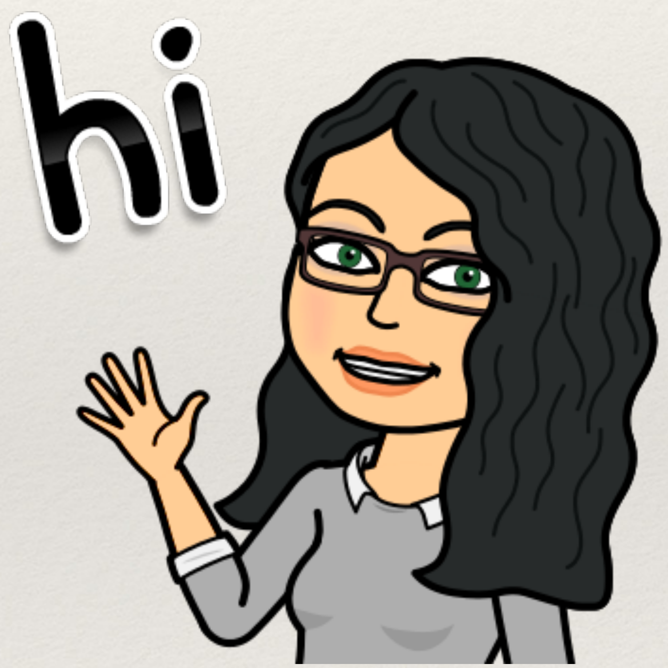
Single password authentication

Table of contents

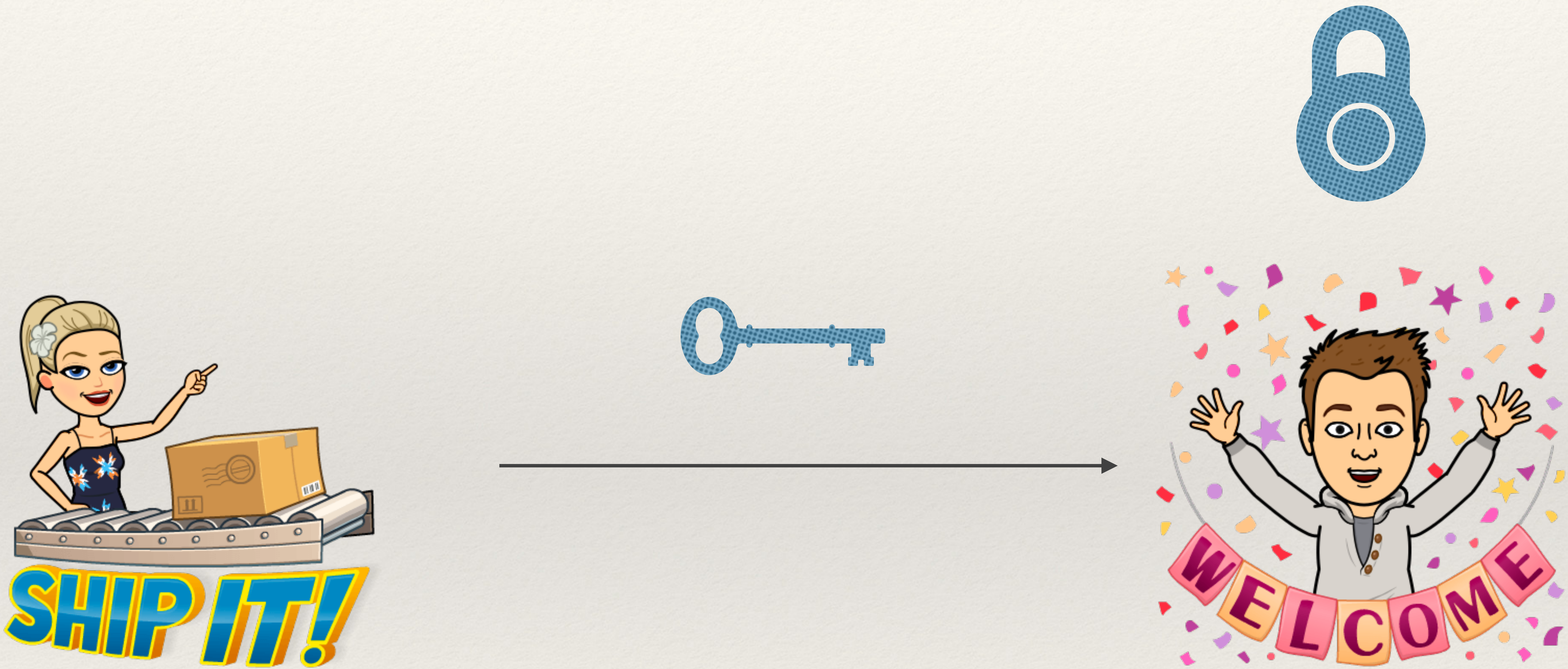
What is SPA?

Say hello to Alice, Bob and Carol

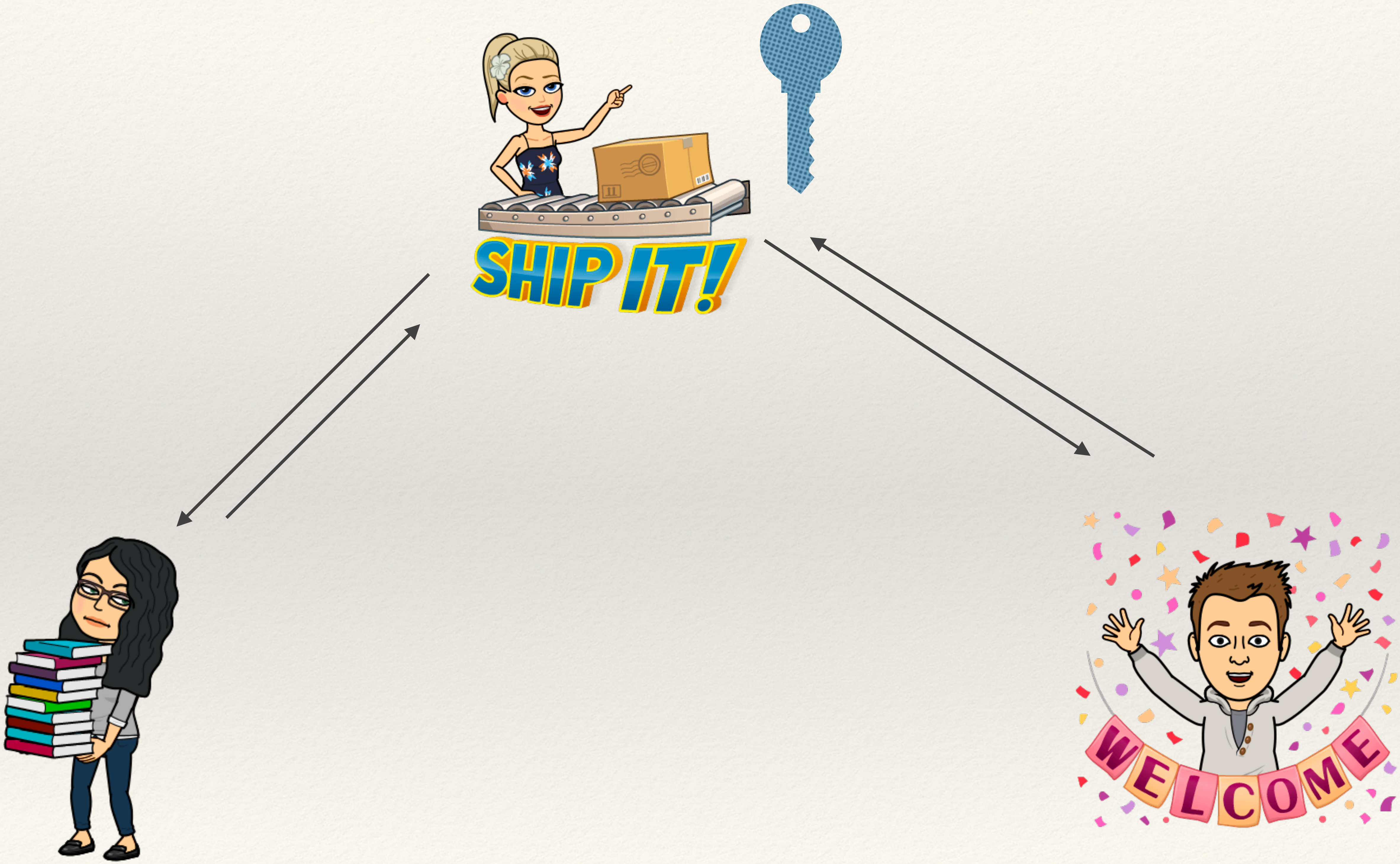




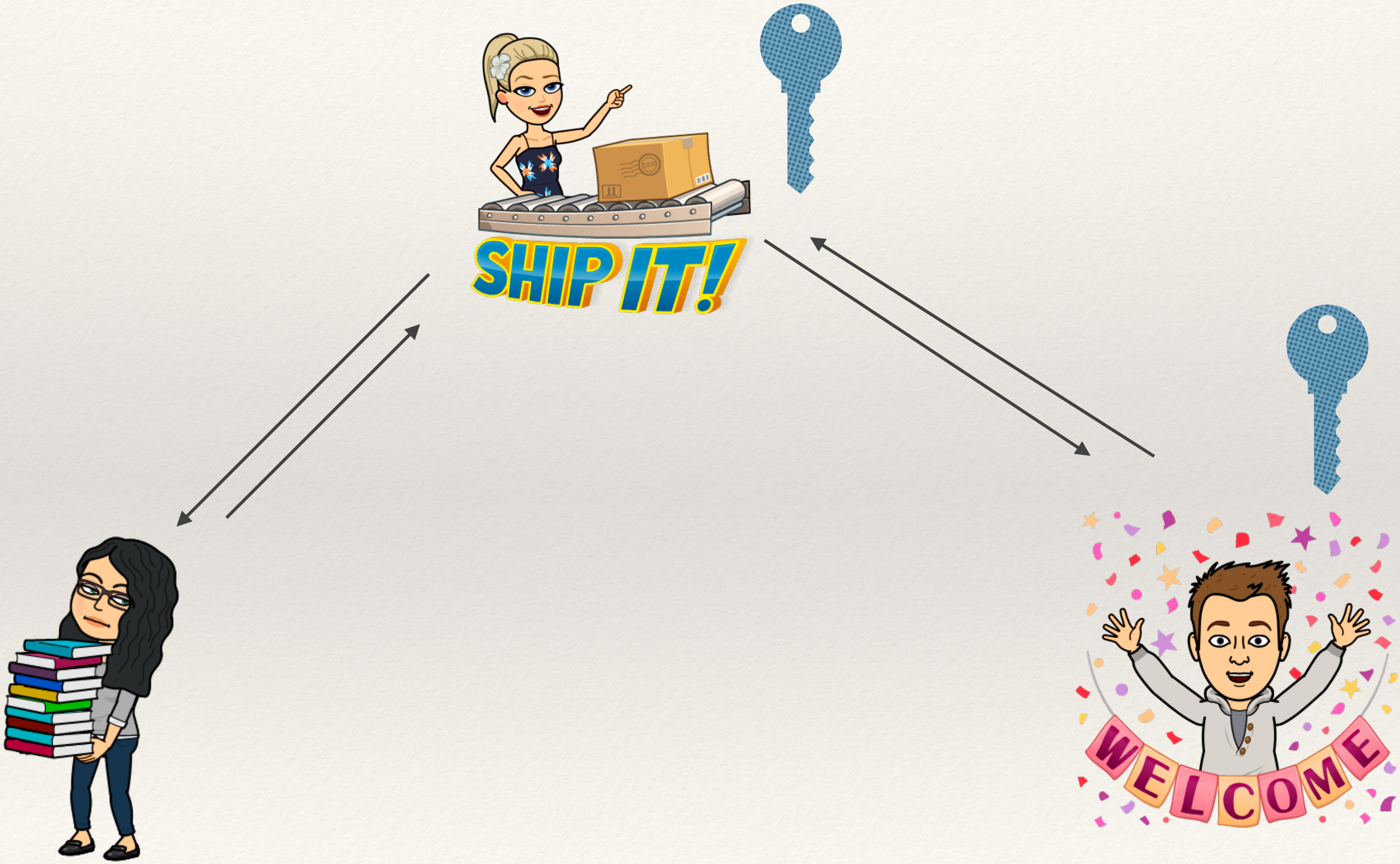
Usual authentication



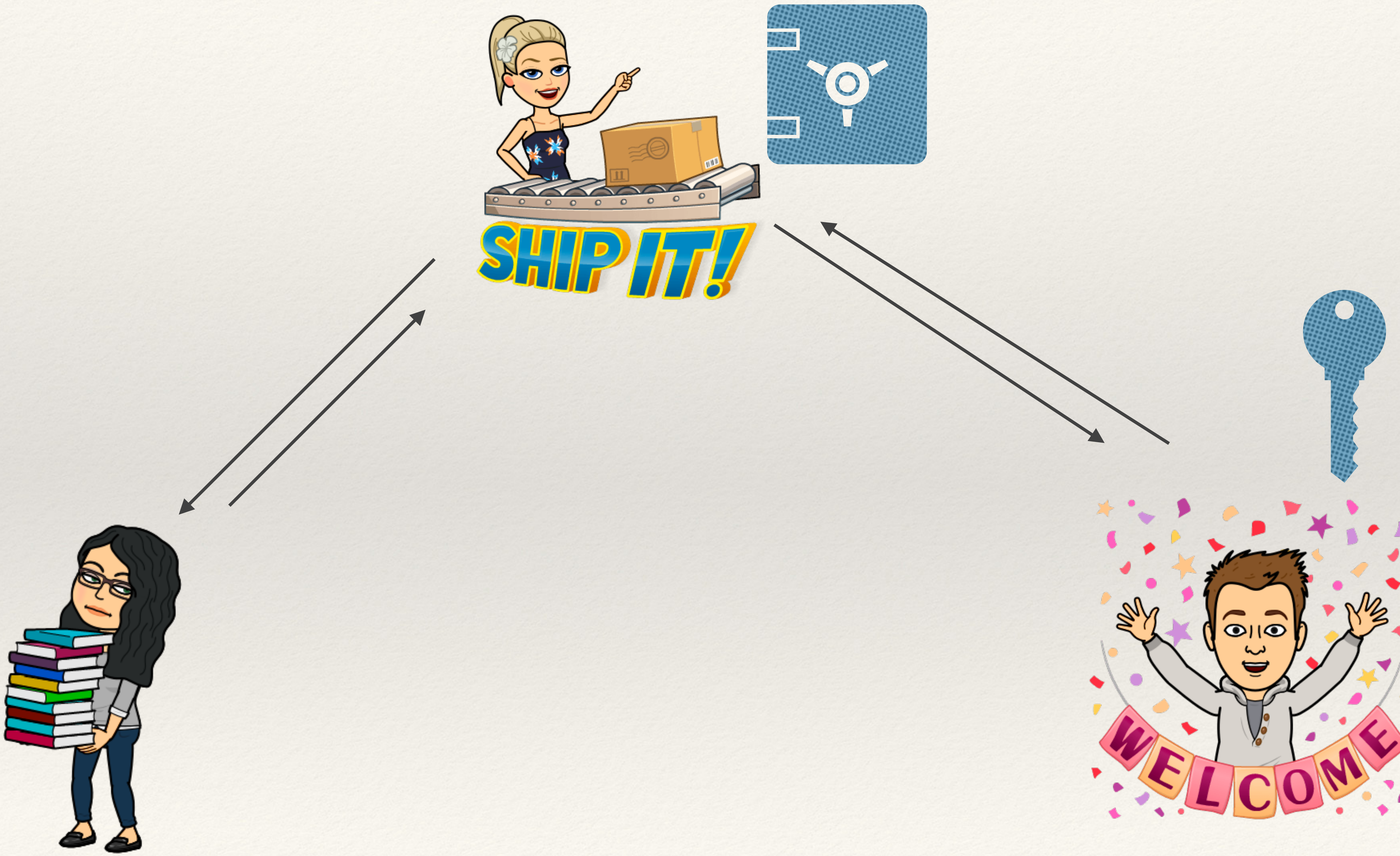
SPA



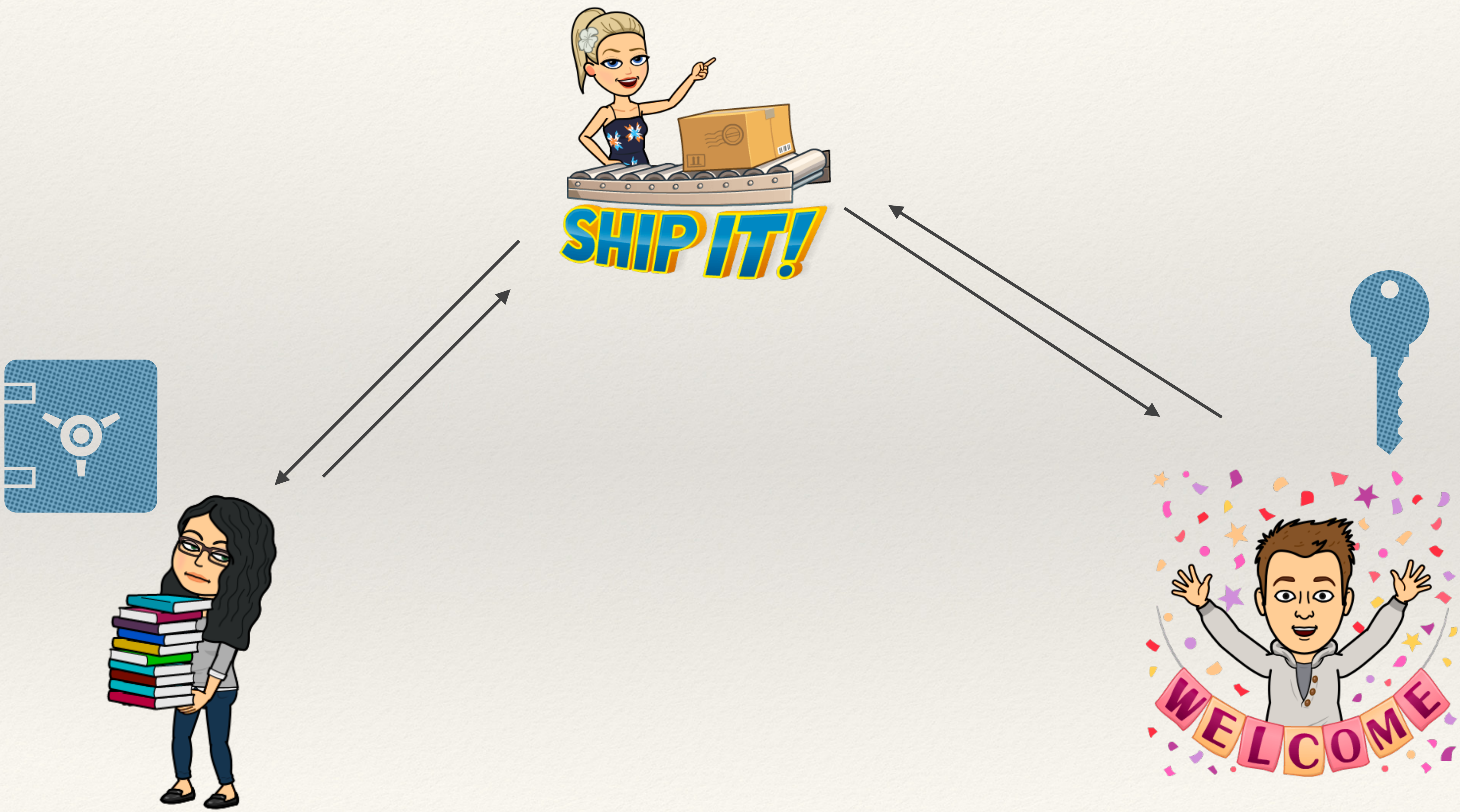
SPA



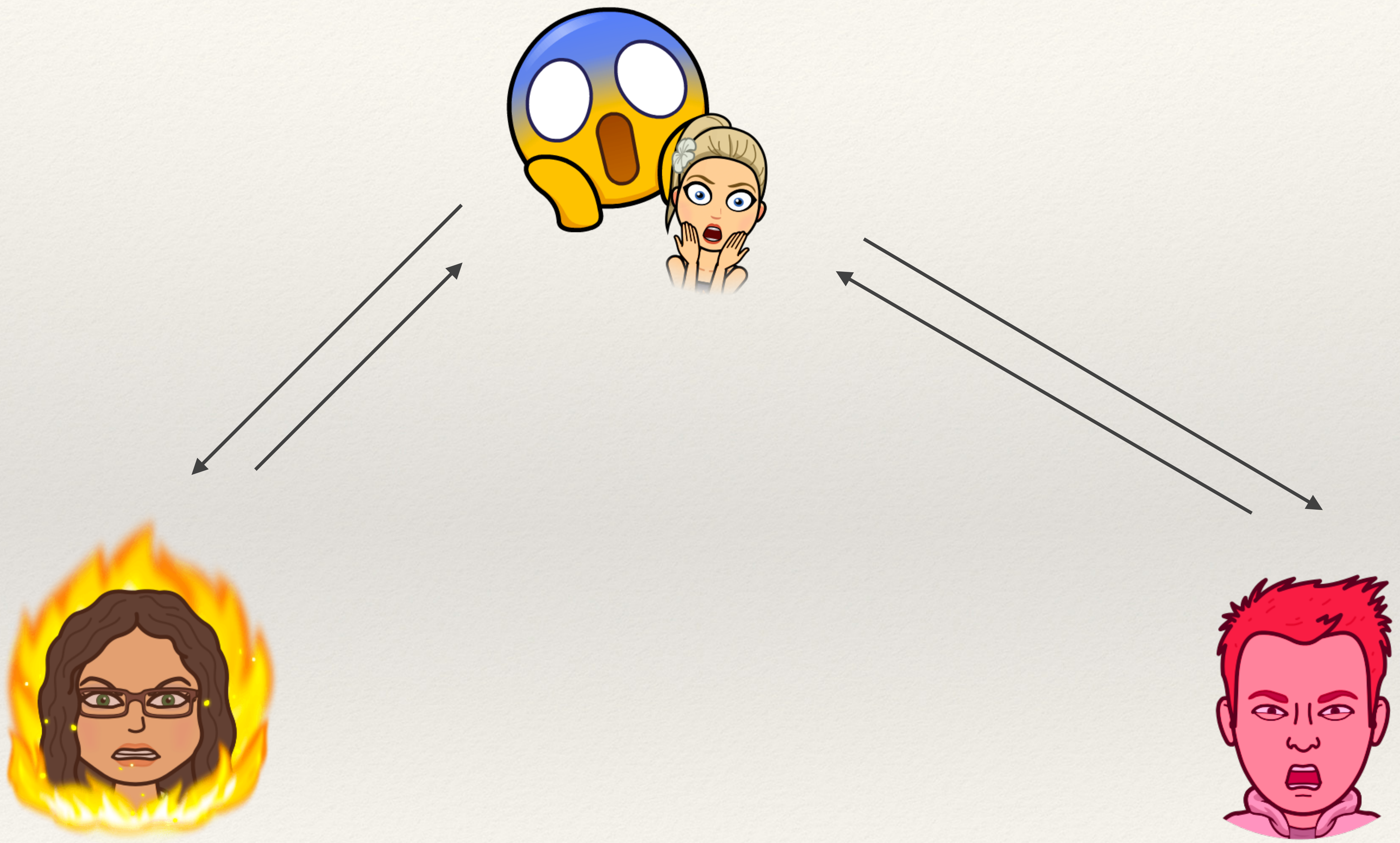
SPA



SPA



Careful!

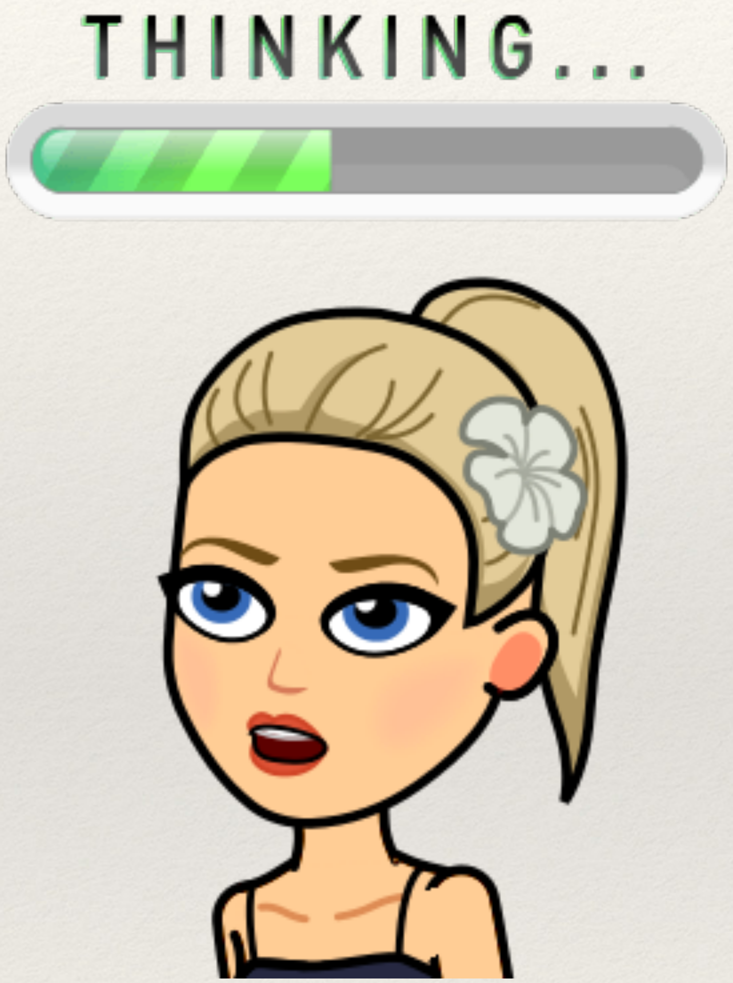


Is this a problem?

- Bob never learns Alice's password
- Alice encrypts her secret before sending to Carol

Why is it useful?

Alice wants to use Facebook



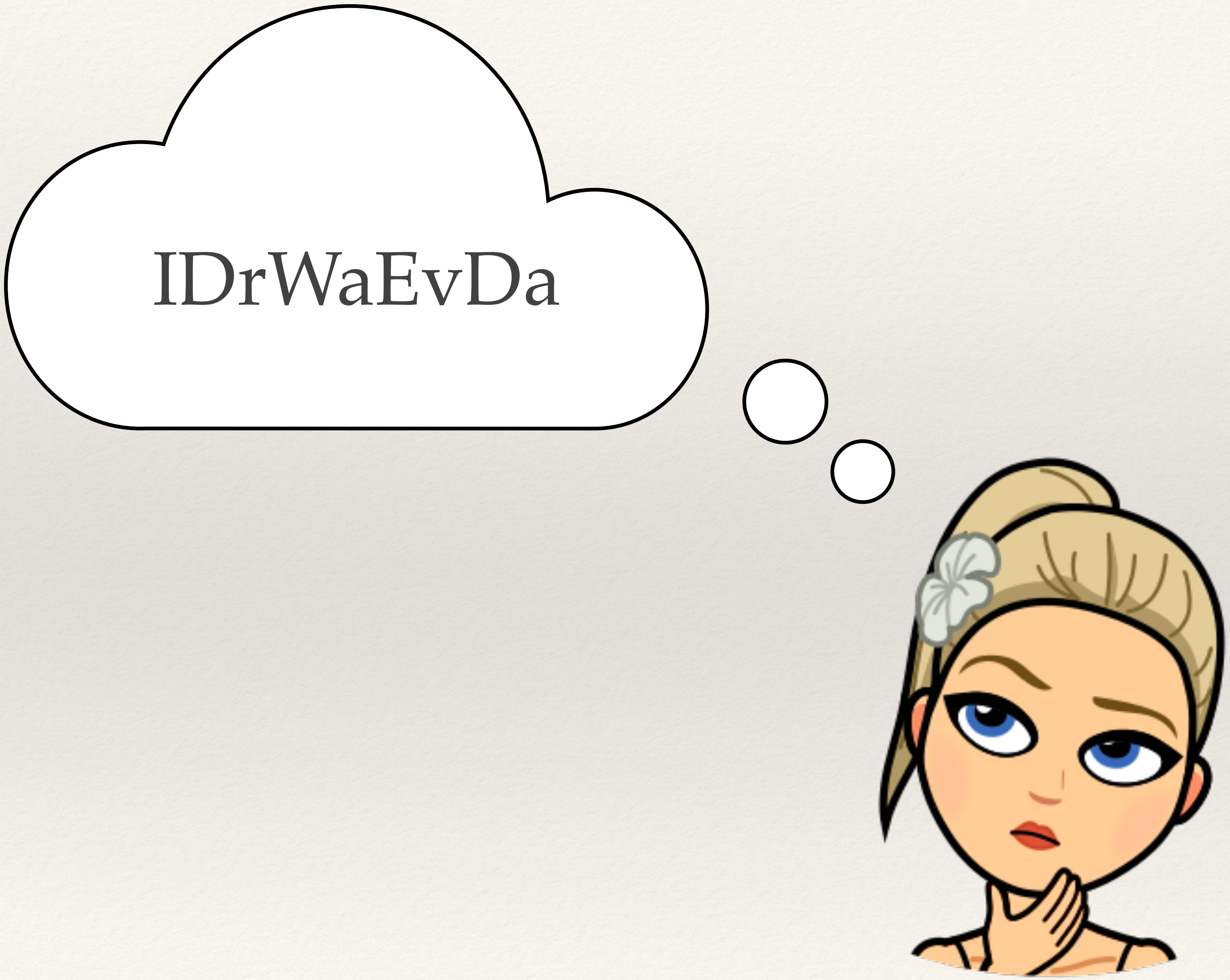
Alice wants to use Facebook

This password is bad

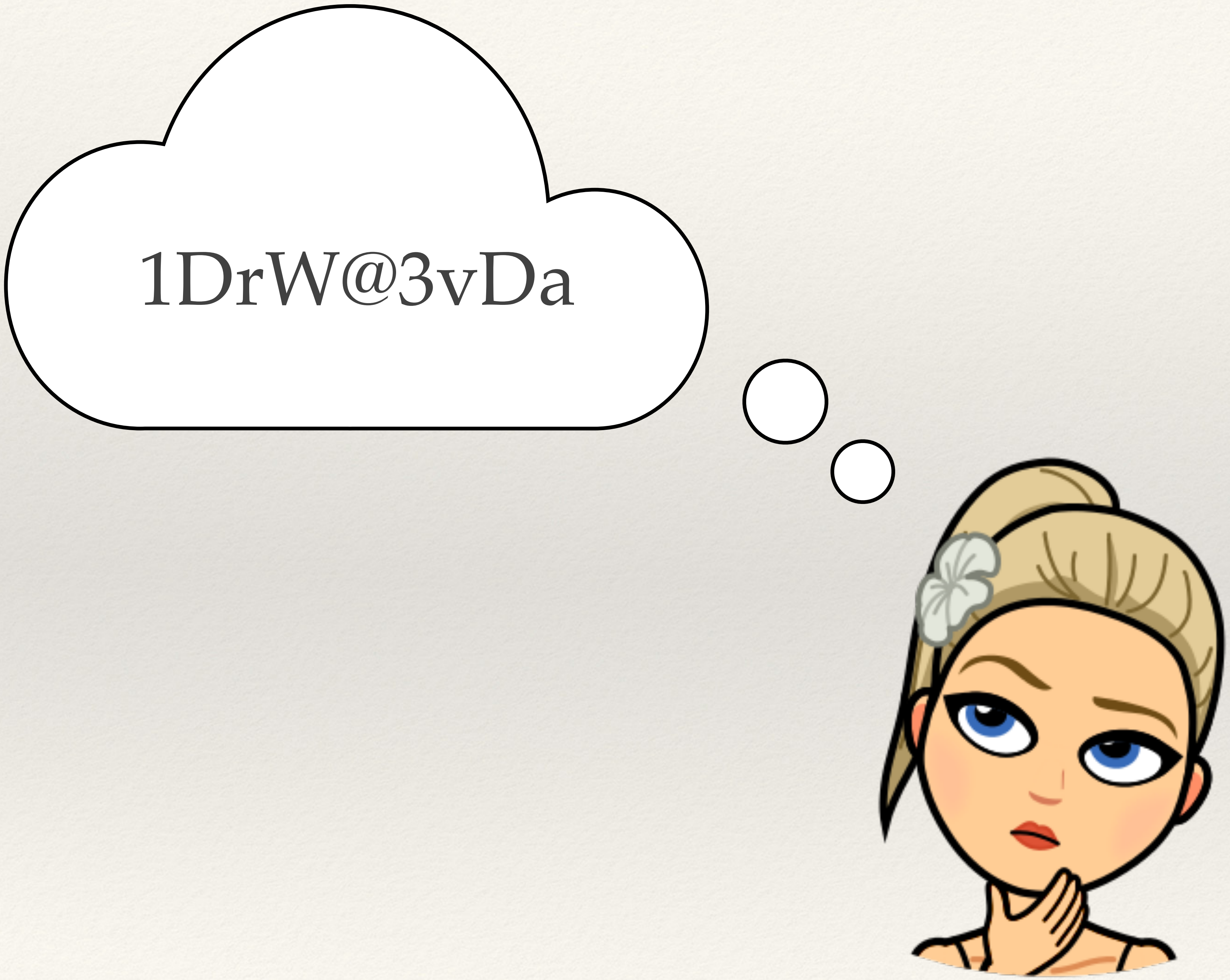
- It includes common words
- It includes elements of Alice's private life
- Can be broken using a dictionary attack

How to generate a good password?

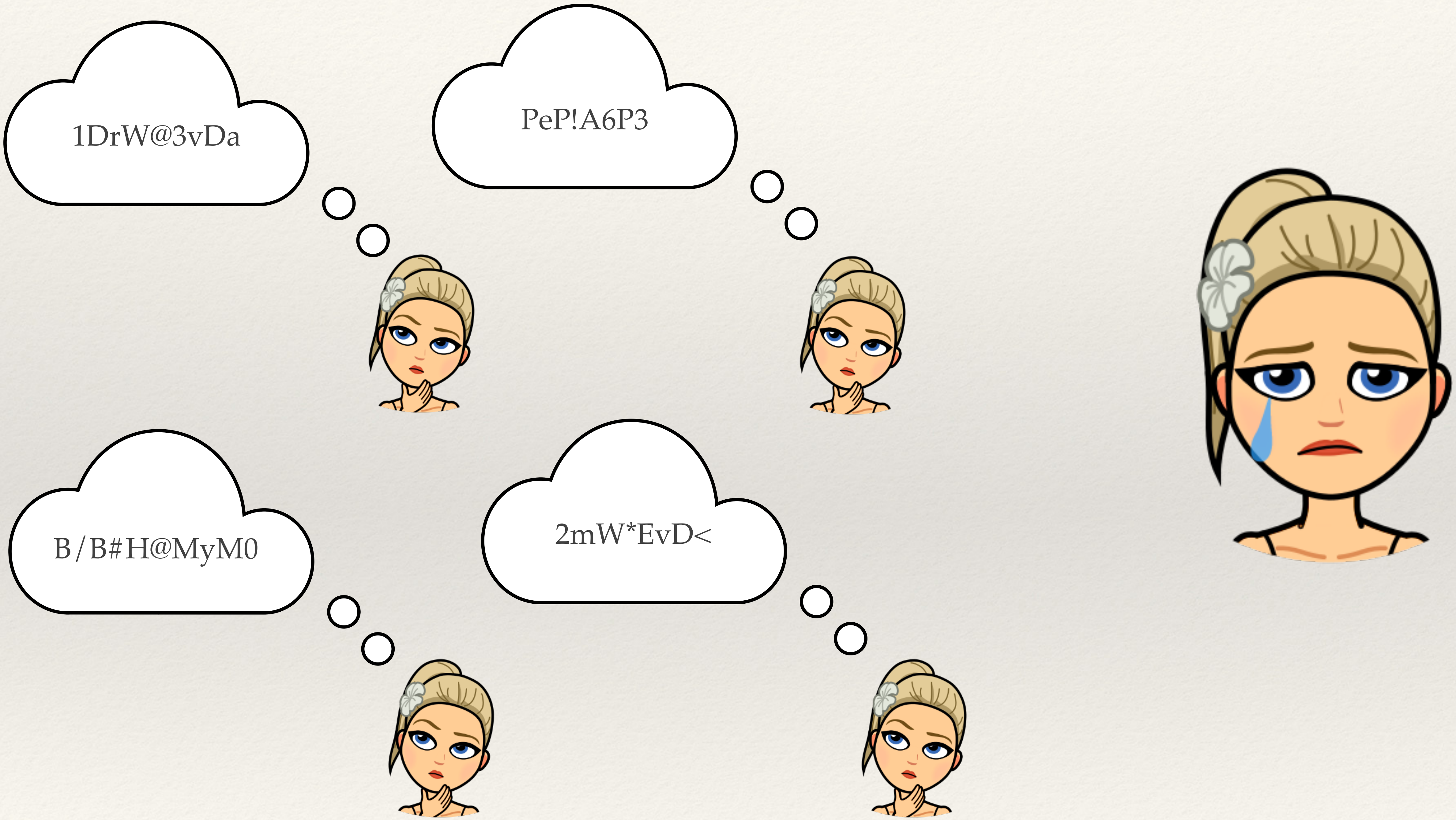
How to generate a good password?



How to generate a good password?

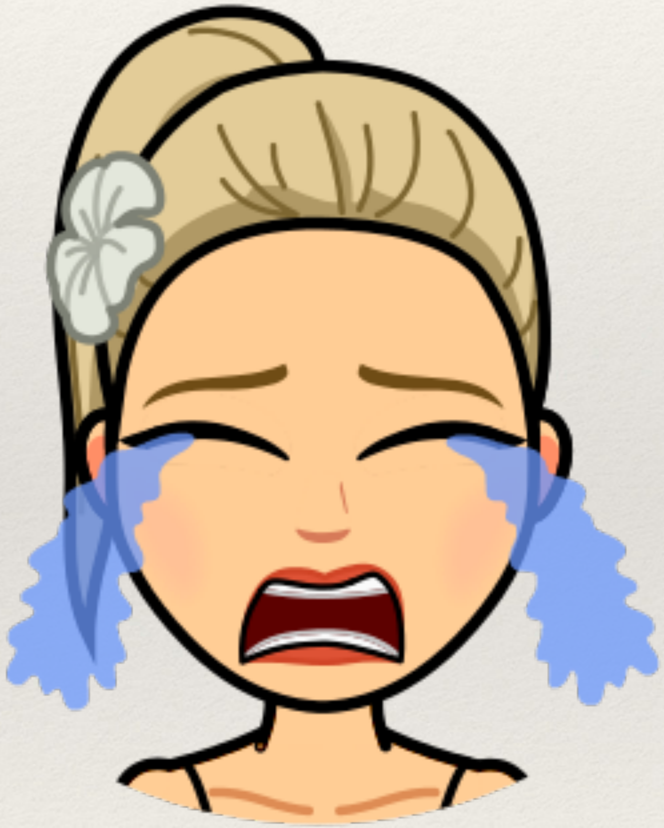


Facebook, Google, Instagram, EPFL,…



A solution: Password manager

What is someone gets the master password?



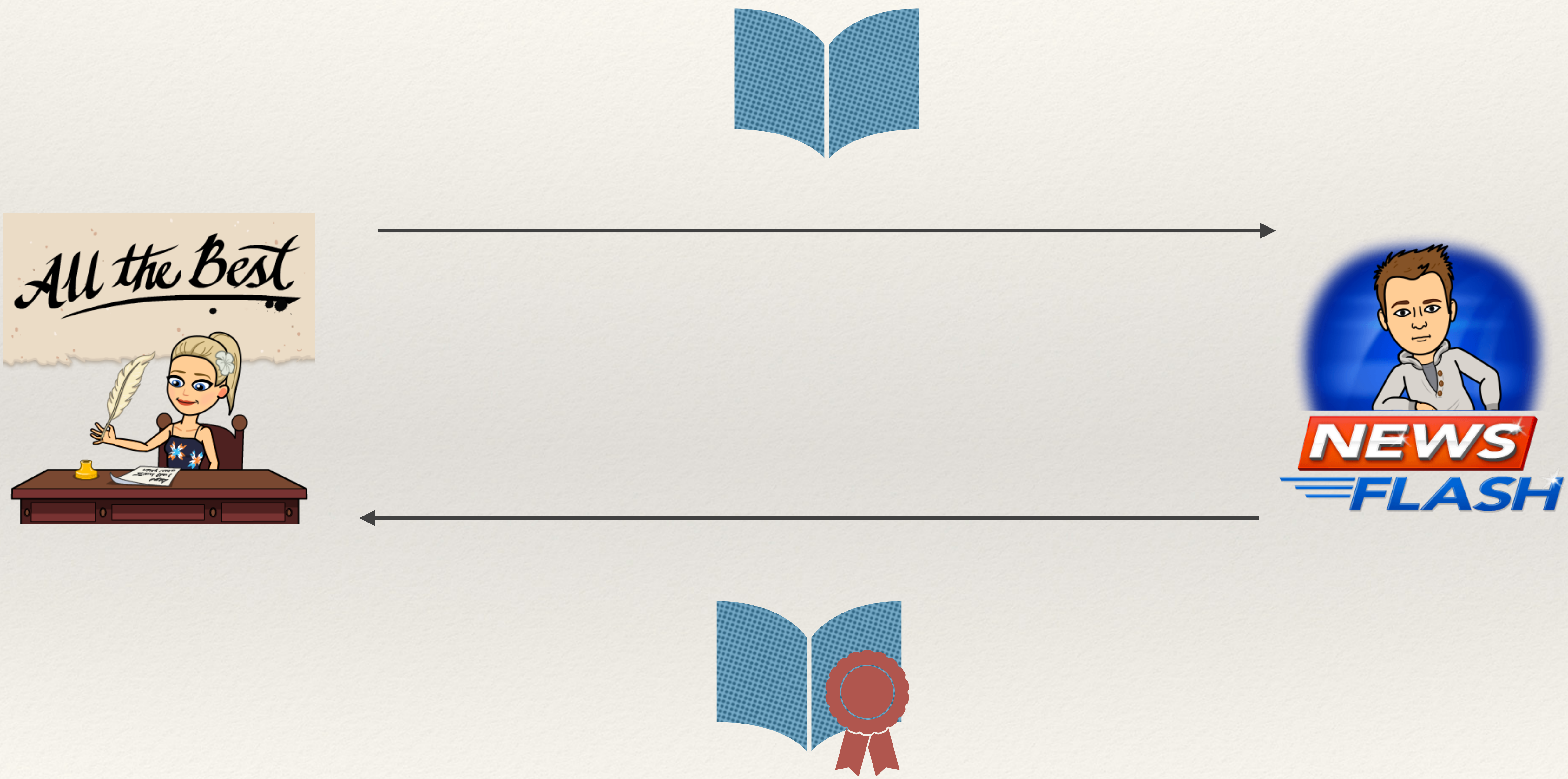
SPA provides a solution

- Alice has a unique password for each services
- Stealing the password is not enough to connect

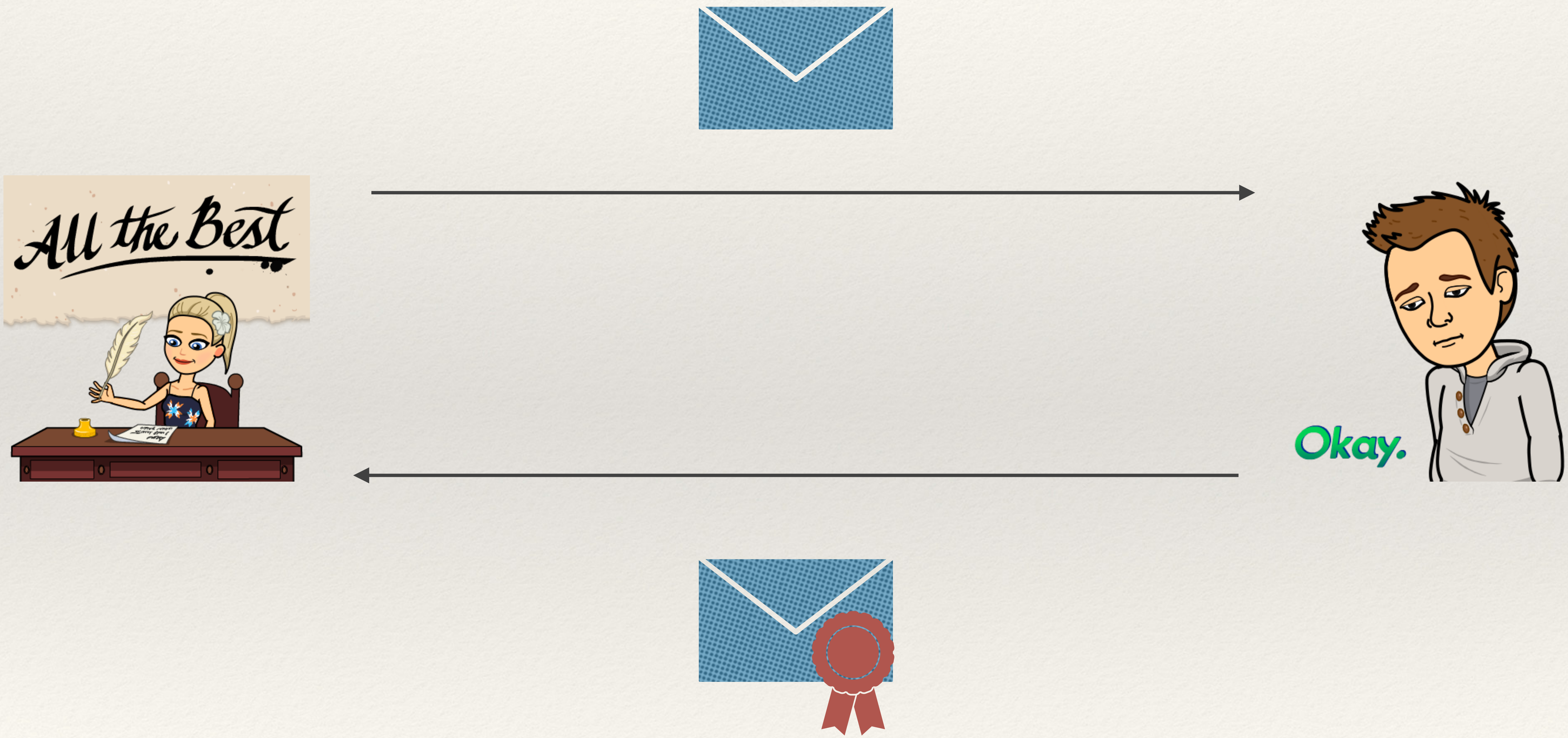
Before starting..

- Cryptography
  - Blind signature
  - Oblivious transfer

Digital signature



Blind signature



Blind signature based on RSA

- Alice has a public key (N,e)
- Bob has the corresponding private key (d)
- Alice sends the message m with a blind factor r

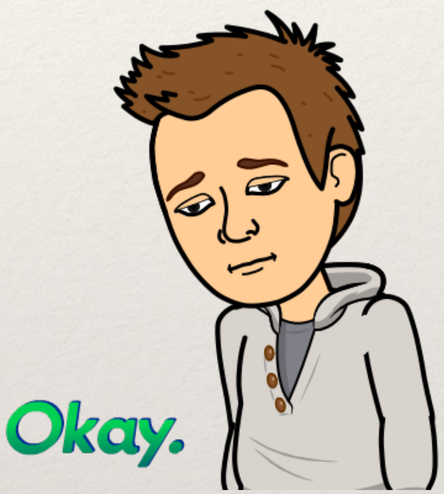
Blind signature

$m' = r^e m \mod N$

m'

$s' = (m')^d \mod N$

$\longrightarrow$



$\longleftarrow$

$s = r^{-1} s' = m^d \mod N$

s'

$m' = rm^d \mod N$

Transfer

2

hey

(1, hello)
(2, hey)
(3, hi)
…

Oblivious transfer by Ogata and Kurosawa

- Alice has a public key (N,e)
- Bob has a private key (d)
- Both share the same random generator G and Hash function H

Oblivious transfer

$$w_j$$

$$Y = r^e H(w_j) \mod N$$

$$(w_i, c_i)$$

$$K_i = (H(w_i))^d \mod N$$

$$E_i = G(w_i||K_i||i) \oplus (0^l||c_i)$$

$$\xrightarrow{\quad Y \quad}$$

$$\xleftarrow[\quad K_j, E_i \quad]{}$$

$$K_j = Y^d \mod N$$

$$(a_i||b_i) = E_j \oplus G(w_j||K_j||i)$$

$$a_i = 0^l \rightarrow b_i = c_j$$

How does it work?

- Four versions of the protocol
  - Cloud SPA
    - Server optimal cloud SPA
    - Storage optimal cloud SPA
    - Privacy optimal cloud SPA
  - Mobile SPA

Server optimal: Registration with Bob

$Alice, password$

Generate ssk, svk

$$\xrightarrow{Alice,\ svk}$$

Server optimal: Registration with Carol

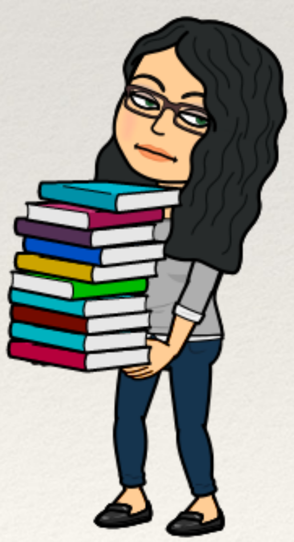
$$Alice, password, ssk, svk$$

Generate bsk

$$id = Hash(Alice, Bob)$$

$$sig = BSign(bsk, Hash(pwd))$$

$$ctext = Encrypt(Hash(sig), ssk)$$

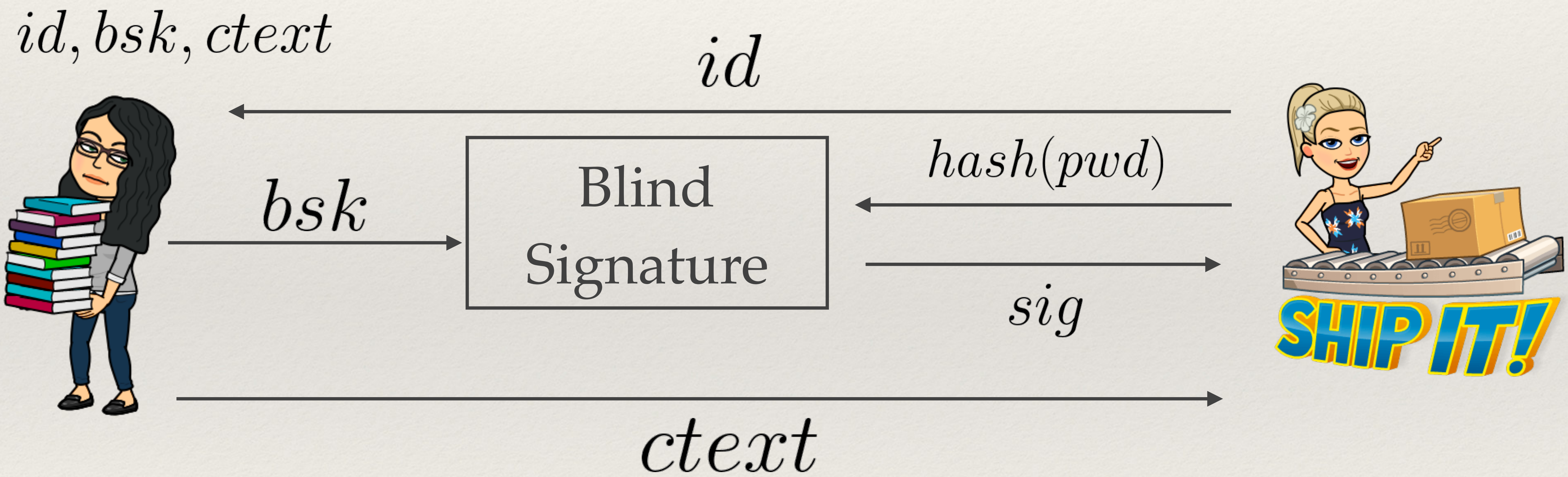


$$\xleftarrow{id, bsk, ctext}$$

Server optimal: Connection with Carol

$$Alice, password$$

$$id = Hash(Alice, Bob)$$



$$ssk = Decrypt(Hash(sig), ctext)$$

Server optimal: Connection with Bob

$Alice, password, ssk$ $\qquad$ $Alice, svk$

$Alice$

$Challenge$

$Response$

$Accept/Deny$

$$Response = sign(ssk, challenge)$$

$$Verify(svk, Response, challenge)$$

Server optimal

- Most efficient for the server
- If Bob is malicious, Alice is safe because Bob does not learn the password
- If Carol is malicious
  - Alice is not anonymous because id = Hash(Alice,Bob)
  - Alice is not unlinkable because id is related to Alice

Storage optimal: Registration with Bob

$Alice, password$

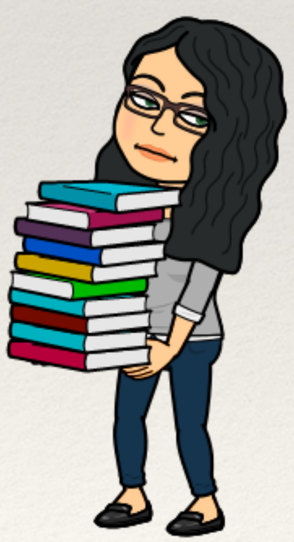
Generate ssk, svk, bsk

$$\xrightarrow{Alice, svk, bsk}$$

Storage optimal: Registration with Carol

$$Alice, password$$

$$\text{Generate } ssk, svk, bsk$$
$$id = BSign(bsk, Hash(pwd))$$
$$ctext = Encrypt(Hash(pwd), ssk)$$



$$id, ctext$$

Storage optimal: Connection with Bob
Step 1

$Alice, password$ $\qquad$ $Alice, svk, bsk$

$\xrightarrow{Alice}$

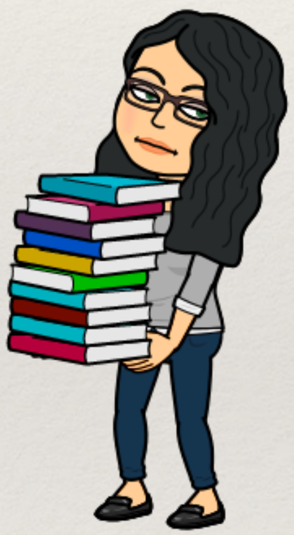
$\xrightarrow{hash(pwd)}$ Blind Signature $\xleftarrow{bsk}$

$\xleftarrow{id}$

Storage optimal: Connection with Carol

$id, ctext$

$Alice, password, id$



id

$ctext$

$$ssk = Decrypt(Hash(pwd), ctext)$$

Storage optimal: Connection with Bob
Step 2

$Alice, password, ssk$ $\qquad\qquad$ $Alice, svk$

$Alice$ $\longrightarrow$

$\longleftarrow$ $Challenge$

$Response$ $\longrightarrow$

$\longleftarrow$ $Accept/Deny$

$$Response = sign(ssk, challenge)$$

$$Verify(svk, Response, challenge)$$

Storage optimal

- Most efficient for the storage
- If Bob is malicious, Alice is safe because Bob does not learn the password
- If Carol is malicious,
  - Alice is anonymous because id is a signature
  - Alice is still linkable because Carol has access to the id

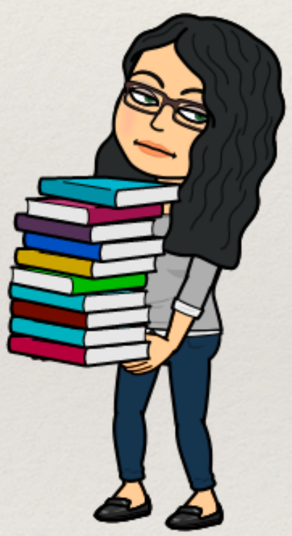
Privacy optimal

- Registration is similar to Storage optimal
- Connection with Bob is similar to Storage optimal
- Connection with Carol uses Oblivious transfer

Privacy optimal: Connection with Carol

$id, ctext$

$Alice, password, id$



$\xrightarrow{id,\, ctext}$ Oblivious transfer $\xleftarrow{id}$ $\xrightarrow[ctext]{}$

$$ssk = Decrypt(Hash(pwd), ctext)$$

Privacy optimal

- If Carol is malicious
  - Alice is anonymous (from Storage optimal)
  - Alice is unlinkable because of the Oblivious transfer

Mobile SPA

- It does not require a cloud service but a trusted mobile device
- It permits to connect from anywhere, especially if the terminal is untrusted

Mobile SPA: Registration with Bob

$Alice, password$

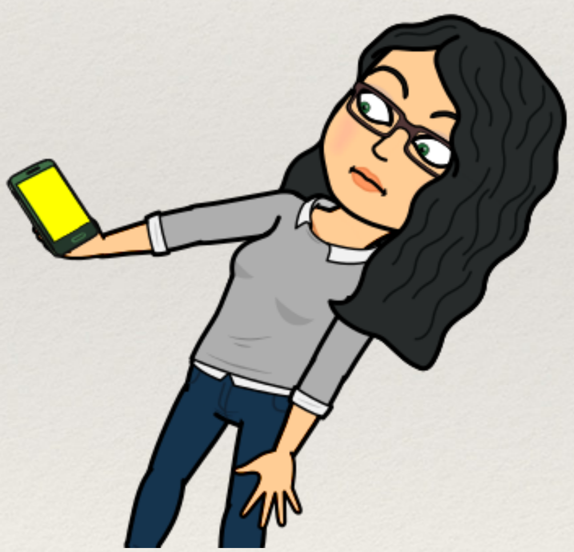
Generate K

$$\xrightarrow{\quad Alice, K \quad}$$

Mobile SPA: Registration with Carol

$$Alice, password, K$$

$$ctext = encrypt(hash(pwd), K)$$



$$ctext$$

Mobile SPA: Connection with Bob

$Alice, password, ctext$ $\qquad$ $Alice, K$

$Alice$

$Challenge$

$Response$

$Accept/Deny$

$$K = decrypt(Hash(pwd), ctext)$$

$$Response = Trim(Mac(K, chal))$$

$$Verify(Trim(Mac(K, challenge)), Response)$$

Mobile SPA

- If Bob is malicious, Alice is safe because Bob does not learn the password
- If the terminal is malicious, Alice is safe because she does not provide her password
- If Carol becomes malicious (device stolen), Alice is safe because K is encrypted and her username and here password are not stored in the device

How do I implemented it?

- How to create
  - Carol
  - Bob
  - Alice
  - Connection between them

Cryptography

- Symmetric algorithms
- Asymmetric algorithms
- Hash
- Oblivious transfer

Symmetric algorithms

- Encryption/decryption with one time pad
- MAC with HMacSHA256

Asymmetric algorithms

- RSA
- Digital signature with RSA-PKCS#1-V1.5
- Blind signature

Hash

- SHA256

Oblivious transfer

- Sender class from Map<Wi,Ci>
  - Automatically generates Ei and Ki
- Receiver class from single Key Wj
  - Functions to generate Y and find Cj

Implementation of Carol - Cloud

- Database
    - To store data from Alice
- Java software
    - To access the database
    - To execute cryptographic schemes (blind signature, oblivious transfer, …)

What does it look like?

- Database implemented using Microsoft Azure
- Java Software
  - JDBC driver to connect to the database
  - Create server using SSL socket on an unused port to receive Alice connection

Implementation of Carol - Mobile

- Android application
- Based on existing application from Google
- Find QR Code with the camera, read the corresponding challenge et display the response
- Open SSL socket during registration phase

Implementation of Bob

- Website based on Java EE
- Run with TomCat
- Wait for connection on two different addresses
  - https://128.178.73.85:8443/spa/register during registration
    - Accessed from Java Software (POST)
  - https://128.178.73.85:8443/spa/connect during connection
    - Accessed from browser (GET and POST)

Implementation of Alice

- Java Software using SWT
- Alice types her data (Protocol, username, password, website)

Connection from Alice

- Bob
  - HTTP Request
- Carol
  - Exchange data with SSL socket

Time for a demo

Is it good?

- Performance test
  - Compare each protocol
  - Compare RSA key-length (1024 vs 2048)

Server optimal cloud SPA

- Registration is similar with Bob and Carol
- Connection is faster with Bob
- Generally fast

	Alice-Bob	Alice-Carol	Total
Registration	15	35	50
Connection	22	218	245

Storage optimal cloud SPA

- Registration and total time are similar than Server optimal
- Connection faster with Carol

	Alice-Bob	Alice-Carol	Total
Registration	29	36	65
Connection	237	41	284

Privacy optimal cloud SPA

- When the database is empty, similar result to Storage optimal
- Connection time grows really fast relatively to the database size

	Alice-Bob	Alice-Carol	Total
Registration	26	38	64
Connection (0 element)	225	48	279
Conn. (200 elements)	218	277	505
Conn. (500 elements)	223	1773	2004
Conn. (1000 elements)	229	3164	3407

Mobile SPA

- Generally fast
- Total time is harder to estimate since it requires user interactions

	Alice-Bob	Alice-Carol	Total
Registration	21	127	148
Connection	15	1362	-

1024 vs 2048 bits key-length

- Compare connection time of Server optimal and Storage optimal
- 2048 is longer but difference is not significant

1024	Alice-Bob	Alice-Carol	Total
Server optimal	22	218	245
Storage optimal	237	41	284

2048	Alice-Bob	Alice-Carol	Total
Server optimal	23	258	287
Storage optimal	282	43	338

Performance conclusion

- 2048 is better than 1024
- Storage optimal is better than Server optimal because it provides anonymity
- Privacy optimal is useless because total time grows to quickly
- Mobile SPA provides good performance

Conclusion

- SPA is an authentication which permits a user to connect to many services using a unique password
- Four versions: Server optimal, Storage optimal, Privacy optimal and Mobile SPA
- Implementation in Java
  - Java software
  - Android application
  - Website
- Good performance except for Privacy optimal

Questions?