Computer Security (COM-301)

Authentication

Basics and passwords

Carmela Troncoso

SPRING Lab

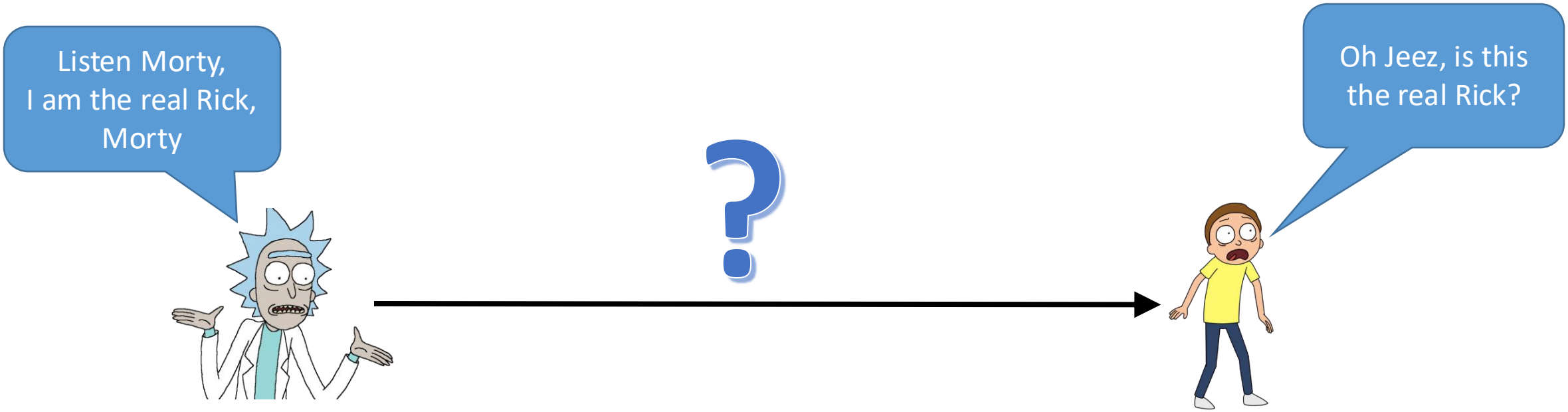
carmela.troncoso@epfl.ch

Some slides/ideas adapted from: Tuomas Aura, Yoshi Kohno, Trent Jaeger

What is authentication?

AUTHENTICATION

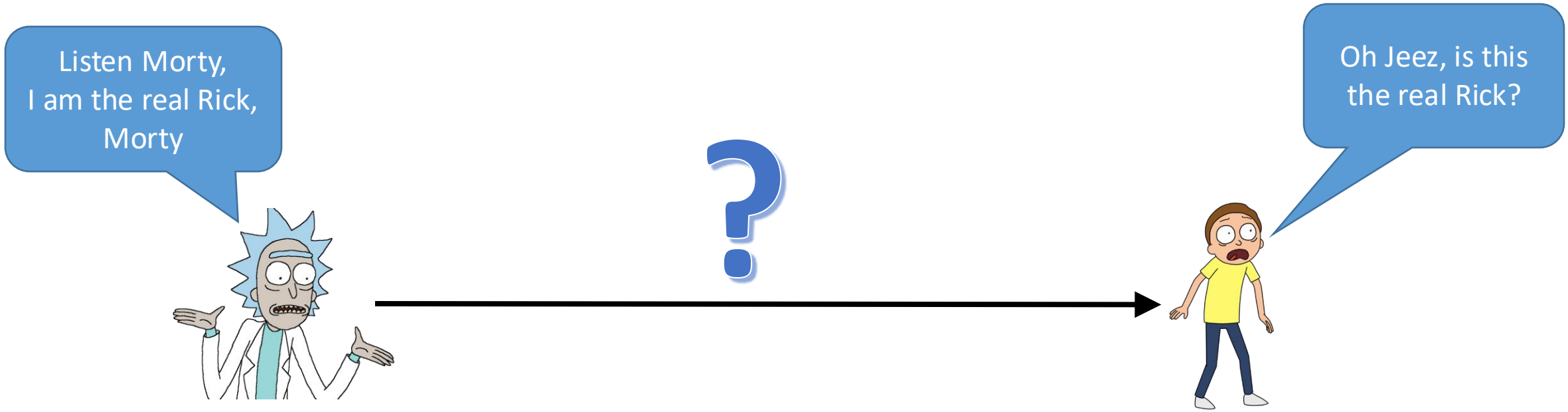
The process of verifying a claimed identity



What is authentication?

AUTHENTICATION

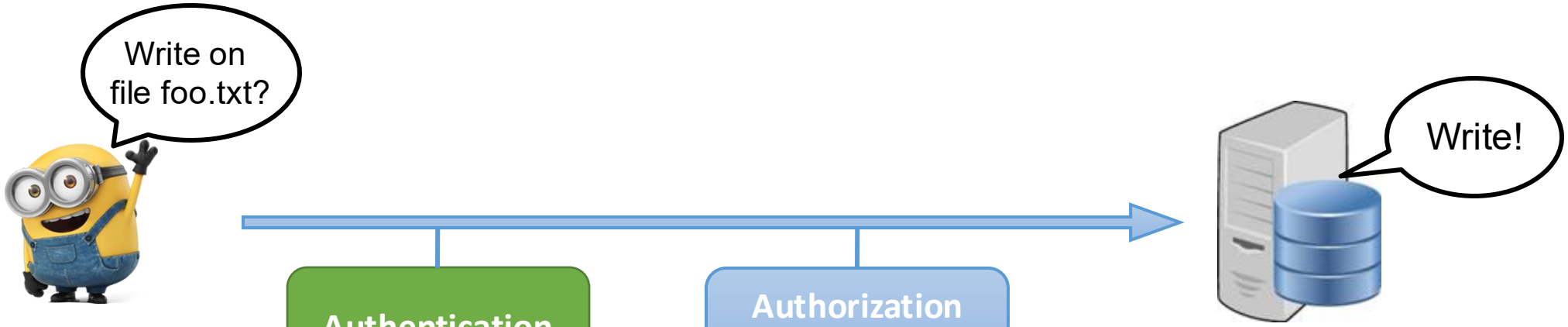
The process of verifying a claimed identity



!= MESSAGE AUTHENTICATION

The message comes from the designated sender, and has not been modified

Where does Authentication fit?



The system needs to bind the actor to a **principal** before authorization

abstract entity that is authorized to act

(***users***, *connections, processes)*

The system needs to decide whether the **principal** is authorized

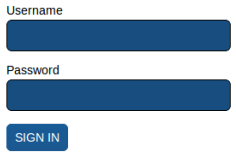
Access control

Ways to Prove Who You Are

Traditional

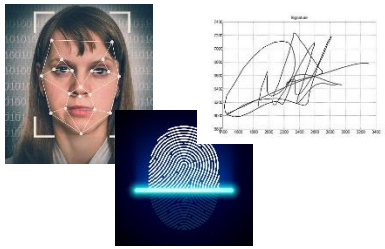
What you know

password, secret key



What you are

biometrics



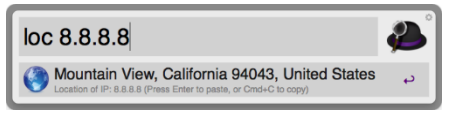
What you have

Smart card, secure tokens

Modern

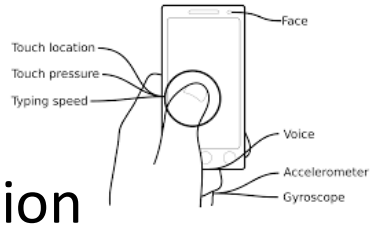
Where you are

location, IP address



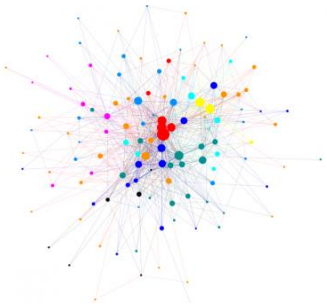
How you act

behavioural authentication



Who you know

social ties



Many others…

Ways to Prove Who You Are

TRADITIONAL

What you know

pas

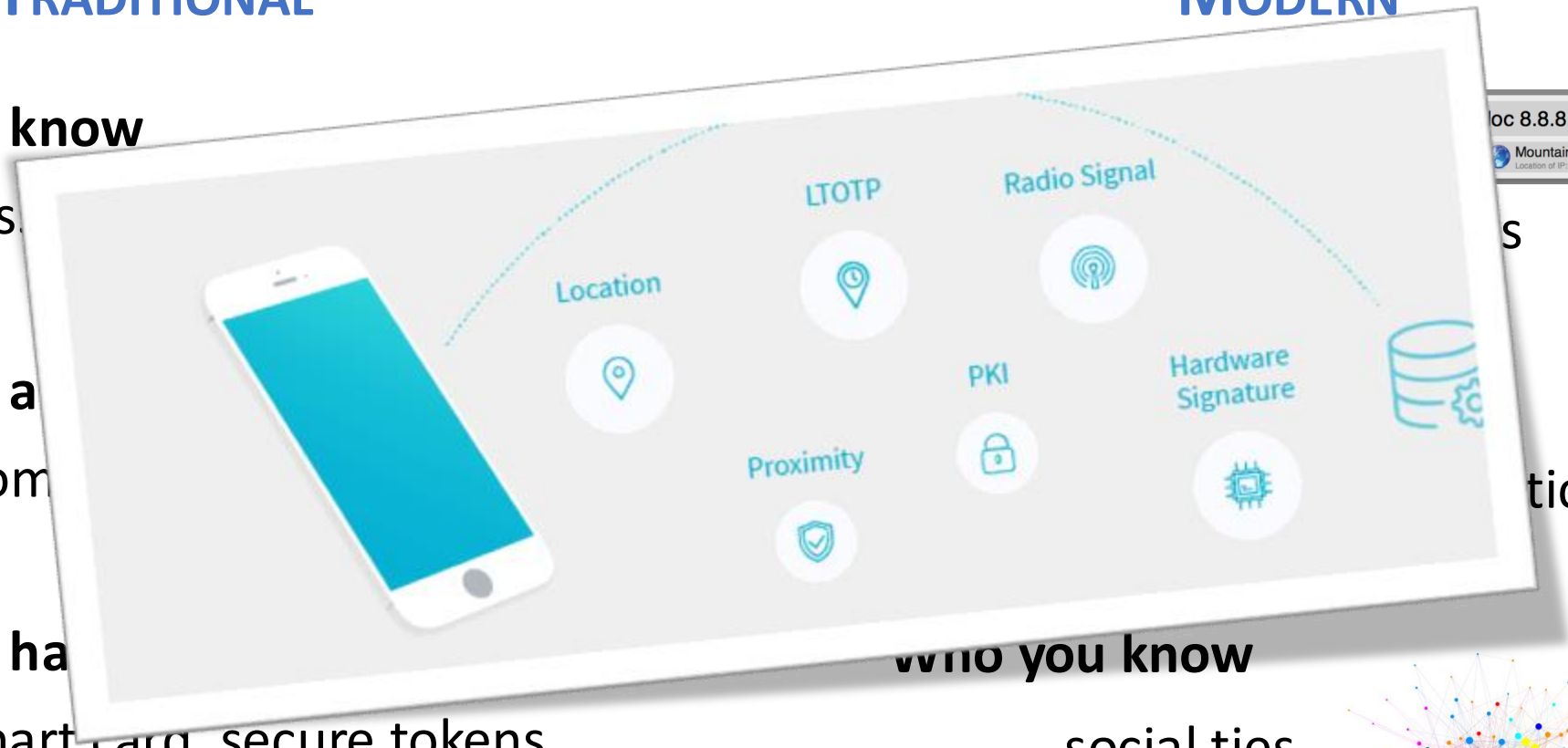
What you a

biom

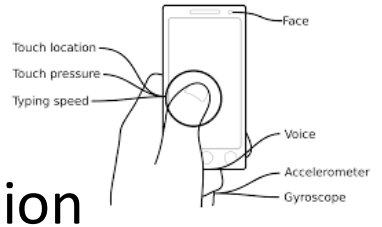
What you ha

Smart card, secure tokens

MODERN



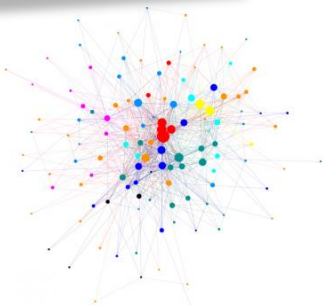
s



tication

Who you know

social ties



What you know: Passwords

PASSWORD
Secret shared between user and system

User has a secret password → System checks it to authenticate the user

What you know: Passwords

PASSWORD
Secret shared between user and system

User has a secret password → System checks it to authenticate the user

PROBLEMS TO BE SOLVED

Secure transfer: the password may be eavesdropped when communicated

What you know: Passwords

PASSWORD
Secret shared between user and system

User has a secret password → System checks it to authenticate the user

PROBLEMS TO BE SOLVED

Secure transfer: the password may be eavesdropped when communicated

Secure check: naïve checks may leak information about the password

What you know: Passwords

PASSWORD
Secret shared between user and system

User has a secret password → System checks it to authenticate the user

PROBLEMS TO BE SOLVED

Secure transfer: the password may be eavesdropped when communicated

Secure check: naïve checks may leak information about the password

Secure storage: if stolen the full system is compromised!

What you know: Passwords

PASSWORD
Secret shared between user and system

User has a secret password → System checks it to authenticate the user

PROBLEMS TO BE SOLVED

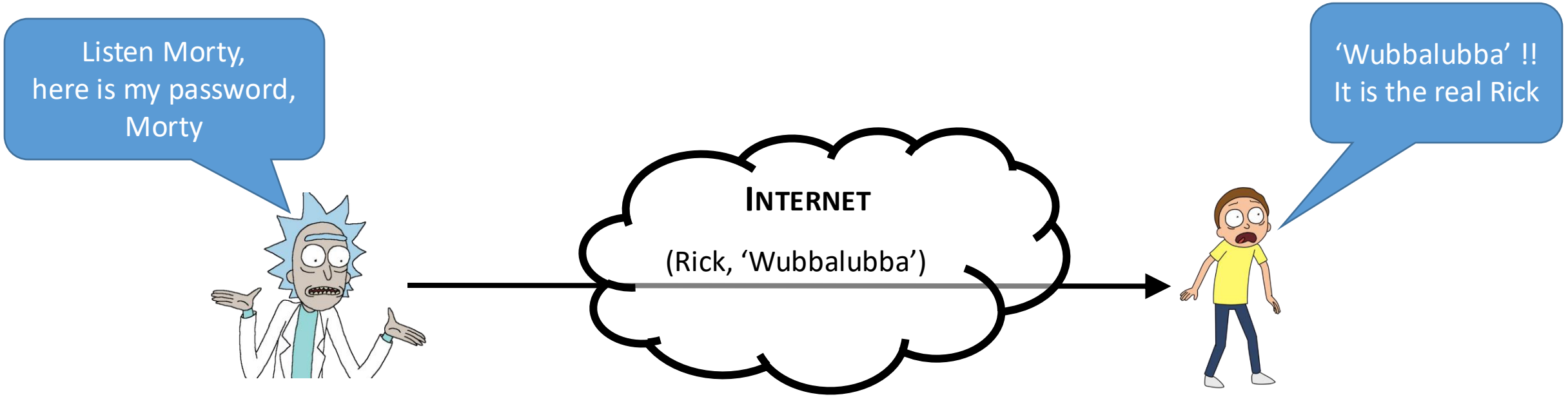
Secure transfer: the password may be eavesdropped when communicated

Secure check: naïve checks may leak information about the password

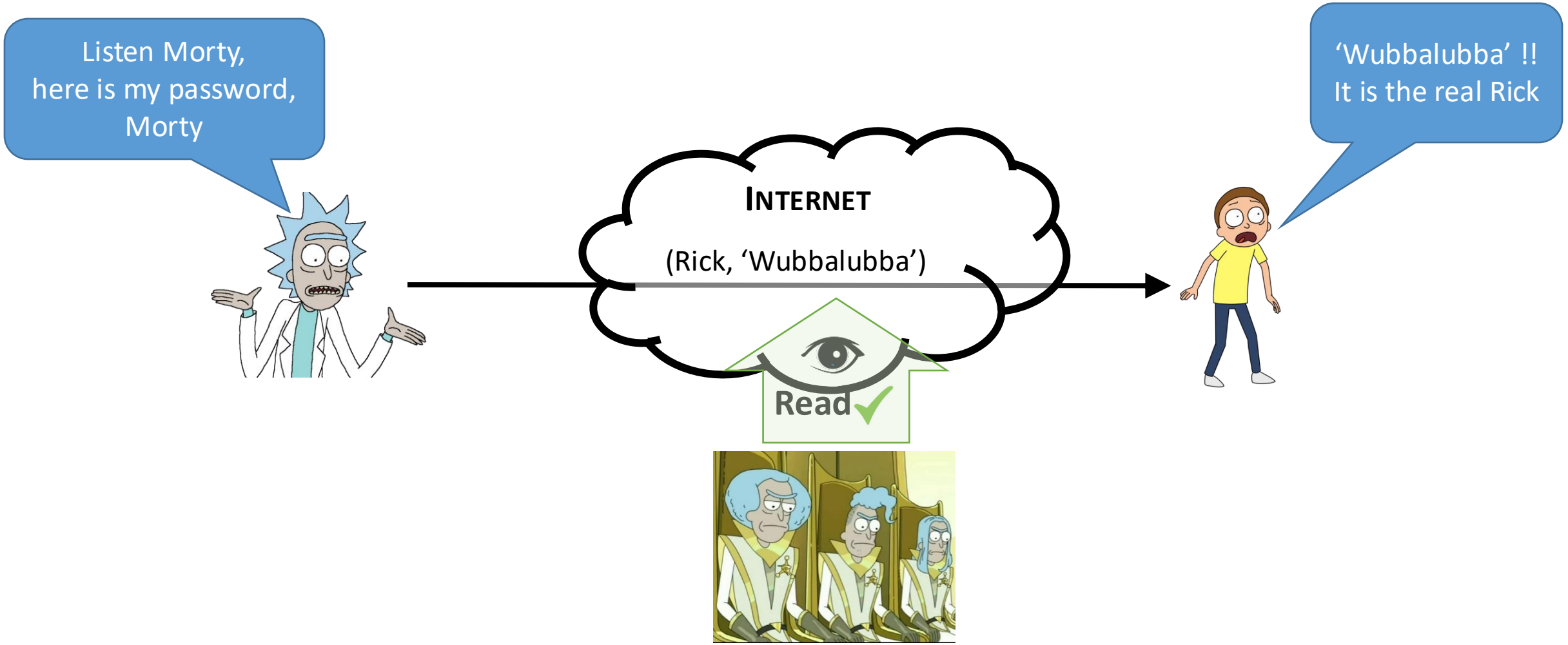
Secure storage: if stolen the full system is compromised!

Secure passwords: easy-to-remember passwords tend to be easy to guess

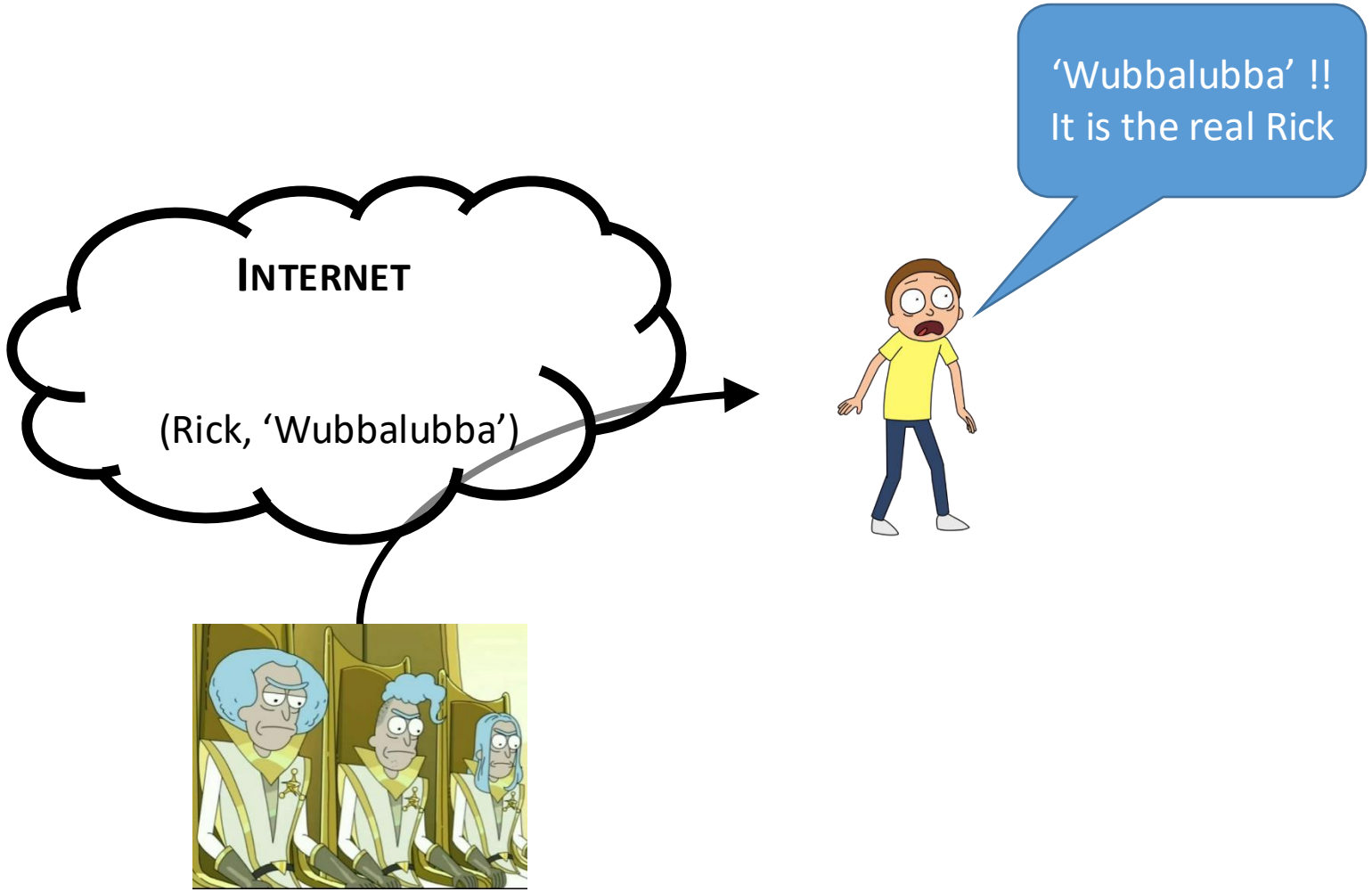
Secure transfer



Secure transfer

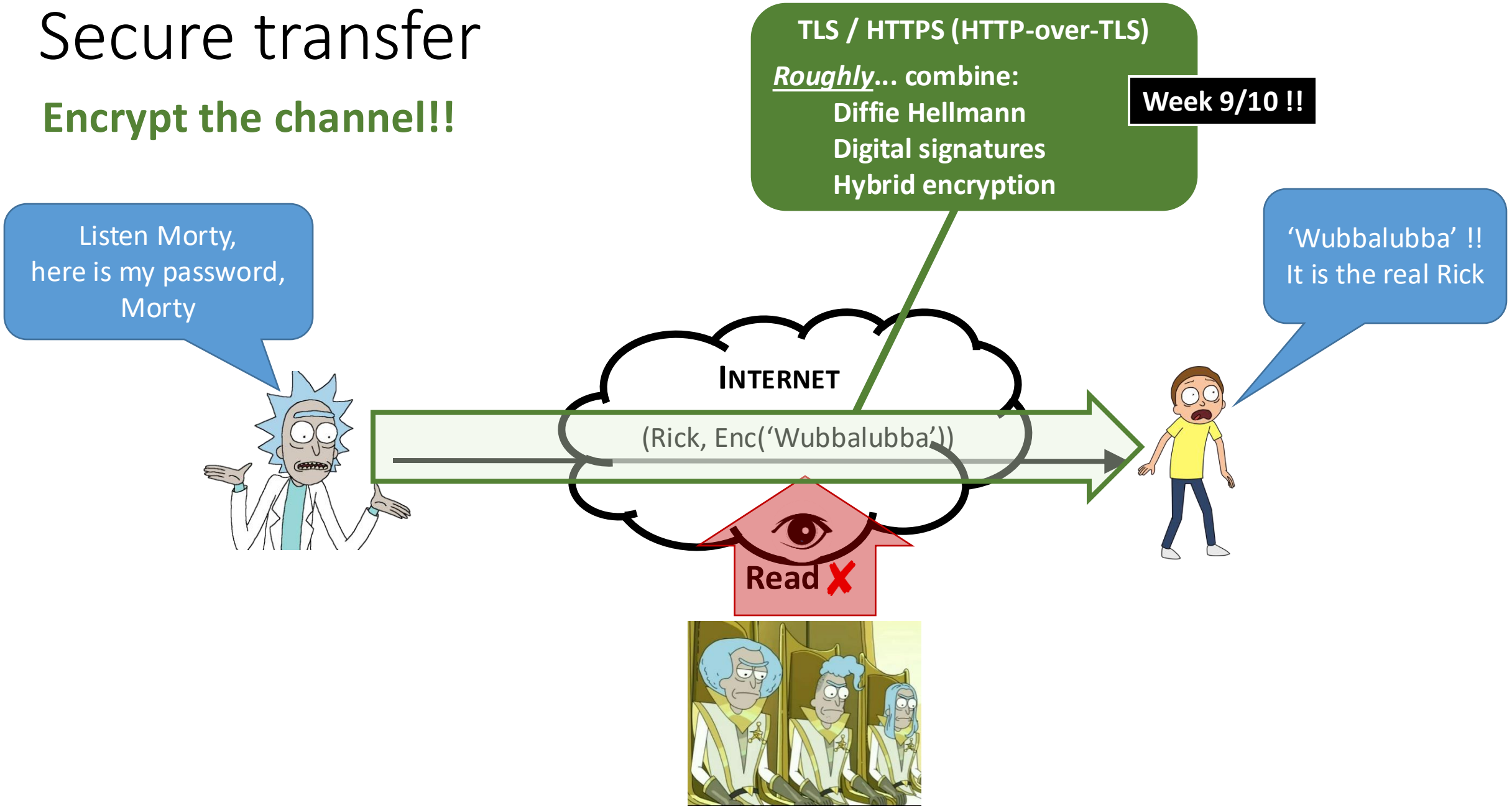


Secure transfer



Secure transfer

Encrypt the channel!!



Secure transfer – Beware of replay attacks

One of the most difficult aspects in authentication

Listen Morty,
here is my password,
Morty

INTERNET

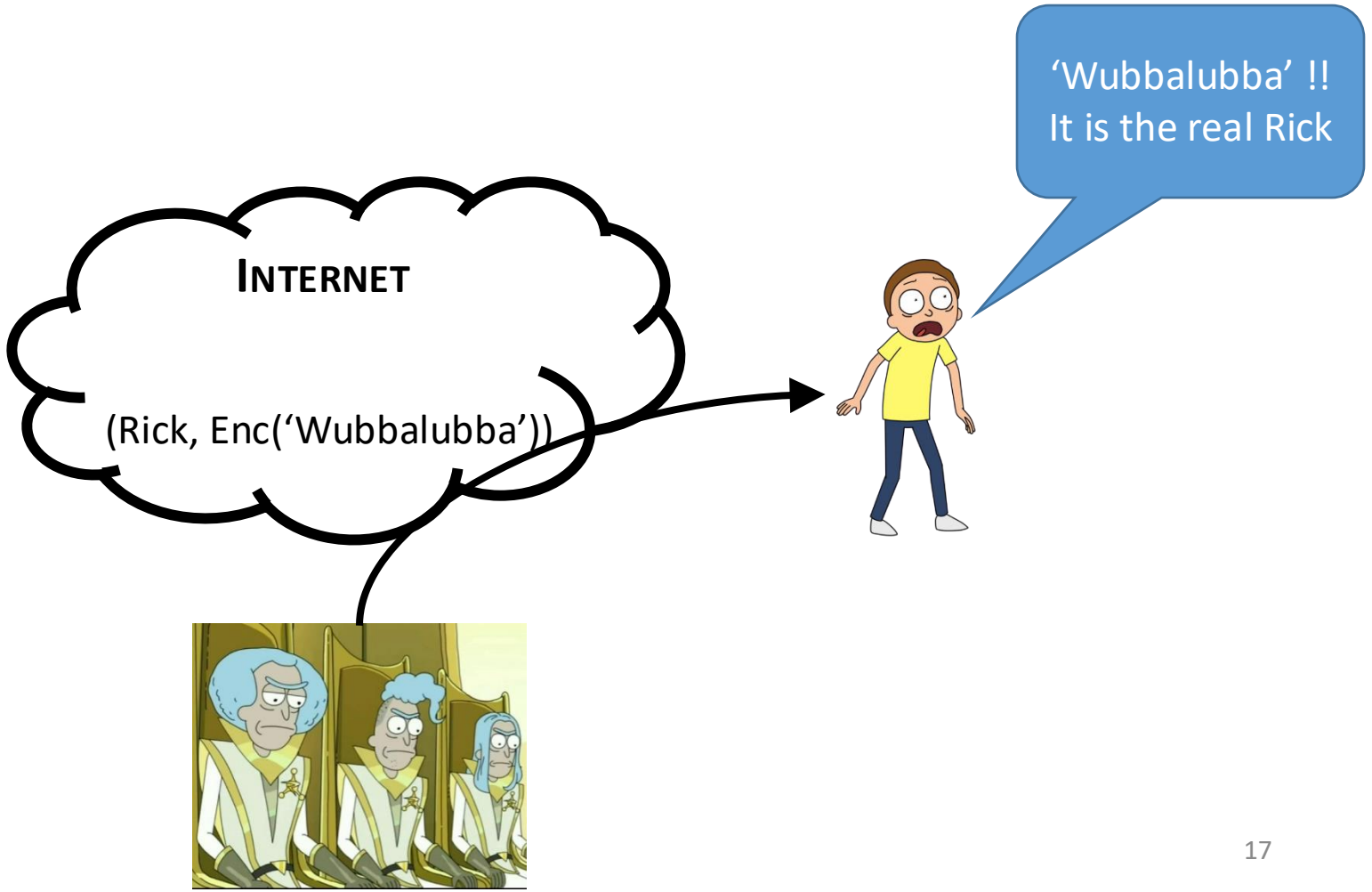
(Rick, Enc('Wubbalubba'))

Read ✗

Copy ✓

'Wubbalubba' !!
It is the real Rick

Secure transfer – Beware of replay attacks

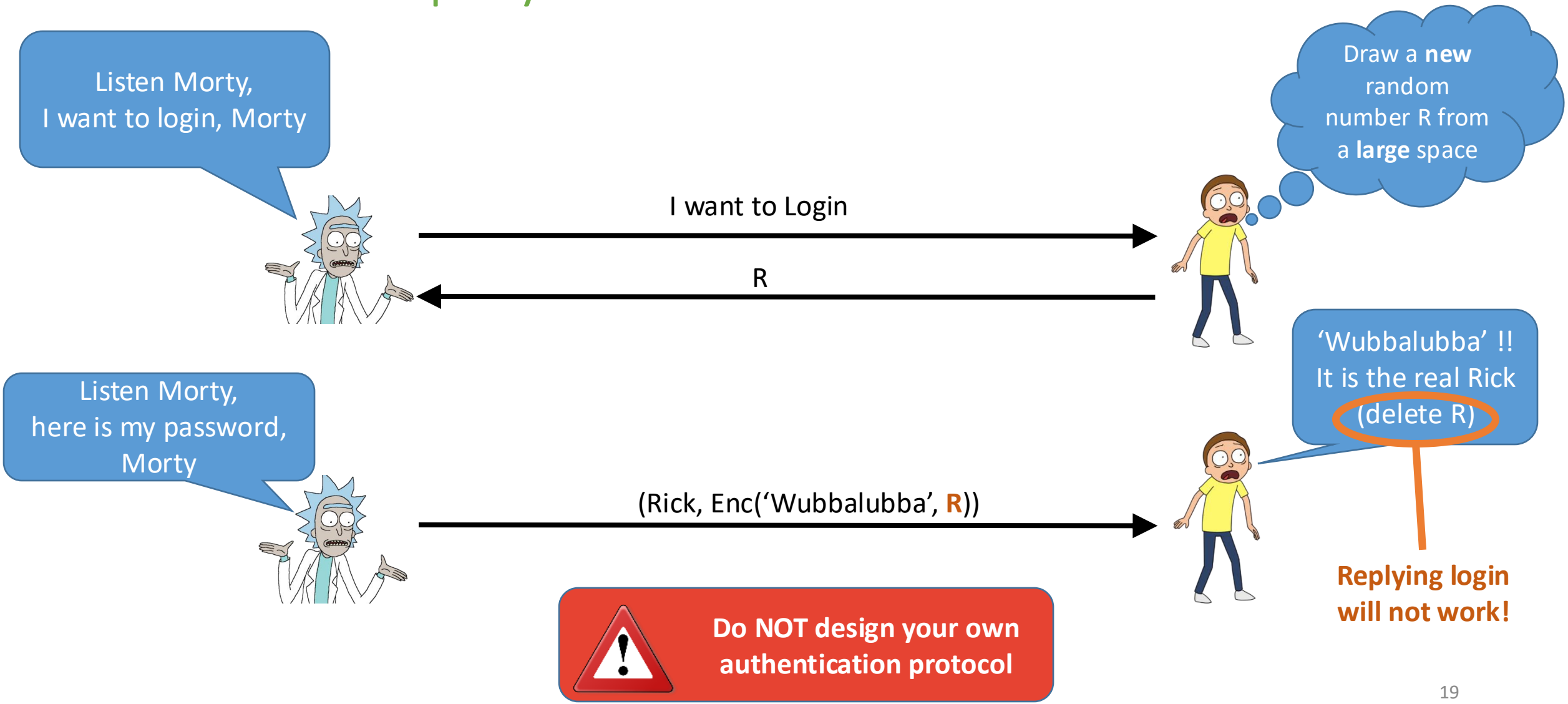


Challenge-Response protocols

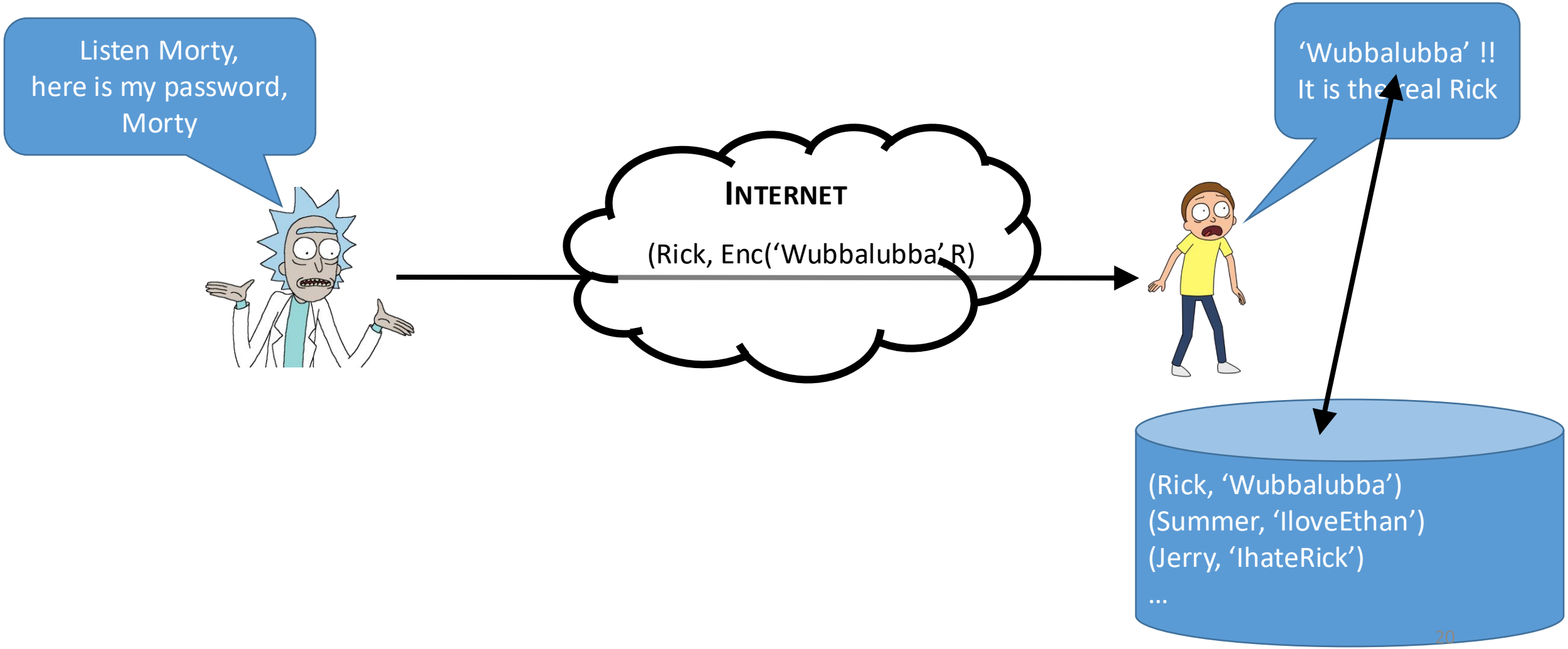
Solution to replay attacks

Challenge-Response protocols

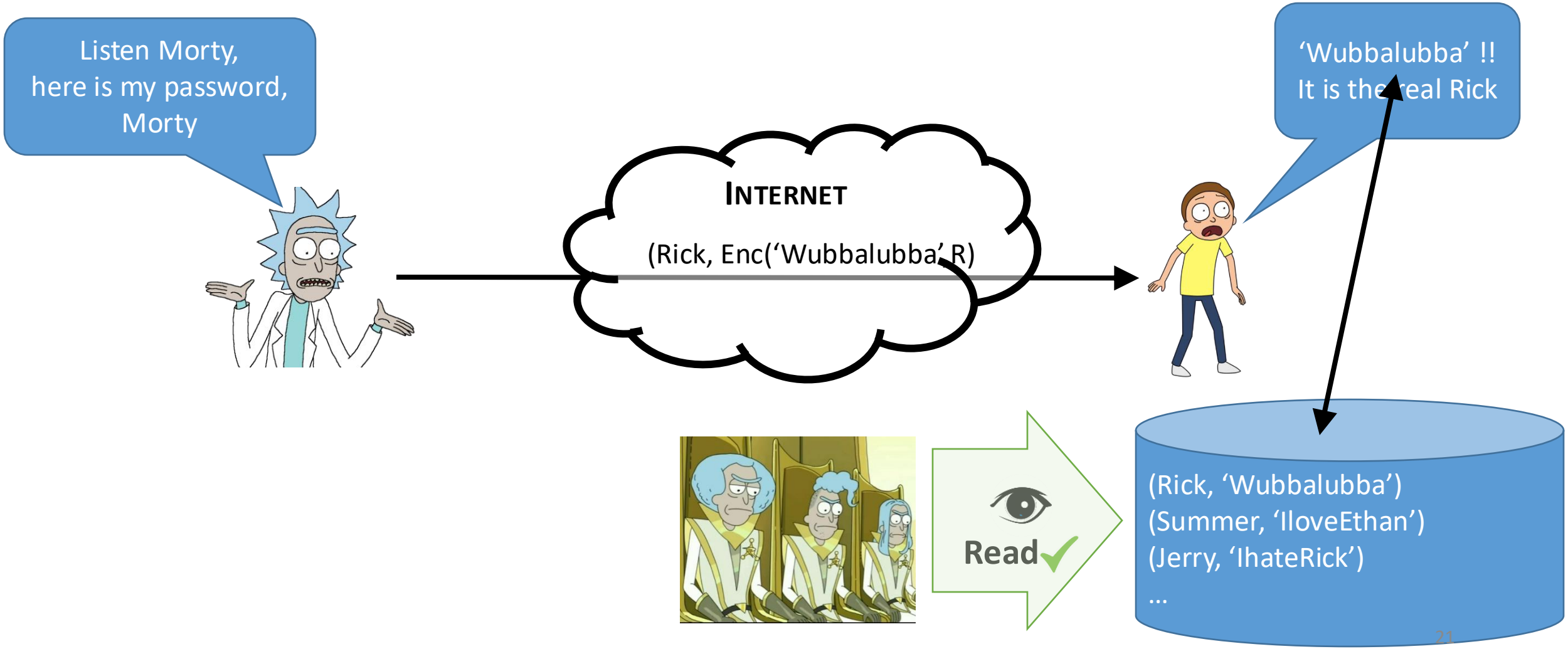
Solution to replay attacks



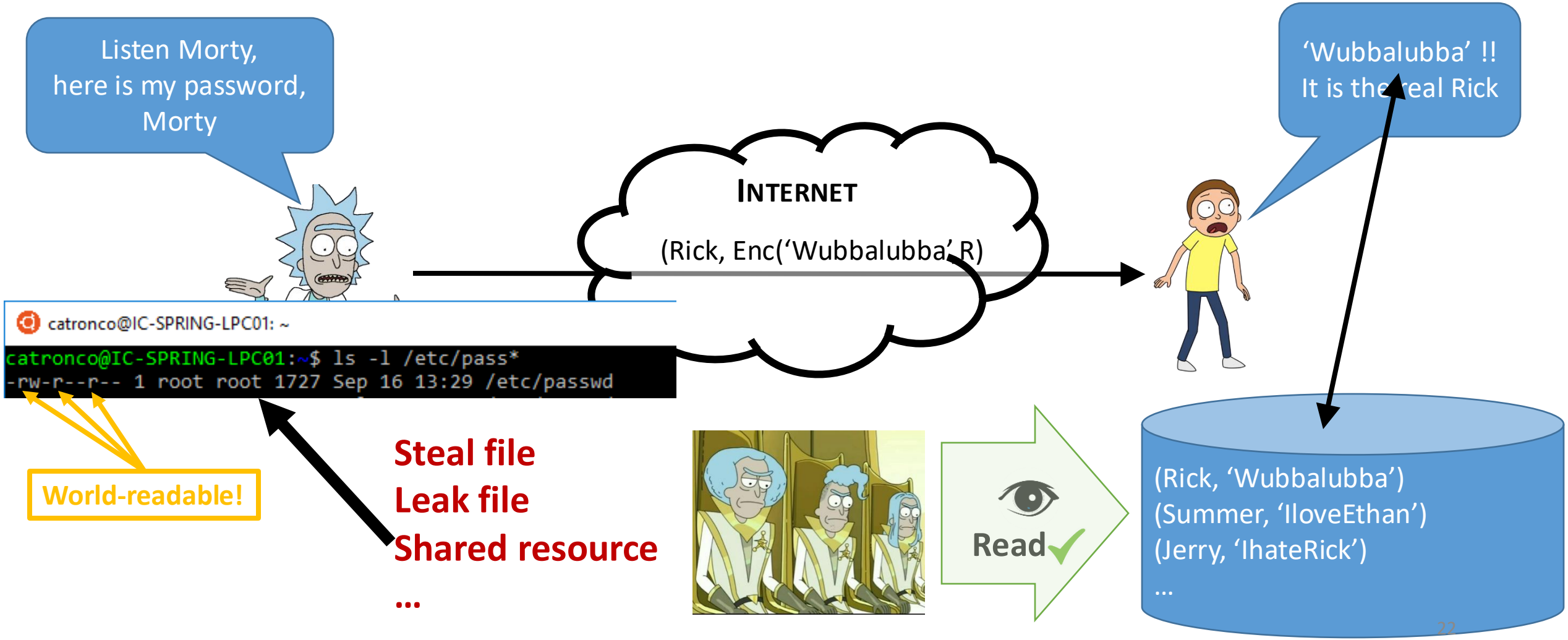
Secure storage



Secure storage



Secure storage



Password database compromises

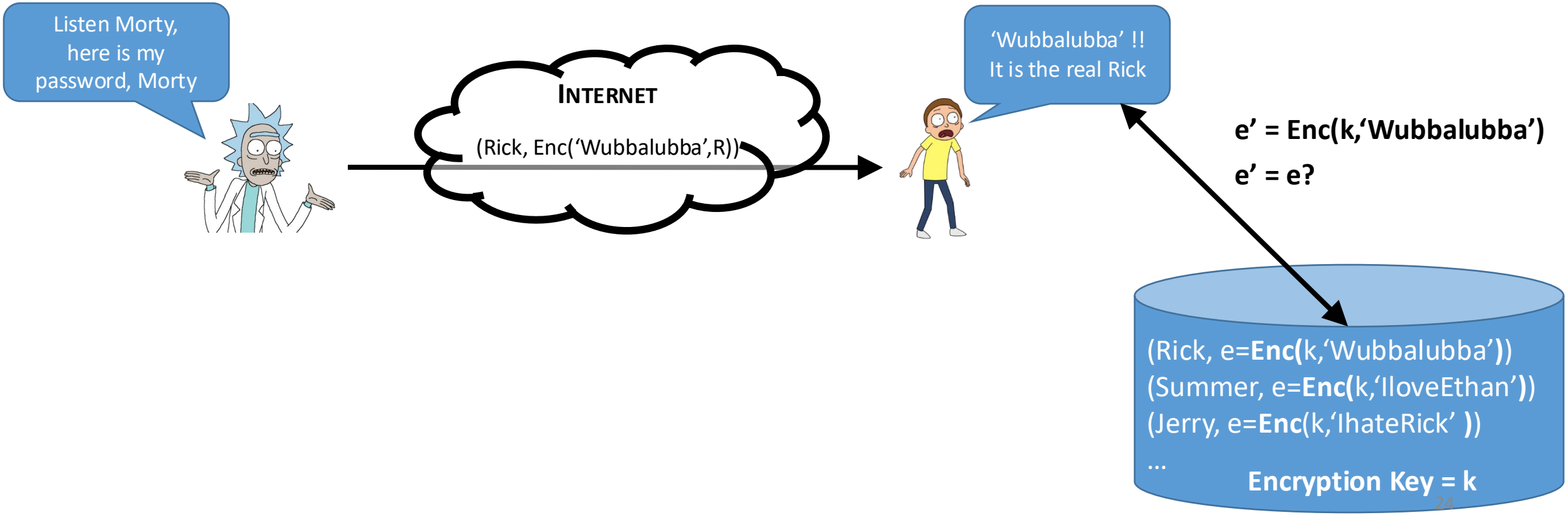
	year	# stolen
rockyou	2012	32.6 million
LinkedIn	2012	117 million
Adobe	2013	36 million
YAHOO!	2014	~500 million
ASHLEY MADISON® Life is short. Have an affair.®	2015	36 million

Source: Tom Ristenpart

Secure storage - Do not store in the clear!

OPTION 1

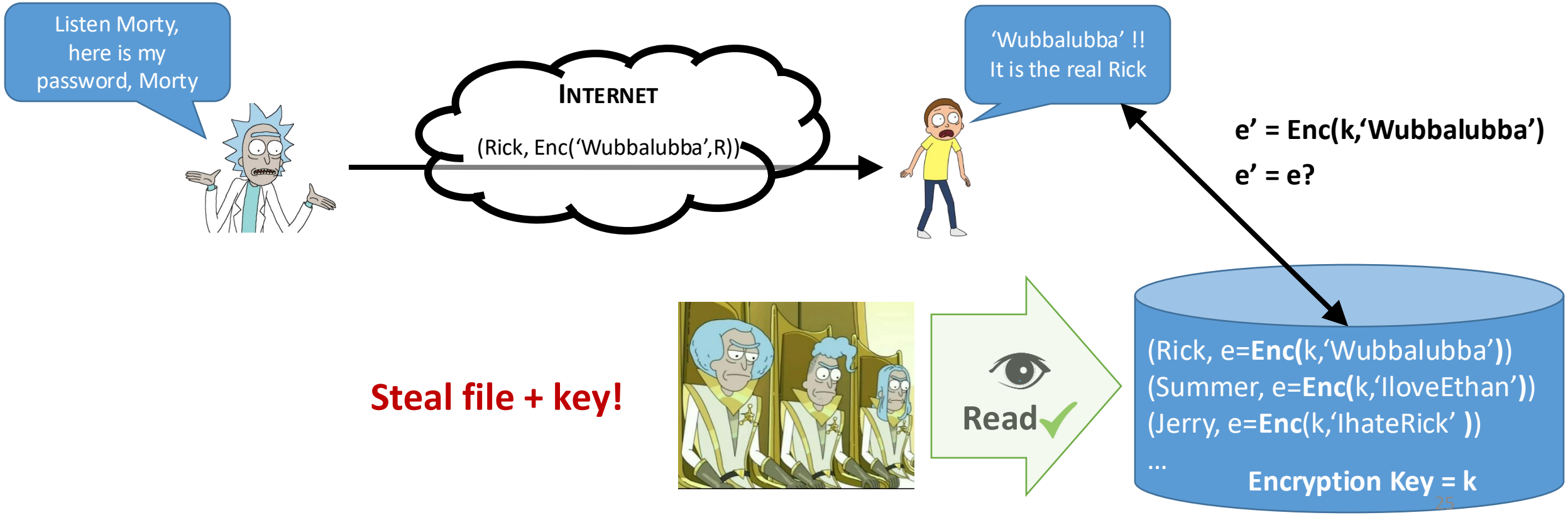
Store password encrypted



Secure storage - Do not store in the clear!

OPTION 1

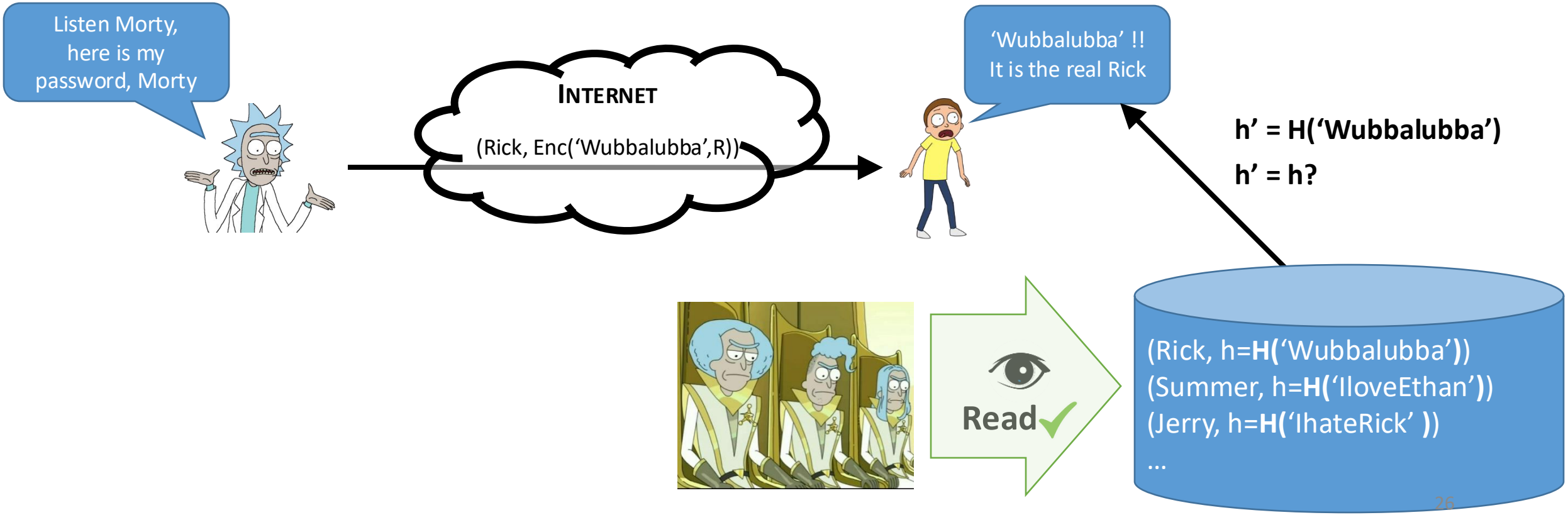
Store password encrypted



Secure storage - Do not store in the clear!

OPTION 2

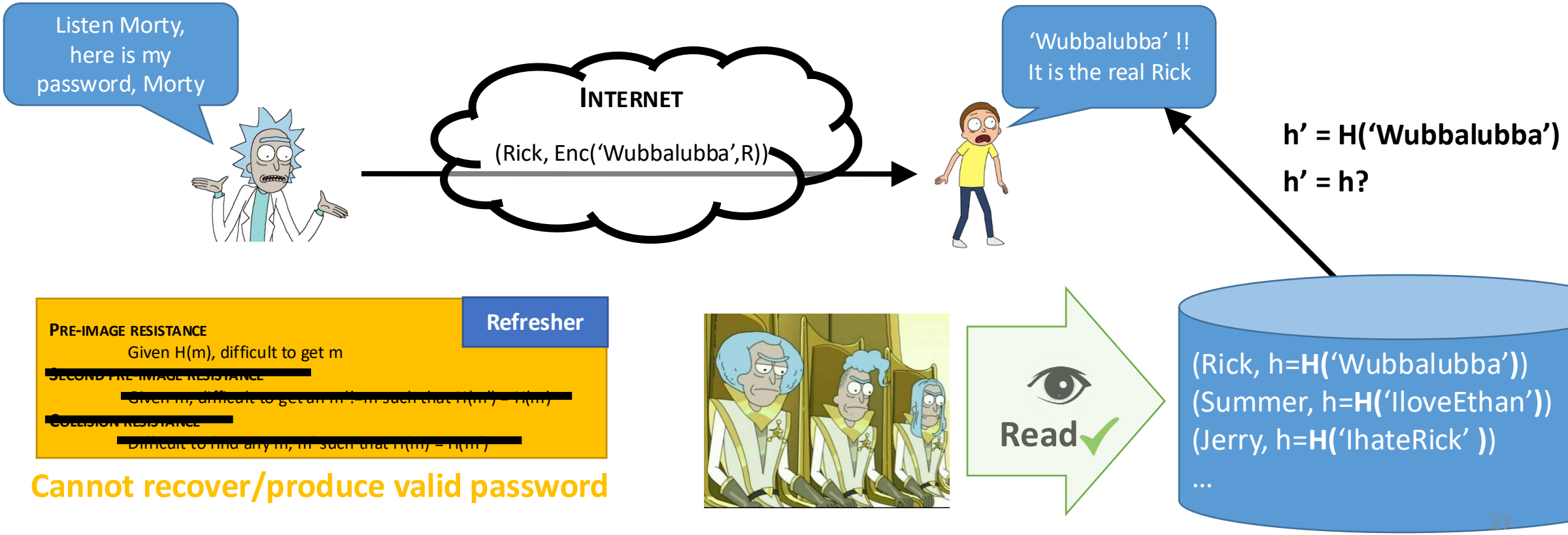
Store password as a “hash” of its value



Secure storage - Do not store in the clear!

OPTION 2

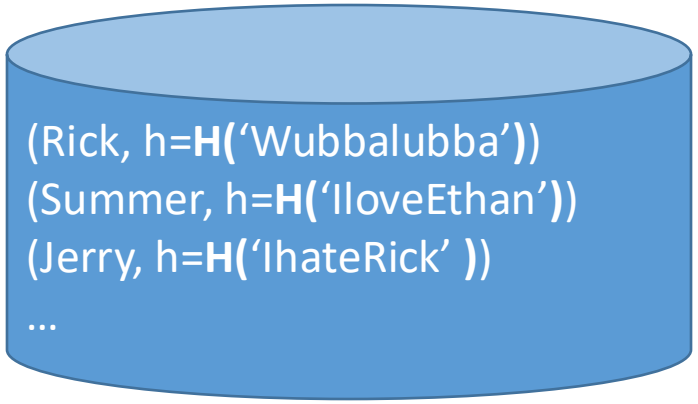
Store password as a “hash” of its value



Secure storage - Do not store in the clear!

OPTION 2

Store password as a “hash” of its value



OFFLINE ATTACKS – DICTIONARY ATTACK

Anyone can compute a hash

Passwords **not truly random**

- 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols,
- 94^8 eight-character passwords (around 2^{52}) possibilities

Users use a **limited set** of passwords (reduced search space)

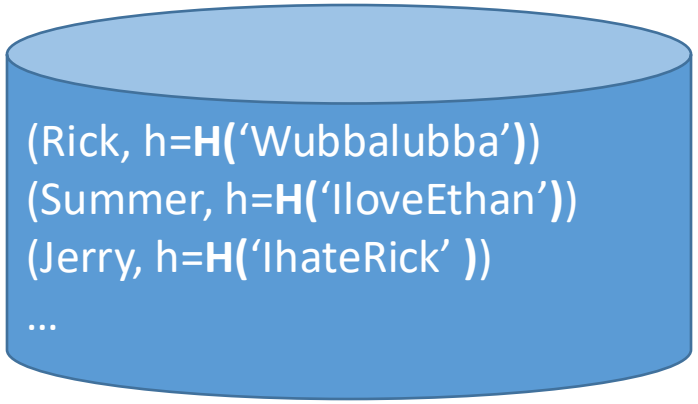
https://en.wikipedia.org/wiki/List_of_the_most_common_passwords

https://resources.infosecinstitute.com/10-popular-password-cracking-tools/#gref

Secure storage - Do not store in the clear!

OPTION 2

Store password as a “hash” of its value



OFFLINE ATTACKS – DICTIONARY ATTACK

Anyone can compute a hash

Passwords **not truly random**

- 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols,
- 94^8 eight-character passwords (around 2^{52}) possibilities

Users use a **limited set** of passwords (reduced search space)

Attacker can compute H(word) for every word in the dictionary and see if the result is in the password file!

Can **reuse** the dictionary

Parallel cracking with GPU accelerates search

Other tricks: rainbow tables, pre-computation,…

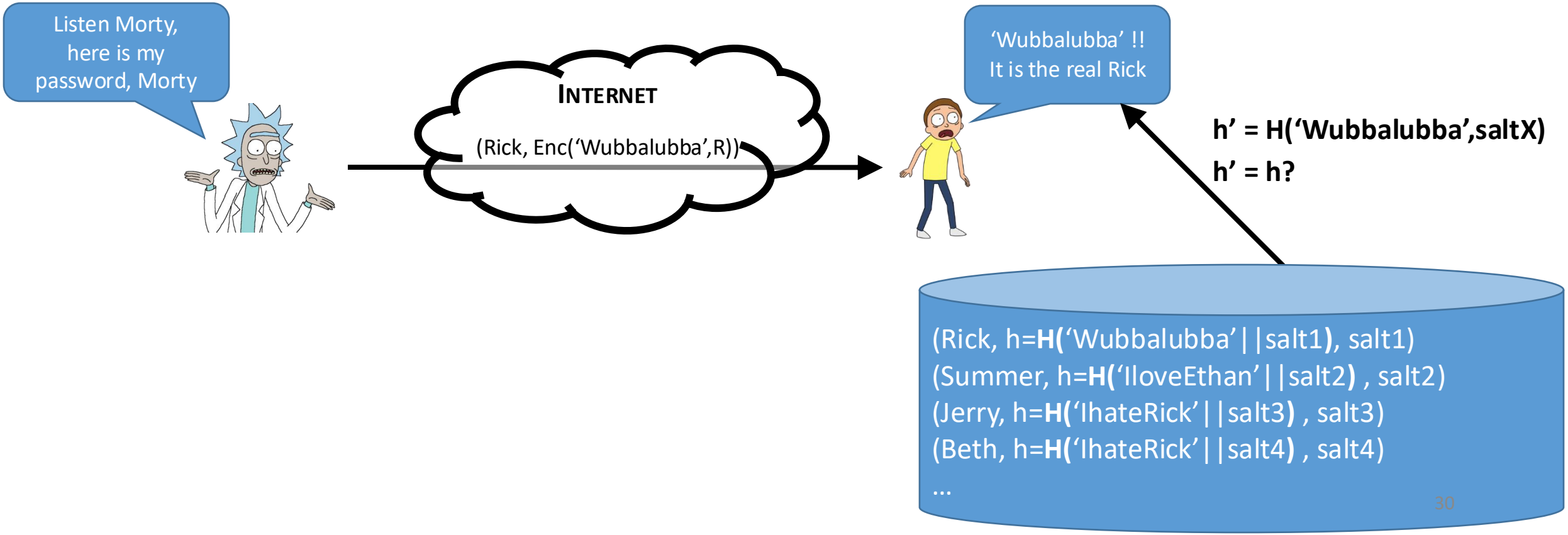
https://en.wikipedia.org/wiki/List_of_the_most_common_passwords

https://resources.infosecinstitute.com/10-popular-password-cracking-tools/#gref

Secure storage – Do this! (with a library, slide 27)

OPTION 3

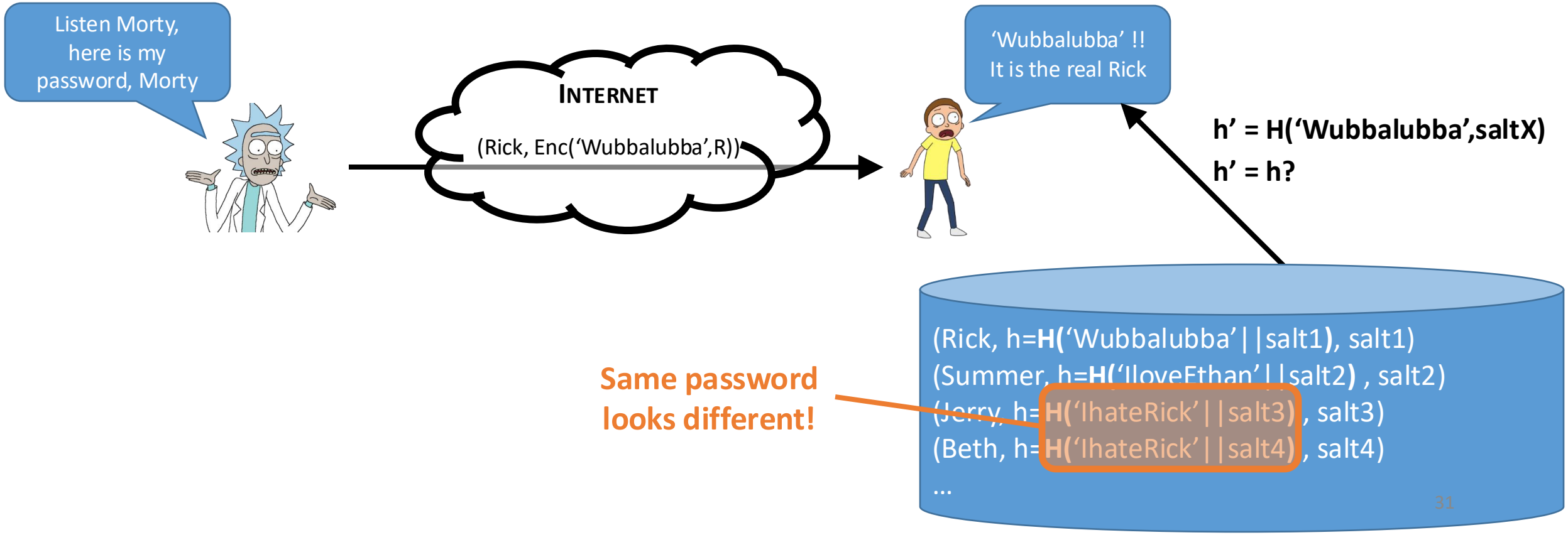
Store password as a “hash”+ “salt”



Secure storage – Do this! (with a library, slide 27)

OPTION 3

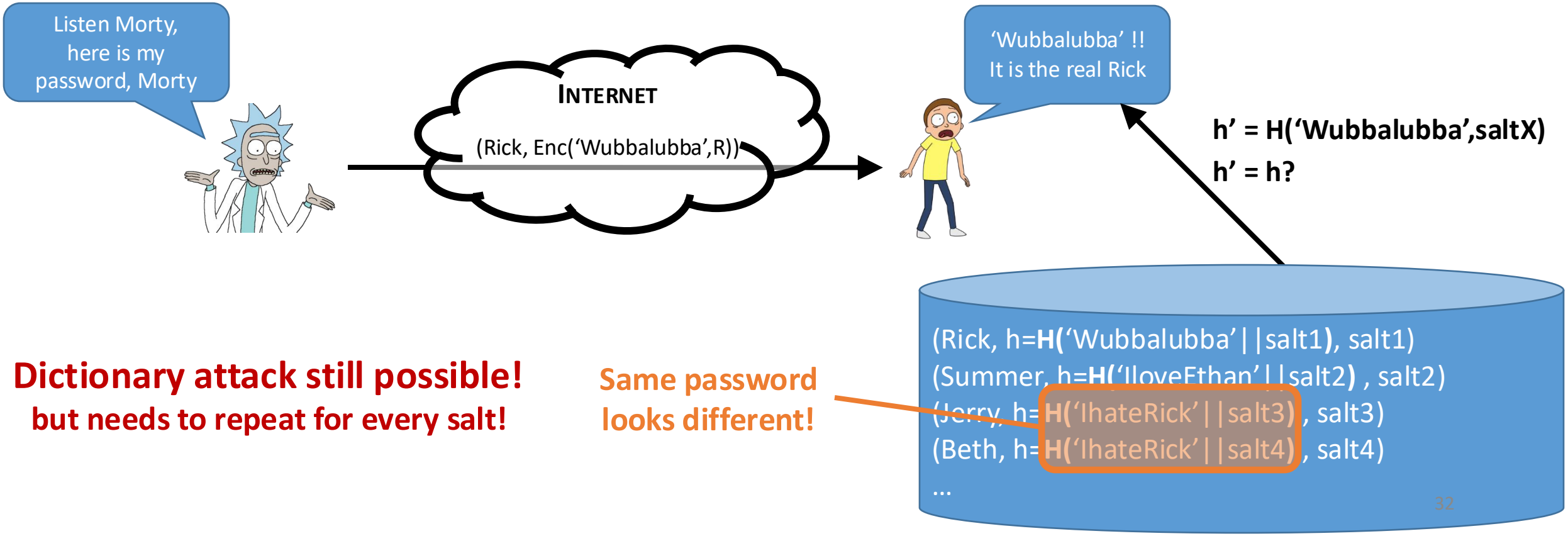
Store password as a “hash”+ “salt”



Secure storage – Do this! (with a library, slide 27)

OPTION 3

Store password as a “hash”+ “salt”



Dictionary attack still possible!

but needs to repeat for every salt!

Secure storage – Do this!

OPTION 3

Store password as a “hash”+ “salt”

COMPLEMENTARY DEFENSES

Use of hash functions designed to be **slow** (bcrypt, scrypt, argon2)

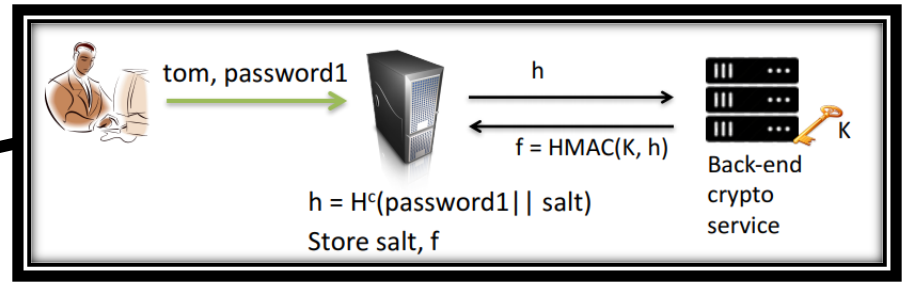
Repeat several times (e.g., 1000)

Require specific elements in passwords

Increase entropy

Split check, require a second server

Invalidate offline attacks



Access control! (`/etc/shadow` in UNIX only accessible by root)

Facebook password onion

```
$cur  = 'password'
$cur  = md5($cur)
$salt = randbytes(20)
$cur  = hmac_sha1($cur, $salt)
$cur  = remote_hmac_sha256($cur, $secret)
$cur  = scrypt($cur, $salt)
$cur  = hmac_sha256($cur, $salt)
```

Why this onion?

Source: Tom Ristenpart

Password database compromises

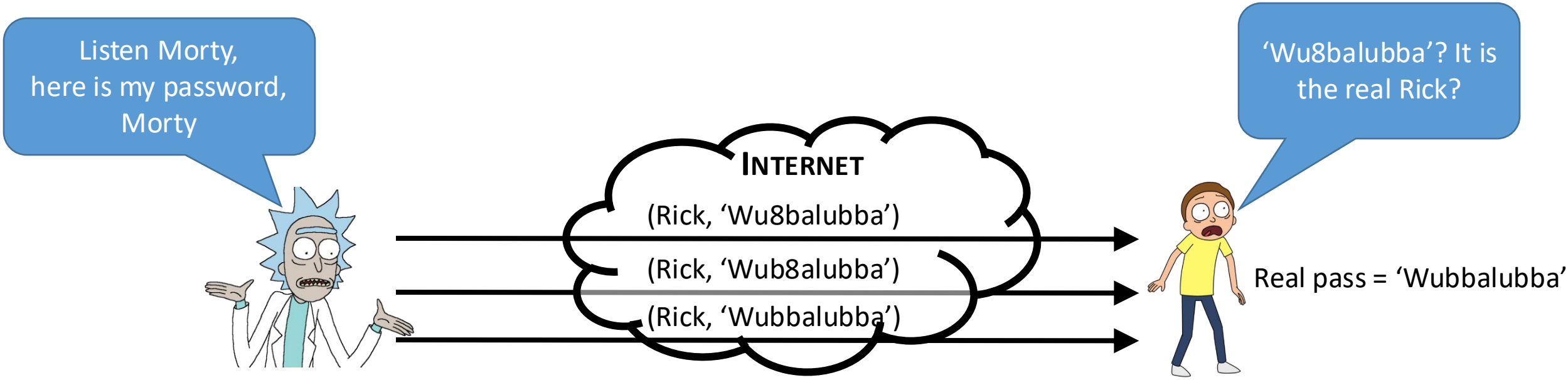
	year	# stolen	% recovered	format
rockyou	2012	32.6 million	100%	plaintext (!)
LinkedIn	2012	117 million	90%	Unsalted SHA-1
Adobe	2013	36 million	??	ECB encryption
YAHOO!	2014	~500 million	??	bcrypt + ??
ASHLEY MADISON® Life is short. Have an affair.®	2015	36 million	33%	Salted bcrypt + MD5

Source: Tom Ristenpart

Secure checking

OPTION 1

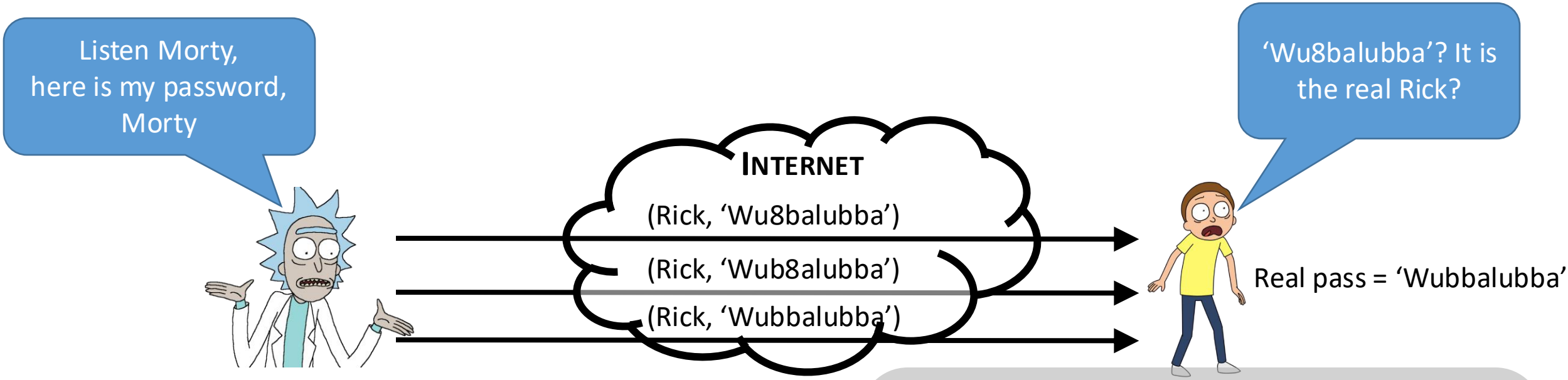
Check letter by letter



Secure checking

OPTION 1

Check letter by letter



Timing channel leaks information about the password

W=W?
u=u?
b=b?
b=b?
a=a?
…
b=b?
a=a? ✓ Accept!

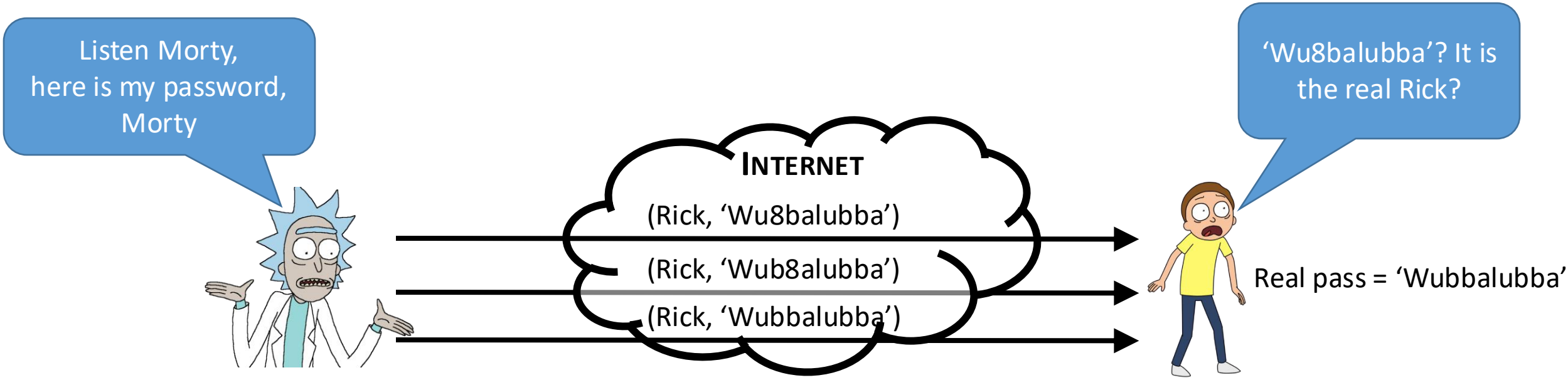
W=W?
u=u?
b=b?
8=b? ✗ Reject!

W=W?
u=u?
b=b? ✗ Reject!

Secure checking

OPTION 2

Always check everything



W=W?
u=u?
b=b?
b=b?
a=a?
…
b=b?
a=a? ✓ Accept!

W=W?
u=u?
b=b?
8=b?
a=a?
…
b=b?
a=a? ✗ Reject!

W=W?
u=u?
8=b?
b=b?
a=a?
…
b=b?
a=a? ✗ Reject!

Hashing will take care of equal operations!
Require same operations for accept and reject!!

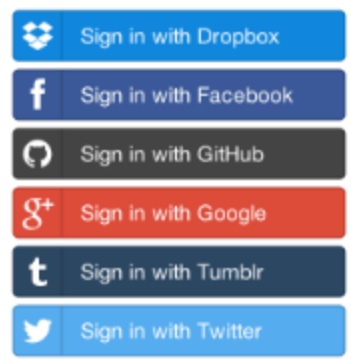
Authentication library

Dedicated security frameworks.

Embedded authentication libraries in web frameworks.

Cross-platform authentication libraries.

OAuth: performs the authentication in a third-party.



Problems with passwords

Strong passwords are difficult to remember

- Written passwords
- Reuse across systems

Problems with passwords

Strong passwords are difficult to remember

- Written passwords
- Reuse across systems

Can be stolen

- Keylogger
- Shoulder surfing
- Phishing
- Social engineering

Problems with passwords

Strong passwords are difficult to remember

- Written passwords
- Reuse across systems

Can be stolen

- Keylogger
- Shoulder surfing
- Phishing
- Social engineering

Jul 6, 2017, 10:10am

Help! Hackers Stole My Password Just By Listening To Me Type On Skype!

Thomas Brewster Forbes Staff
Security
I cover crime, privacy and security in digital and physical forms.

For many, everyday life involves sitting in front of a computer typing endless emails, presentation documents and reports. Then there's the frequent typing of passwords just to get access to those files. But beware: researchers have hacked together a tool that can harvest what's being typed simply by listening to the sounds of the keys.

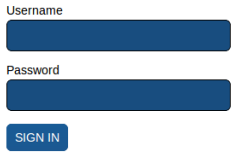
They've created the Skype&Type program for snooping on Skype

Ways to Prove Who You Are

TRADITIONAL

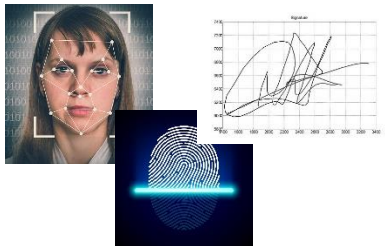
What you know

password, secret key



What you are

biometrics



What you have

Smart card, secure tokens

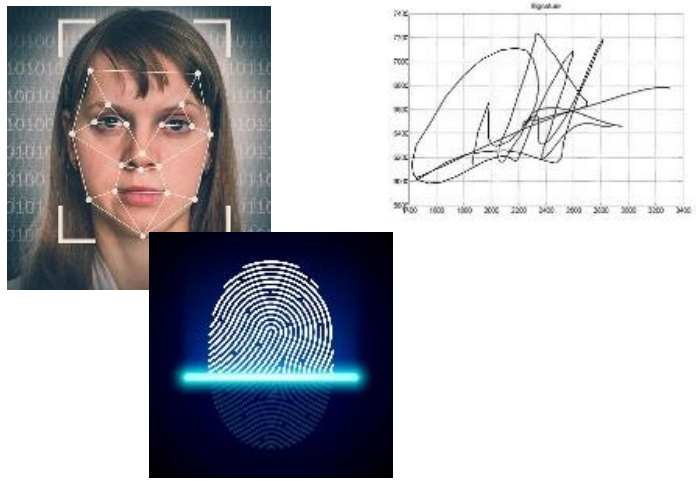
What you are: Biometrics

BIOMETRICS

is the measurement and statistical analysis of people's unique physical characteristics *(modern: also behavioral)*

Popular biometrics

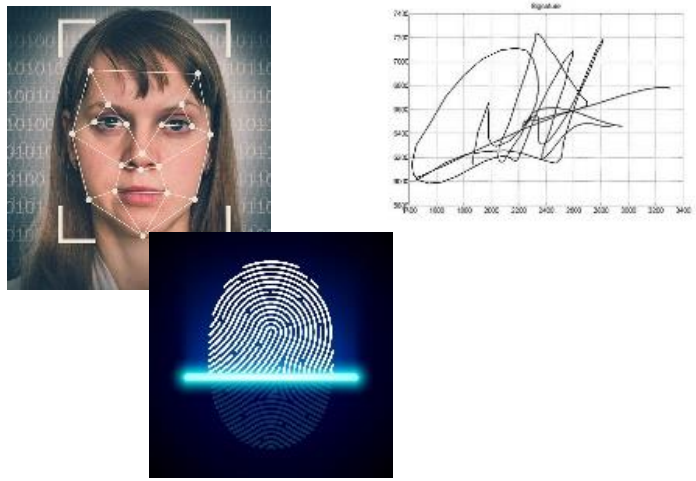
Fingerprint, face recognition, retina, voice, handwritten signature, DNA



What you are: Biometrics

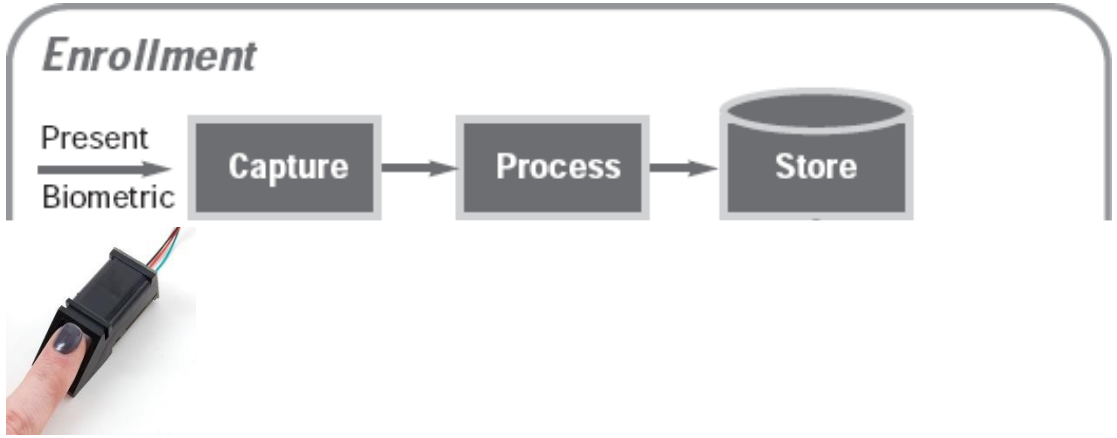
BIOMETRICS
is the measurement and statistical analysis of people's unique physical characteristics *(modern: also behavioral)*

Popular biometrics
Fingerprint, face recognition, retina, voice, handwritten signature, DNA

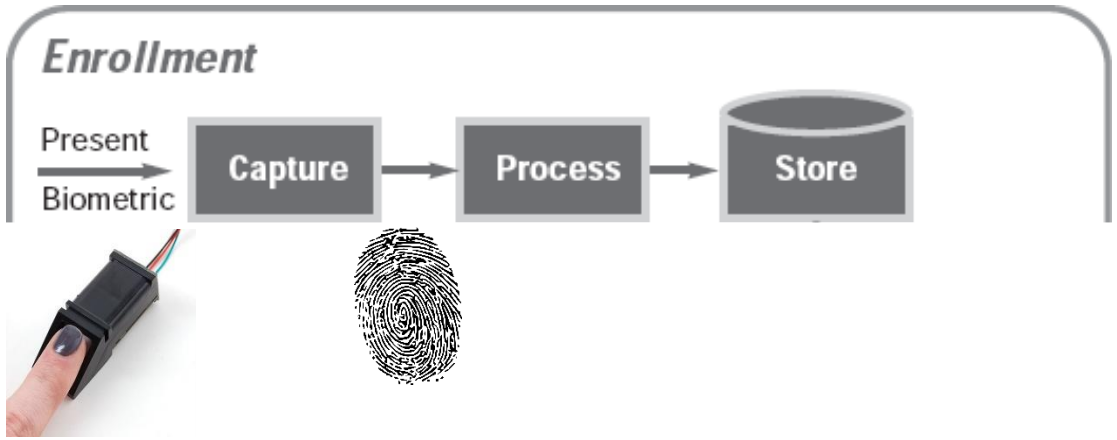


Advantages
Nothing to remember
Passive
Difficult to delegate
If the algorithm is very accurate, they are unique

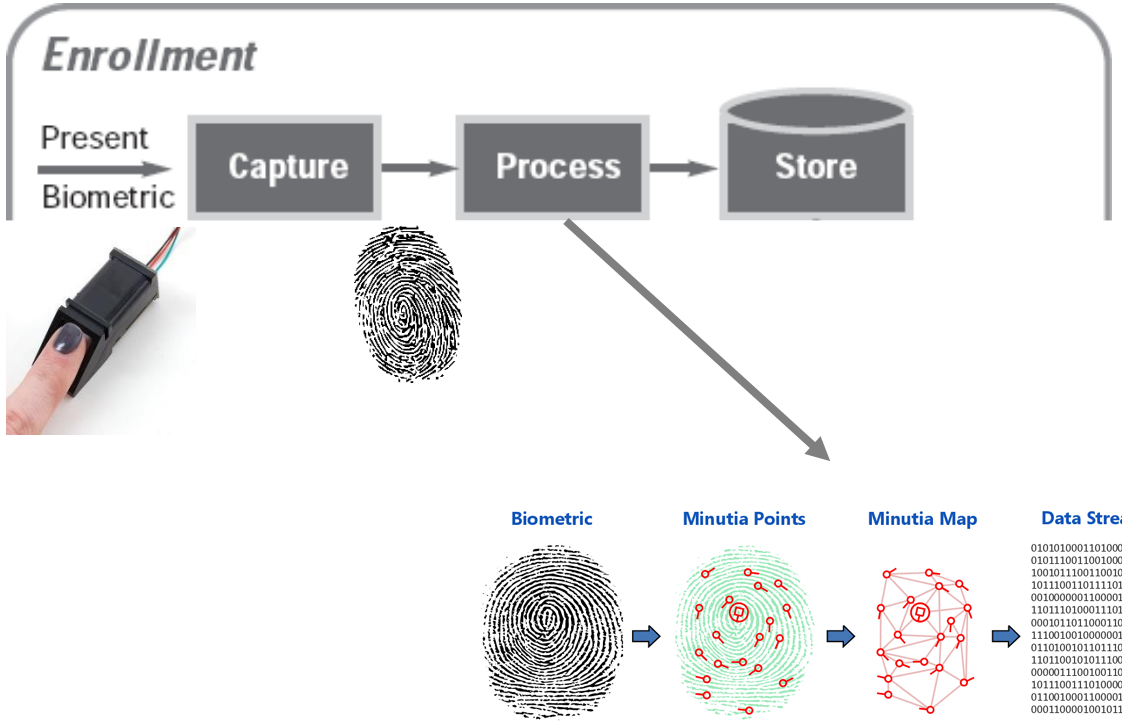
Biometrics authentication: 1) Enrollment



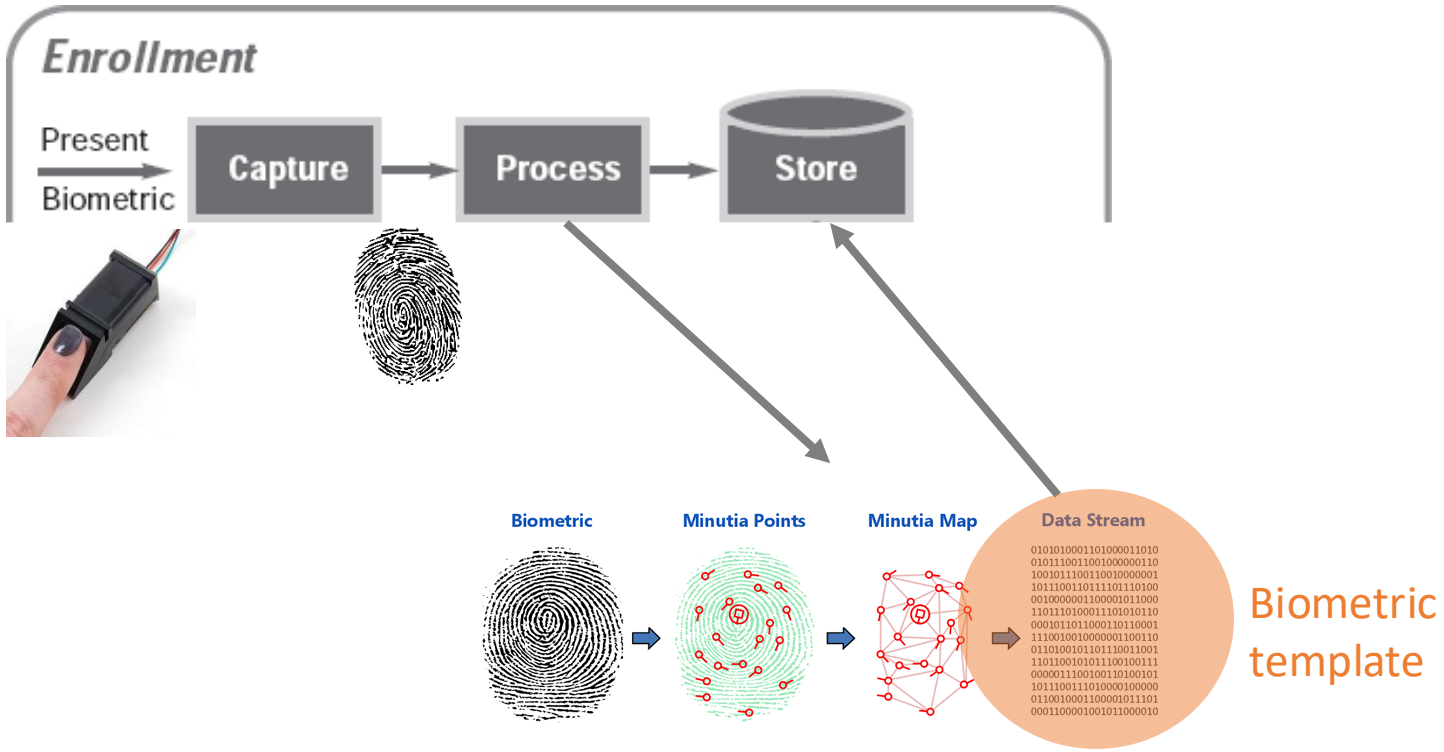
Biometrics authentication: 1) Enrollment



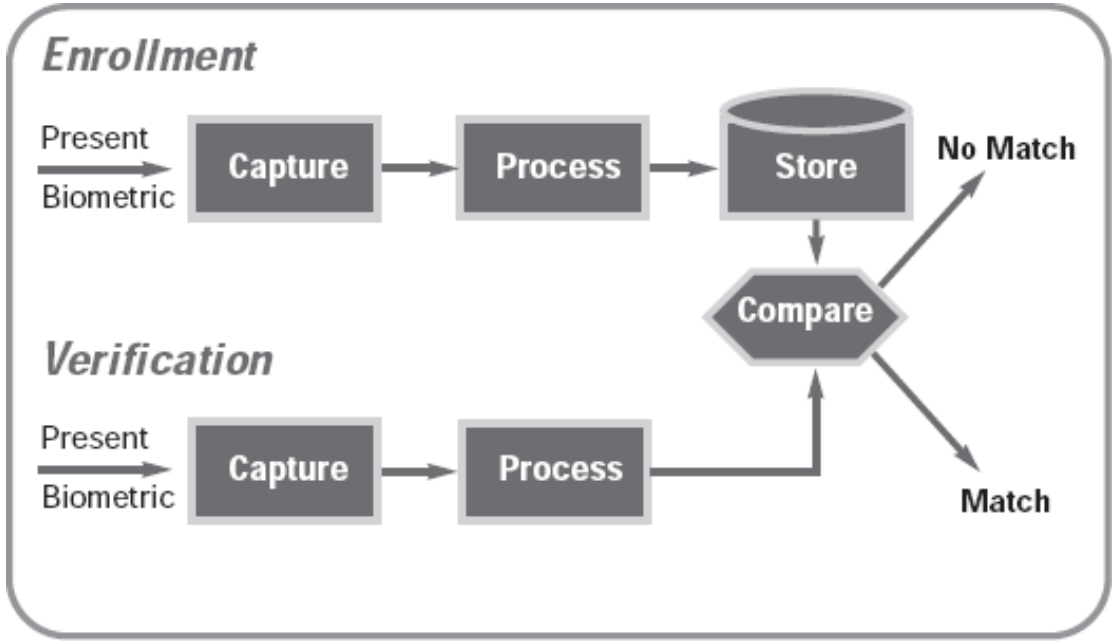
Biometrics authentication: 1) Enrollment



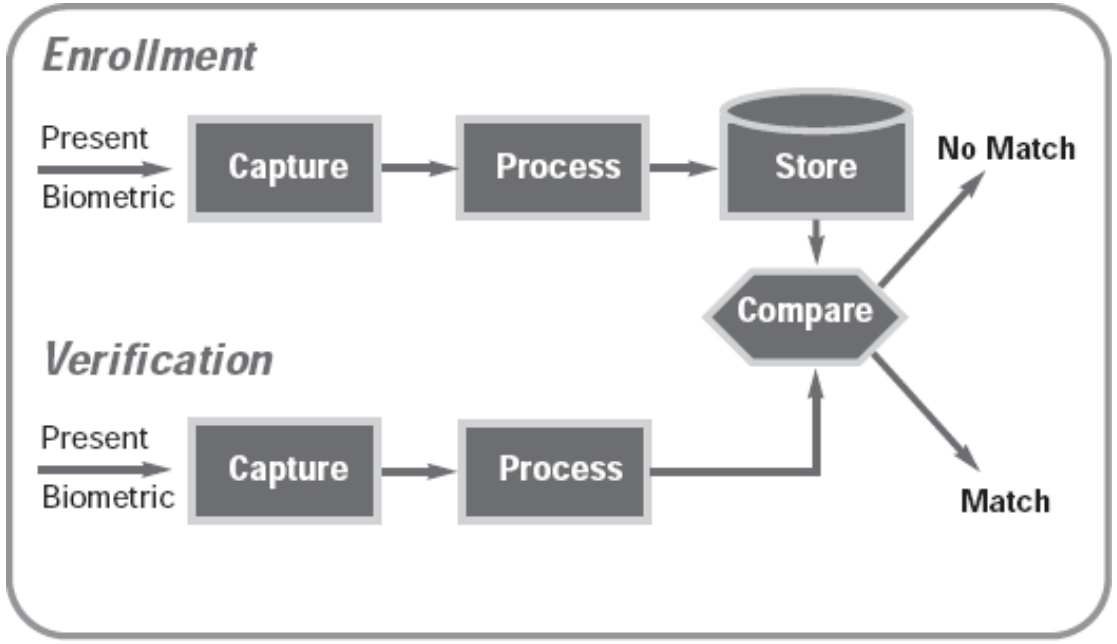
Biometrics authentication: 1) Enrollment



Biometrics authentication: 2) Verification



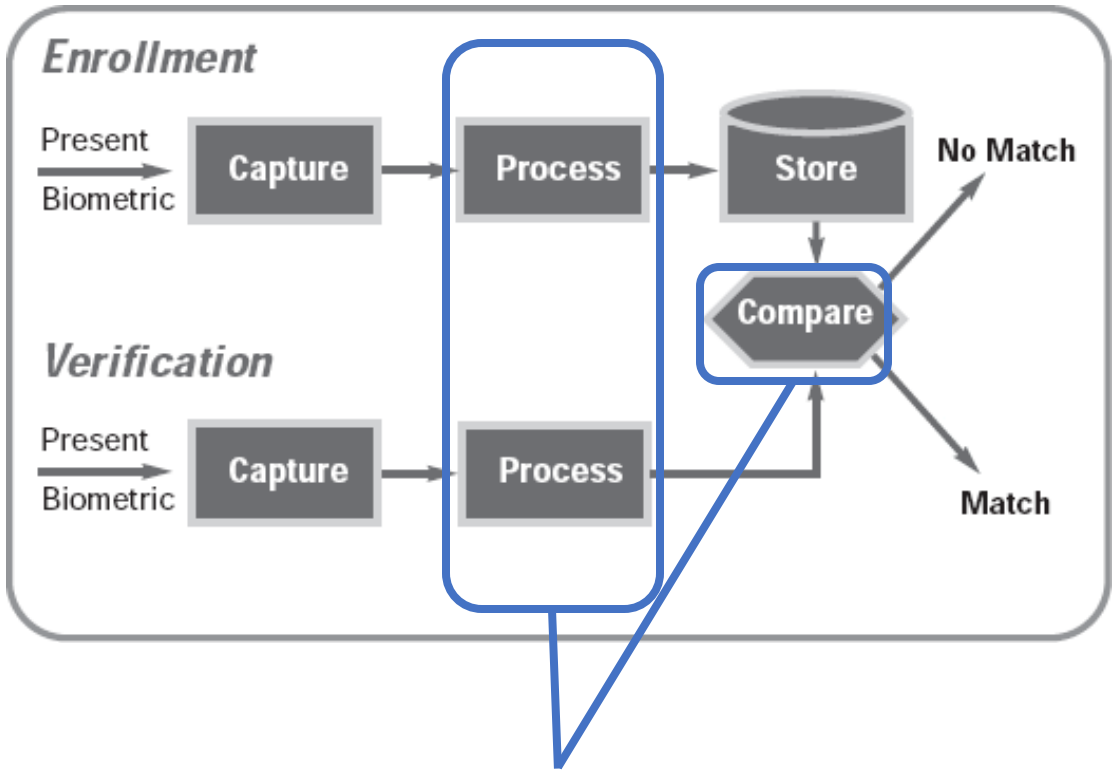
Biometrics authentication: 2) Verification



WHERE DO THESE PROCESSES HAPPEN?

CAPTURE	PROCESS	STORE
Local	Local	Local
Local	Local	Remote
Local	Remote	Remote

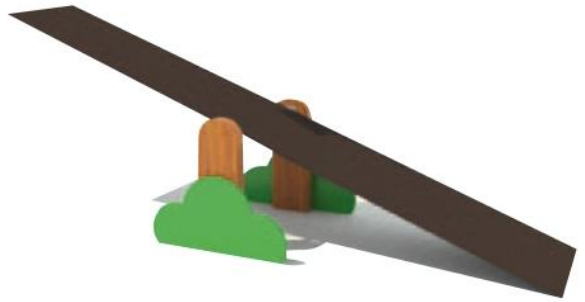
What you are: Biometrics



Parameters determine false positives and false negatives

Wrong authentications accepted

True authentications rejected



Decreasing false negatives increases false positives!!

Configuration depends on applications

Bank: low false positive even if legitimate users need to repeat

Gym: low false negative even if some non-users get in

Computer Security (COM-301)

Authentication

Tokens

Carmela Troncoso

SPRING Lab

carmela.troncoso@epfl.ch

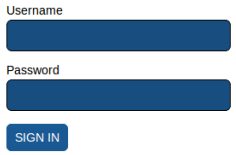
Some slides/ideas adapted from: Tuomas Aura, Yoshi Kohno, Trent Jaeger

Ways to Prove Who You Are

TRADITIONAL

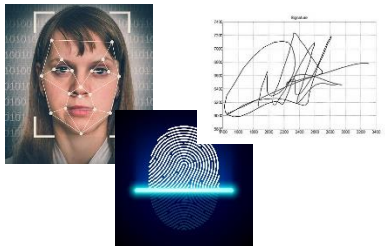
What you know

password, secret key



What you are

biometrics



What you have

Smart card, secure tokens

Problems with Biometrics

Hard to keep secret

Signature on ID card
Fingerprint left on glasses, door handle, ...
Photos (nowadays, everywhere!)

} **Liveness detection**

Revocation is difficult (impossible?)

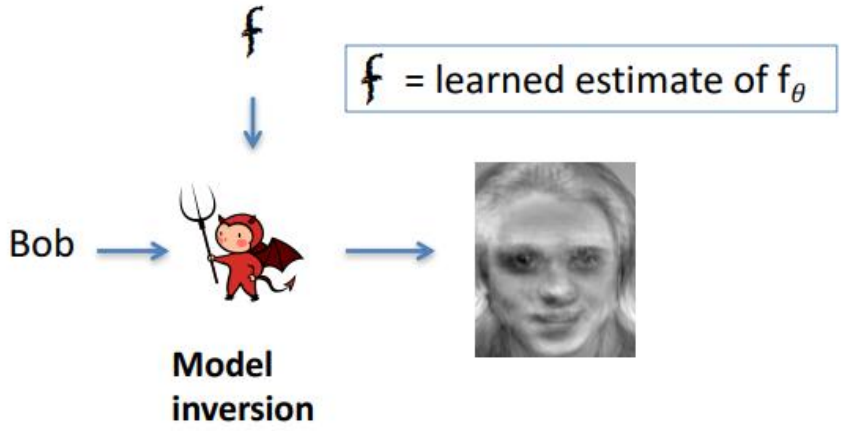
Sorry, your iris has been compromised, please create a new one... !

Identifiable and unique

Linking across systems

May reveal private information

Iris → disease
Face → identity



Source: Tom Ristenpart

Not always universal or immutable

Fingerprints disappear, iris changes with lenses,...

What you have: Tokens

What you have: Tokens

Step 1 – Offline - Initialization: token and server establish a common “seed” & synchronize their clocks

“seed” = common random number

What you have: Tokens

From then on - Operation: obtain a random number from the seed that can only be computed by the token

1. I am Rick

2. Prove it

What you have: Tokens

From then on - Operation: obtain a random number from the seed that can only be computed by the token

1. I am Rick

2. Prove it

3. The token first computes **n**, using the synchronized clock

$$n = \frac{\textbf{now-start}}{\textbf{interval}}$$

4. The token applies a **keyed** cryptographic function **f()** n times on seed

$$v = f^n(seed)$$

n=1 → v=f(seed);
n=2 → v=f(f(seed));
n=3 → v=f(f(f(seed)));
…

5. The token sends the result of the operation to the server

v

The adversary only sees an encrypted value.
Cannot know recover the seed, nor compute future values

6. The server computes n and realizes the same operation as the token

$$v' = f^n(seed)$$

7. The server compares the computed value v' with the received value v

$$v' == v?$$

Why the cryptographic function cannot be a hash

From then on - Operation: obtain a random number from the seed that can only be computed by the token

$v_n = h^n(seed)$

I am Rick →

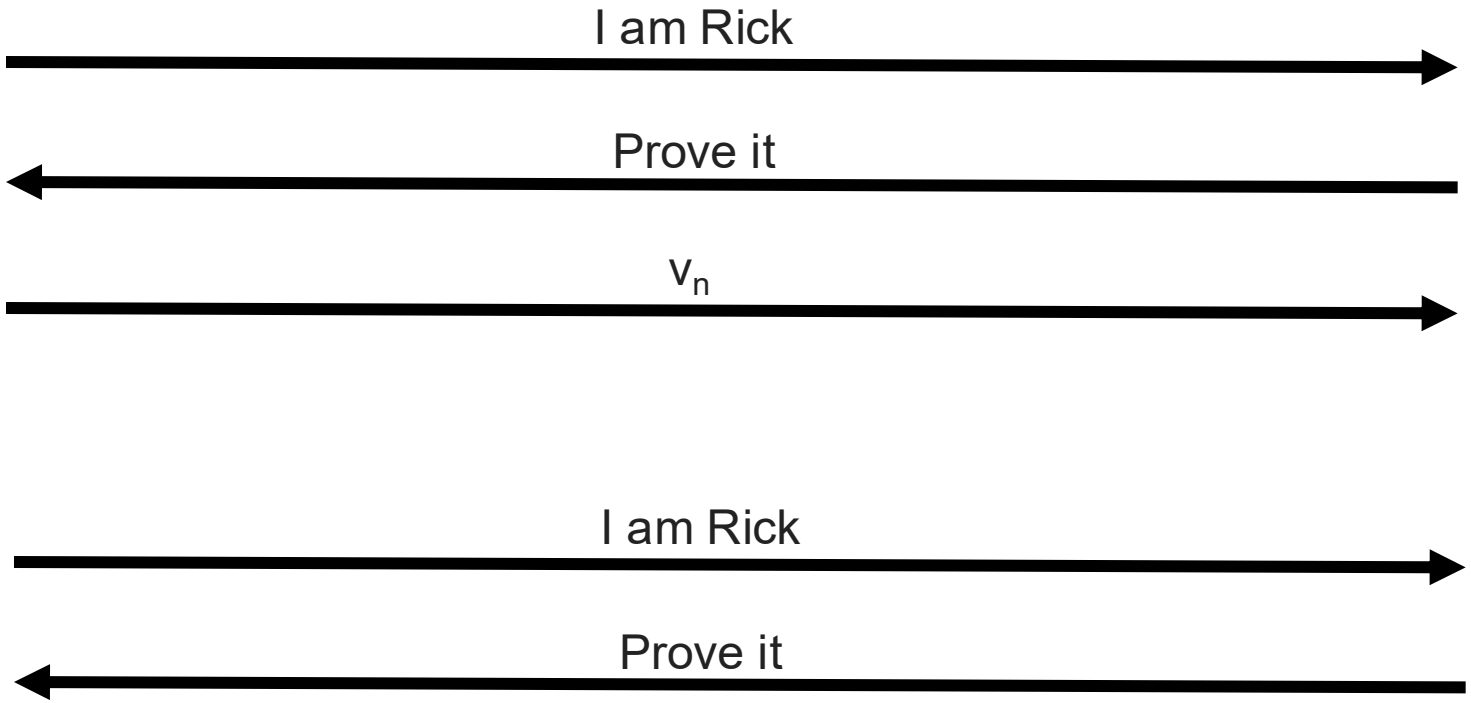
← Prove it

v_n →

Why the cryptographic function cannot be a hash

From then on - Operation: obtain a random number from the seed that can only be computed by the token

$v_n = h^n(seed)$



Why the cryptographic function cannot be a hash

From then on - Operation: obtain a random number from the seed that can only be computed by the token

$v_n = h^n(seed)$

I am Rick

Prove it

v_n

An adversary observing v_n can produce v_{n+1} !!!
The hash does not need a key! Anyone can compute it

I am Rick

Prove it

V_{n+1}

$v_{n+1} = h^{n+1}(seed)$
$= h(h^n(seed))$

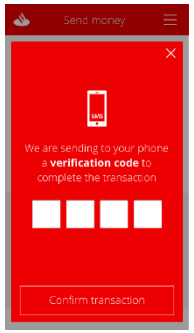
What you have: 2FA – Two factor authentication

Combine two out of the three factors:
(What you know, what you have, what you are)

what you have
what you know

what you have
what you know

what you have
what you know



what you have
what you know

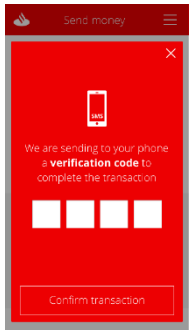
What you have: 2FA – Two factor authentication

Combine two out of the three factors:
(What you know, what you have, what you are)

Card = what you have
+ PIN = what you know

Token = what you have
Identification number = what you know

Token = what you have
(+ Card = what you have)
+ identification number = what you know



Modern approaches: mobile phone = what you have
The phone cannot hold a key (is not secure). Prove via SMS or showing a QR code

What machines have: Secret key

Use secret keys to produce **Digital signatures** to authenticate parties

e.g., used in internet protocols HTTPS/TLS to authenticate **the server**

(and can be used also to authenticate the client)

What machines have: Secret key

Use secret keys to produce **Digital signatures** to authenticate parties

e.g., used in internet protocols HTTPS/TLS to authenticate **the server**

(and can be used also to authenticate the client)

Building authentication protocols **is hard**!

defending from **man in the middle – Use signatures**

defending from **replay attacks – Use challenges / nonces**

What machines have: Secret key

Use secret keys to produce **Digital signatures** to authenticate parties

e.g., used in internet protocols HTTPS/TLS to authenticate **the server**

(and can be used also to authenticate the client)

Building authentication protocols **is hard**!

defending from **man in the middle – Use signatures**

defending from **replay attacks – Use challenges / nonces**

Still difficult to get right!

Use well established protocols!! (TLS 1.3, ISO 9798-3)

Summary of the lecture

Authentication is the process by which an entity proves its identity

Three flavours:

- What you know: passwords – hard to handle!
- What you are: biometrics – difficult to revoke and not infallible
- What you have: tokens – usability issues (you need to have the device

Machines authenticate using keys