

# Security & Privacy - Part 2

## Password Management

Course Esilv 2025

[arnaud.nauwynck@gmail.com](mailto:arnaud.nauwynck@gmail.com)

# Outline

Http Authorization

Transport: Need For confidentiality

Backend-end: storing password in Database ?

Cryptographic "Hash" functions, Salt

Who is "John The Ripper" ?

2FA & MFA (Multi Factor Authentication)

# Outline



## Http Authorization

Transport: Need For confidentiality

Backend-end: storing password in Database ?

Cryptographic "Hash" functions, Salt

Who is "John The Ripper" ?

2FA & MFA (Multi Factor Authentication)

# login/password in Http Request

```
curl -vv http://localhost:8080/page1.html -u user:${PASSWORD}
```

```
Cmder
arnaud@DesktopArnaud /cygdrive/c/arn/devPerso
$ curl -vv http://localhost:8080/page1.html -u user:${PASSWORD}
* Trying [::1]:8080...
* Connected to localhost (::1) port 8080
* Server auth using Basic with user 'user'
> GET /page1.html HTTP/1.1
> Host: localhost:8080
> Authorization: Basic dXNlcj0zMmI0MzE2My1lYjY0LTQyOGItYjYxNi0wOTQ5MDNhZTE3ZTY=
> User-Agent: curl/8.4.0
> Accept: */*
>
< HTTP/1.1 200
< Vary: Origin
```

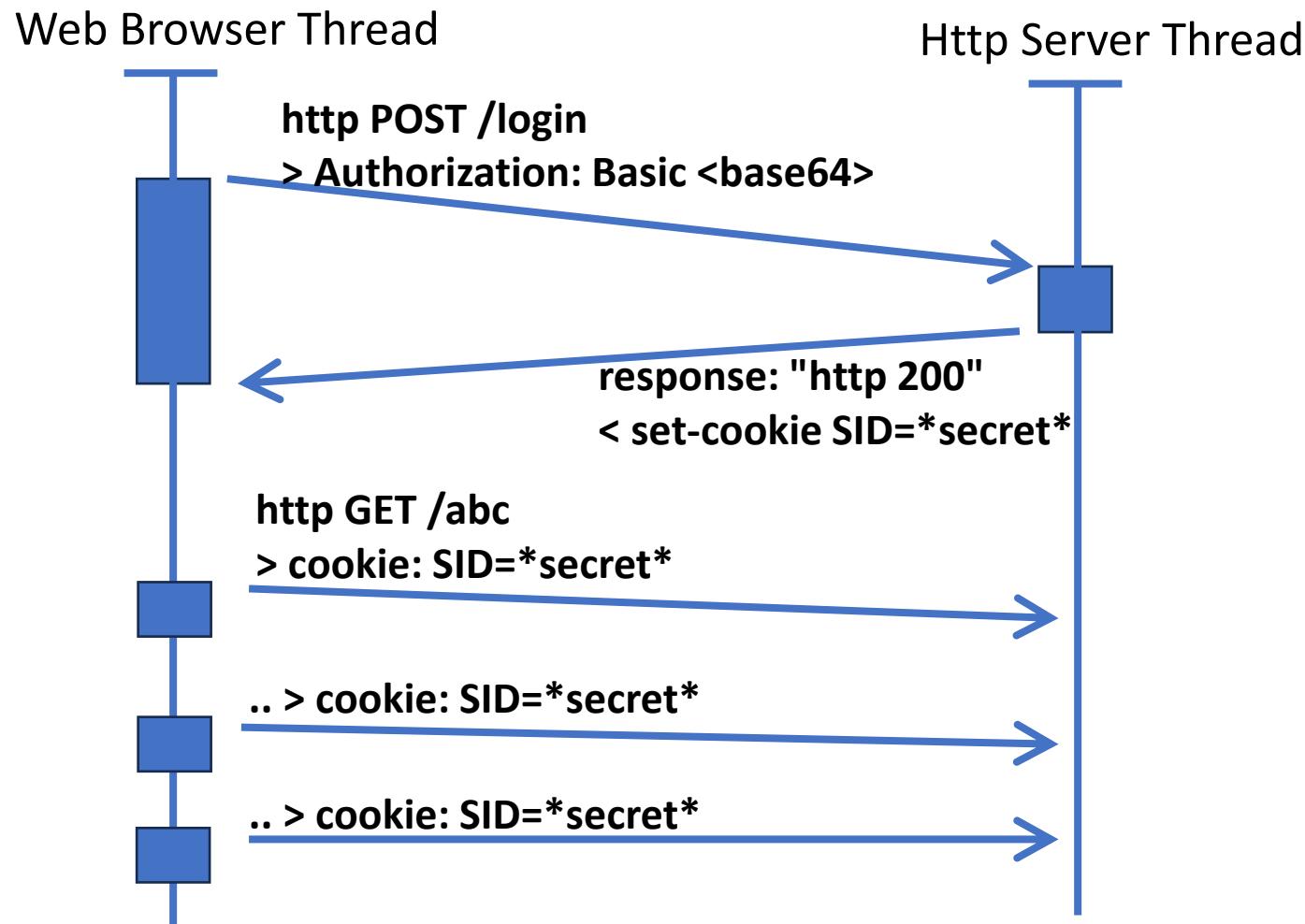
```
Notice: curl -u user:${password}
> Authorization: Basic dXNlcjpwYXNzd29yZA==
```

user:password is sent as Http Header "Authorization: Basic <<base64>>"  
in **CLEAR TEXT** (encoded in base 64)

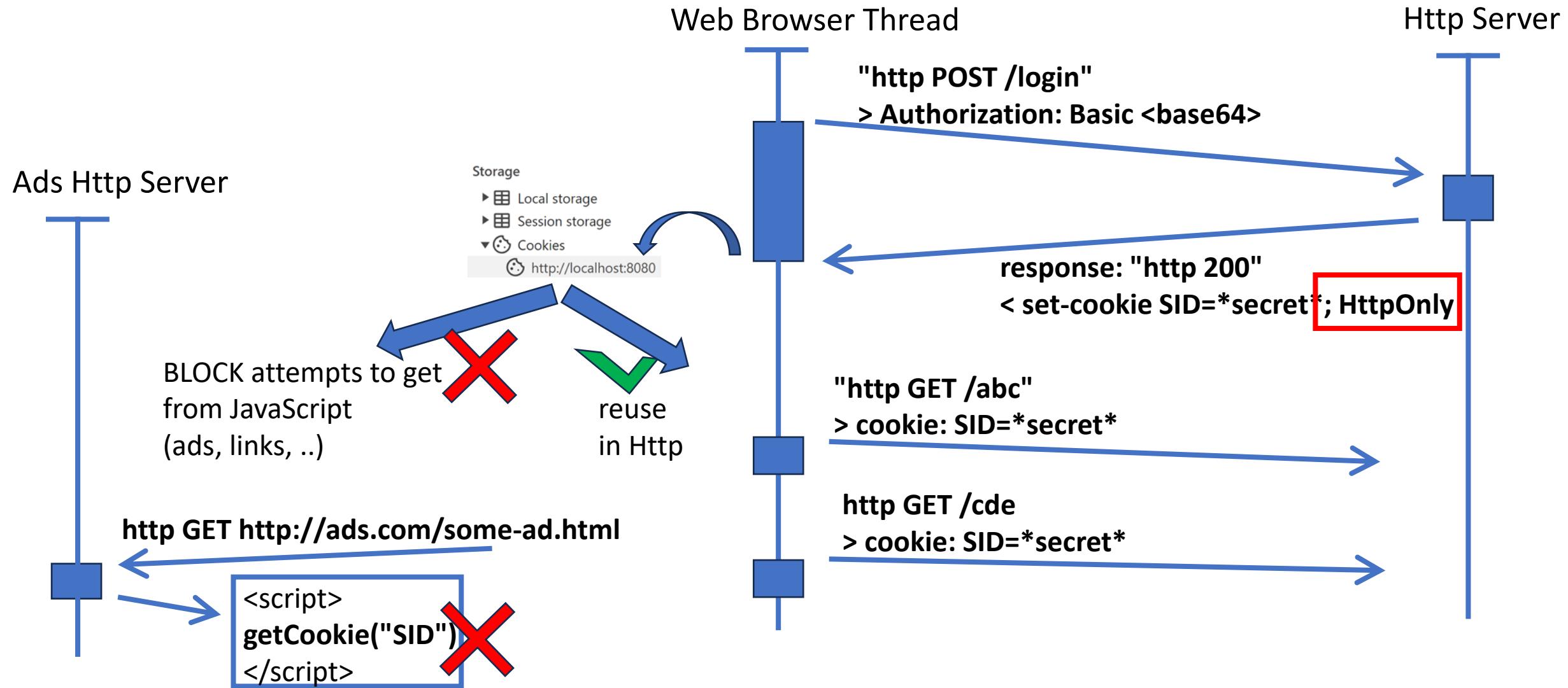
to decode, use "base64 -d"

```
Cmdr
$ echo dXNlcjpwYXNzd29yZA== | base64 -d
user:password
```

# Give Password Once, then temporary SESSIONID using "set-cookie"



# Cookie restriction protections: ";HttpOnly"



# Cookie are "hidden" to End-Users

## How to explicitly give ShareableLink with \*secret\* ?

1/ in URL path

`http://host:port/some-resource/path/**secret-very-long-so-do-not-read**`

2/ in URL query parameter

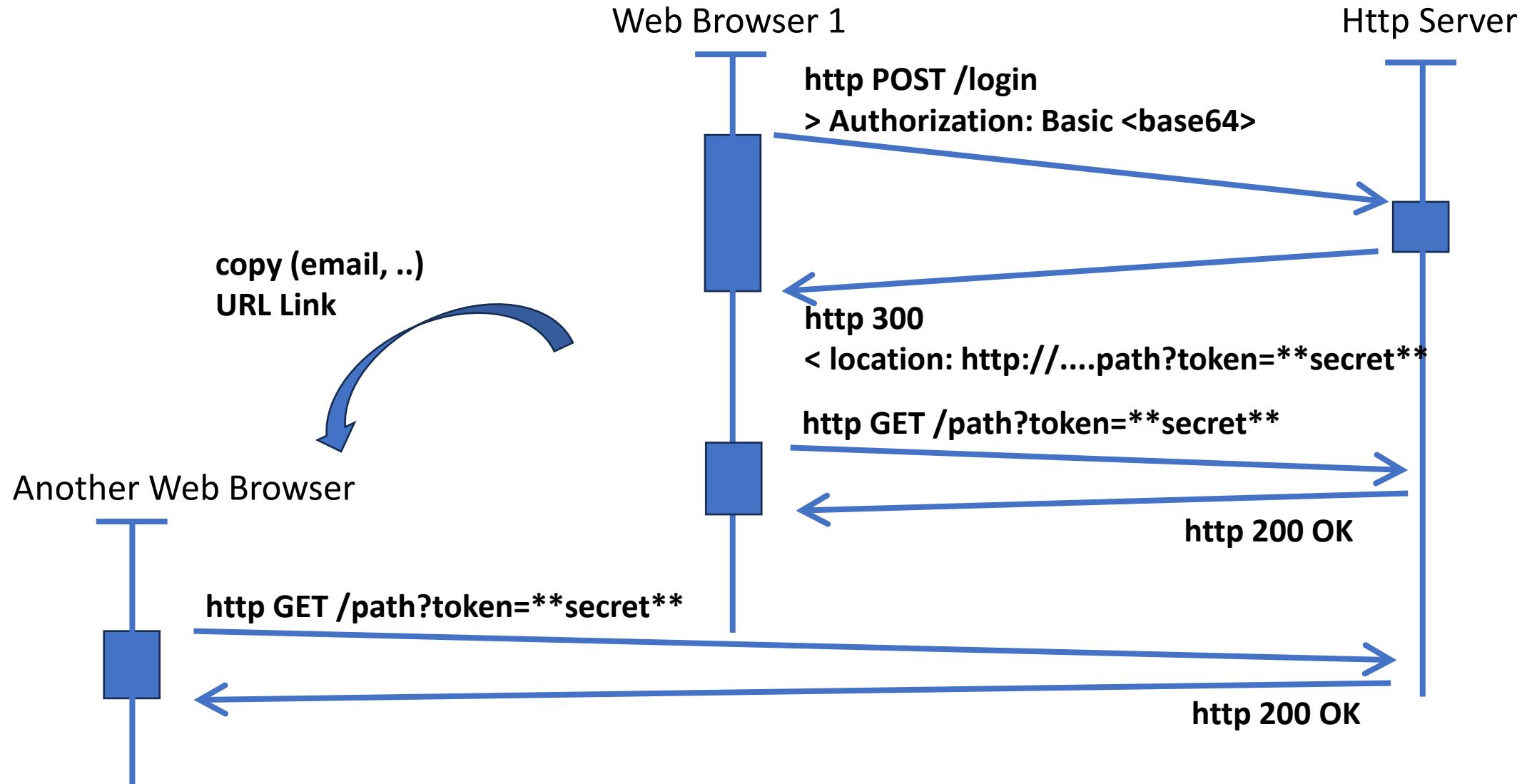
`http://host:port/some-resource/path?param=**secret**`

Notice: impossible in Anchors (not transmitted to servers)

`http://host:port/some-resource/path#anchor-**secret**`

# Shareable URL Link using http query param

`http://....path?token=**secret**`



# Recommandations for url query-param

http://...path?token=\*\*secret\*\*

WARNING : many http servers and equipments (Firewall, Proxies)  
write request URL in log files

Screen sharing, and copy+paste traces can cause unwanted data leakage (by mistake)

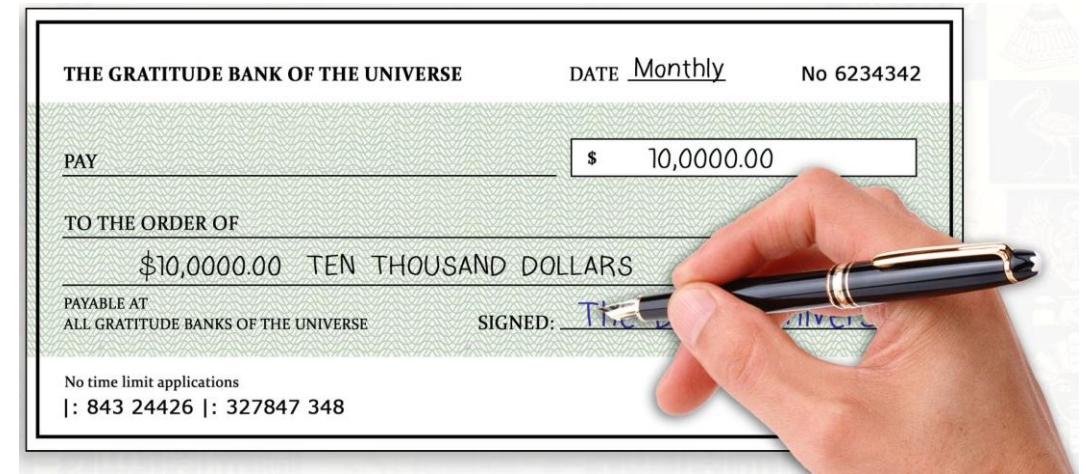
Your neighbor (malicious, camera over your shoulder) can steal secret

# Problem & Analogy of giving Password

Give Identity



Give copy reusable Signature  
Give Blank Check



You want to "Sign"  
(You have to Prove Your ID),  
But put restrictions  
with Amount/Orders/Date information

How to do it for replacing  
your login/password form submission  
on web sites ?

# Using Cryptographic "Claim" Signature instead of Identity & Password ?

asymmetric Key Pair:

Public  
Key

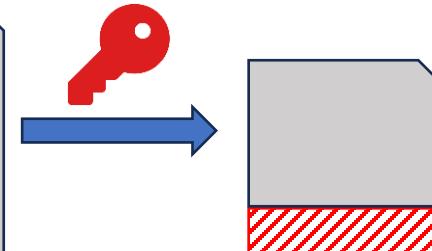


Private  
Key

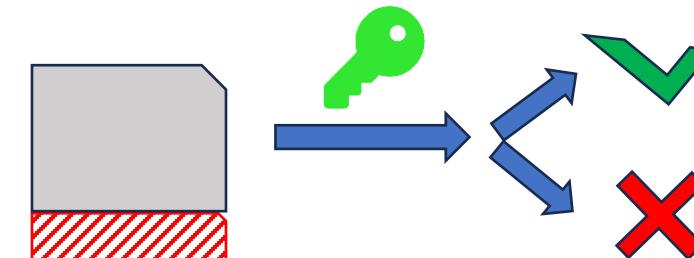


**Sign Data (you only)**

Me "John Smith"  
authorize using  
my power of doing  
"Xyz" Action  
to Bearer of this token  
until 2024-02-03



**Verify (everybody) if valid signature**



# Remind on Security Principles

Always Give **Strict minimum** Access rights to user  
so they can do their authorized actions  
with conditions (time, ..)

Do not give "root" password,  
do not give non-expiring passwords,  
change your passwords frequently.

password <-> Identity    !=    claim, permission, role

Simply do not give your password, only prove your identity / permissions

# User != Group != Role != Permission

Very often, User are associated to Group, which have Roles, and therefore "have" Permissions



Counter examples: John Smith is a "Trader" in a Bank => "has" permission to trade ?  
Maybe

- 1/ Only to buy-sell "Equities" products, in some portfolios Xyz, with limits
- 2/ Only if he is strongly authenticated with SmartCard
- 3/ Only At Bank Office (not home)
- 4/ Only during normal office hours (not midnight in week-end)

# Standard J.W.T. = Json Web Token

**split 3 parts with "."**



<header in base64> . <payload in base 64> . <signature>

**decode base64**



header object in json

```
{ "meta-key1": "value1",
  "meta-key2": "value2"
}
```

payload object in json

```
{ "user": "user-xyz",
  "claim": "permission-123",
  "other-key": "value"
}
```



Cryptographic  
Signature

# JWT Example

eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.**eyJzdWliOilxMjM0NTY3ODkwliwibmFtZSI6IkpvaG4gRG9IiwiWF0IjoxNTE2**

MjM5MDIyfQ.**SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c**

eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9

=>

base64 decode in Javascript

**atob("eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9")**



{"alg": "HS256", "typ": "JWT"}

eyJzWli ...

=>

base64 decode in Javascript

**atob("eyJzWli ...")**



```
{"sub": "1234567890",
"name": "John Doe",
"iat": 1516239022
}
```

SflKxwR...

check if



Cryptographic  
with Public Key

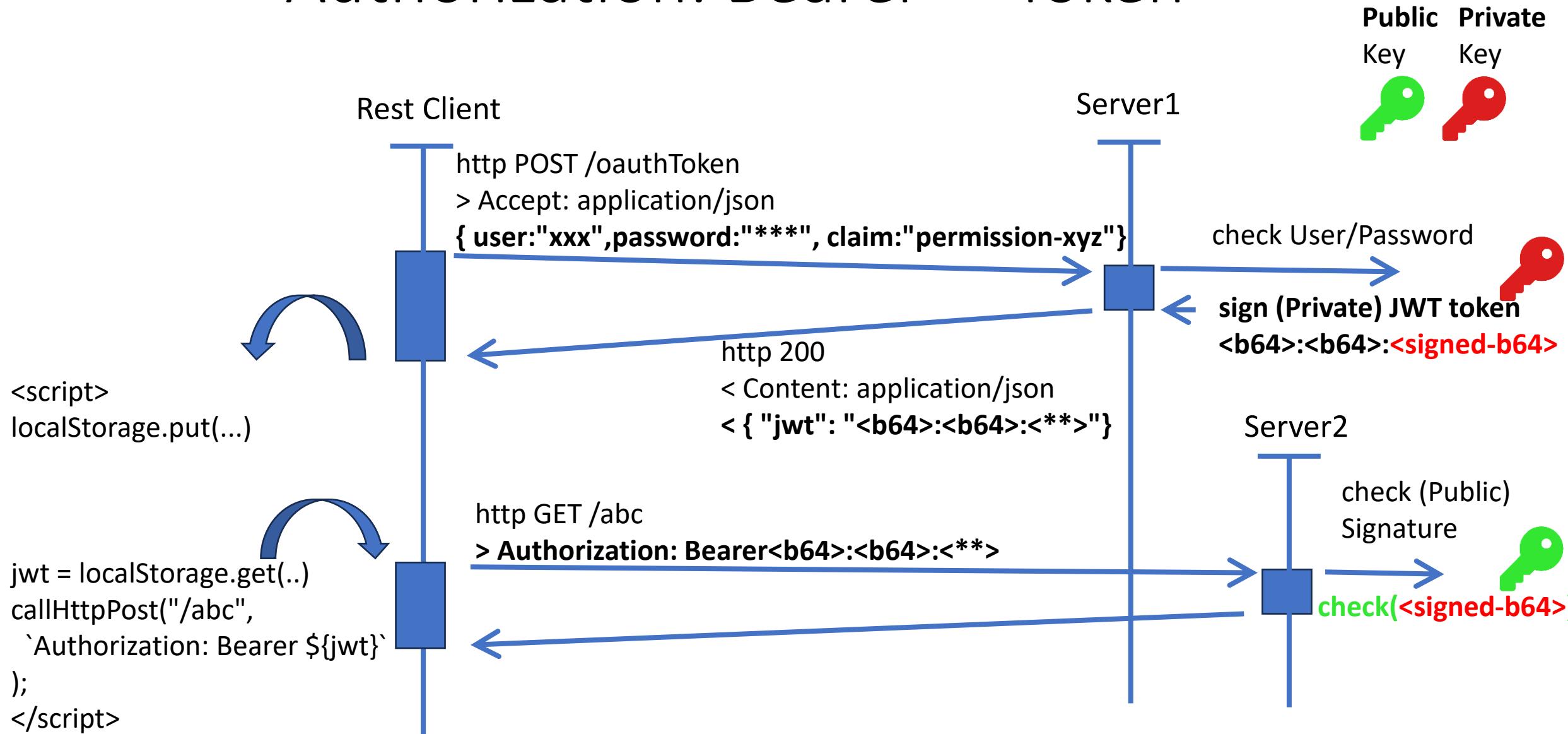


check if  
not expired



# Typical usage in SPA

## "Authorization: Bearer <<Token>>"



# Outline

Http Authorization



**Transport: Need For confidentiality**

Backend-end: storing password in Database ?

Cryptographic "Hash" functions, Salt

Who is "John The Ripper" ?

2FA & MFA (Multi Factor Authentication)

# Problem of Confidentiality

TCP / IP low-level protocols are UNSECURE

Internet signals (electric / radio / fiber channel) are physically NOT secured

Internet is based on TCP-IP

=> consider that everybody could read/write/drop/replay/modify your TCP-IP packets !!

# Http is a clear-text protocol (ASCII lines)

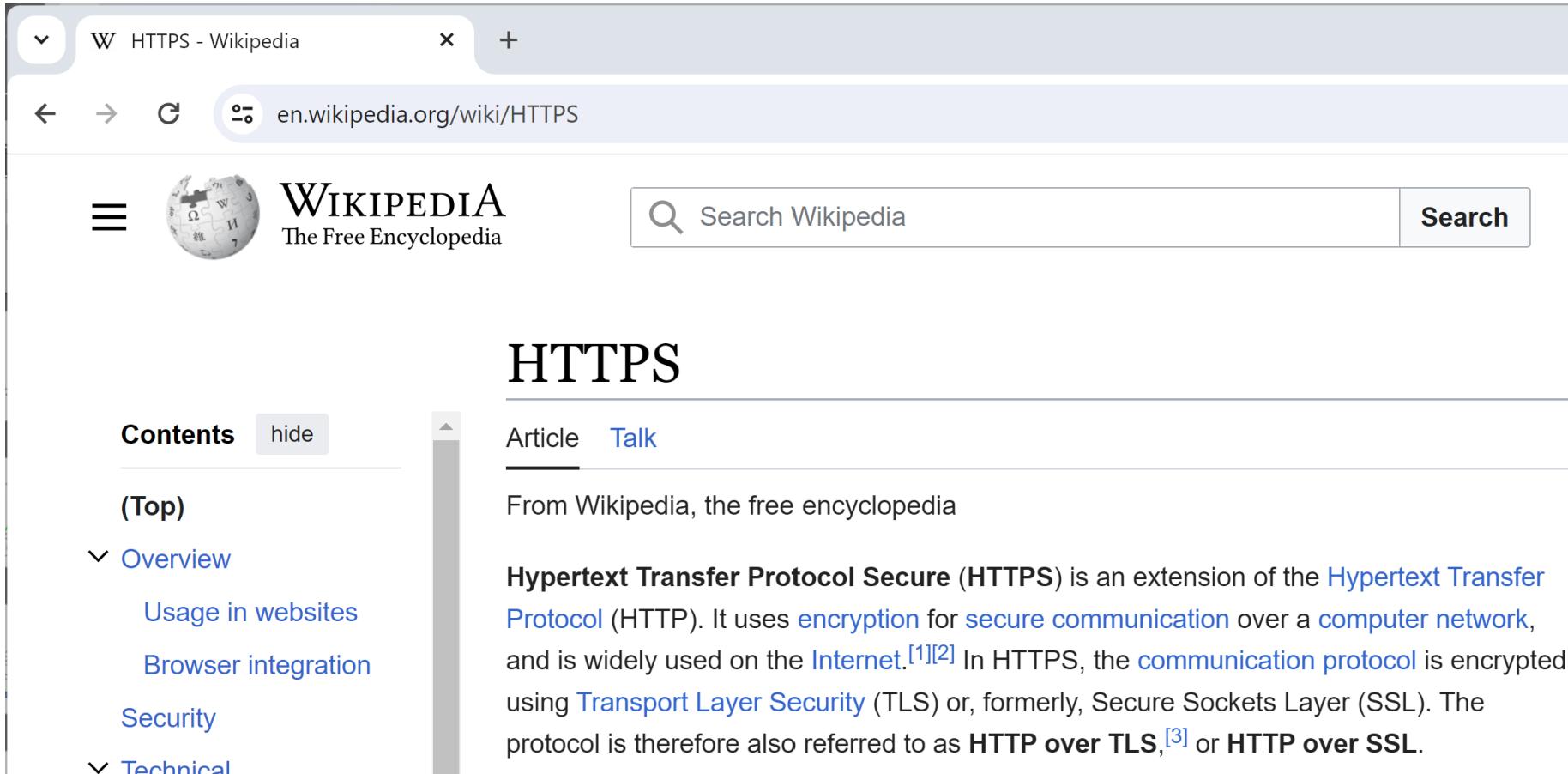
Easy to read (or decode) all information

In particular Http Headers "Authorization: Basic <base64>"

Protocol does not include security feature

Protocol need to be nested (wrapped) in another secured one

# https ... S = Secured, but is it ?



The screenshot shows a web browser window displaying the Wikipedia article on HTTPS. The title bar reads "W HTTPS - Wikipedia". The address bar shows the URL "en.wikipedia.org/wiki/HTTPS". The main content area features the Wikipedia logo and the word "WIKIPEDIA" with the subtitle "The Free Encyclopedia". A search bar with a magnifying glass icon and the placeholder "Search Wikipedia" is visible. Below the search bar is a "Search" button. On the left side, there is a sidebar with a "Contents" section containing links to "Overview", "Usage in websites", "Browser integration", "Security", and "Technical". The main article content starts with the heading "HTTPS" and includes a link to "From Wikipedia, the free encyclopedia". The main text explains that Hypertext Transfer Protocol Secure (HTTPS) is an extension of HTTP, using encryption for secure communication over a computer network, and is widely used on the Internet.

## HTTPS

Article   [Talk](#)

From Wikipedia, the free encyclopedia

**Hypertext Transfer Protocol Secure (HTTPS)** is an extension of the **Hypertext Transfer Protocol** (HTTP). It uses [encryption](#) for [secure communication](#) over a [computer network](#), and is widely used on the [Internet](#).<sup>[1][2]</sup> In HTTPS, the [communication protocol](#) is encrypted using [Transport Layer Security](#) (TLS) or, formerly, [Secure Sockets Layer](#) (SSL). The protocol is therefore also referred to as [HTTP over TLS](#),<sup>[3]</sup> or [HTTP over SSL](#).

# Nesting Protocols

## Nesting Doll / Onion Model Metaphors

[https://en.wikipedia.org/wiki/Matryoshka\\_doll](https://en.wikipedia.org/wiki/Matryoshka_doll)

/wiki/Matryoshka\_doll

As metaphor [ edit ]

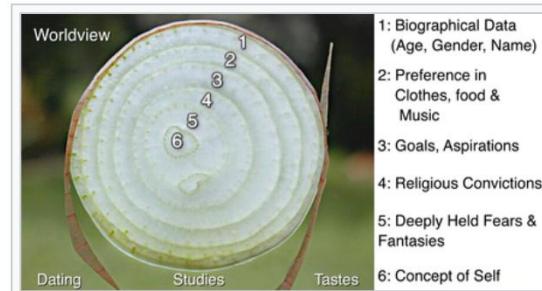


**Nesting and Onion Metaphors** [ edit ]

Matryoshkas are also used [metaphorically](#), as a [design paradigm](#), known as the "matryoshka principle" or "nested doll principle"[\[citation needed\]](#). It denotes a recognizable relationship of "object-within-similar-object" that appears in the design of many other natural and crafted objects. Examples of this use include the [matrioshka brain](#)[\[citation needed\]](#), the [Matroska](#) media-container format[\[citation needed\]](#), and the Russian Doll model of [multi-walled carbon nanotubes](#)[\[citation needed\]](#).

[https://en.wikipedia.org/wiki/Onion\\_model](https://en.wikipedia.org/wiki/Onion_model)

The **onion model** is a [graph-based diagram](#) and conceptual model for describing relationships among levels of a hierarchy, evoking a [metaphor](#) of the layered "shells" exposed when an [onion](#) (or other concentric assembly of [spheroidal](#) objects) is bisected by a plane that intersects the center or the innermost shell. The outer layers in the model typically add size and/or complexity, incrementally, around the inner layers they enclose.



# Https = TLS Transport (on top of TCP-IP) ... with data interpreted as Http data

/wiki/Transport\_Layer\_Security

Wikipedia



Search Wikipedia

Search

## Transport Layer Security

Article [Talk](#)

From Wikipedia, the free encyclopedia

**Transport Layer Security (TLS)** is a [cryptographic protocol](#) designed to provide communications security over a computer network. The [protocol](#) is widely used in applications such as [email](#), [instant messaging](#), and [voice over IP](#), but its use in securing [HTTPS](#) remains the most publicly visible.

The TLS protocol aims primarily to provide security, including [privacy](#) (confidentiality), integrity, and authenticity through the use of [cryptography](#), such as the use of [certificates](#), between two or more communicating computer applications. It runs in the [presentation layer](#) and is itself composed of two layers: the TLS record and the TLS [handshake protocols](#).

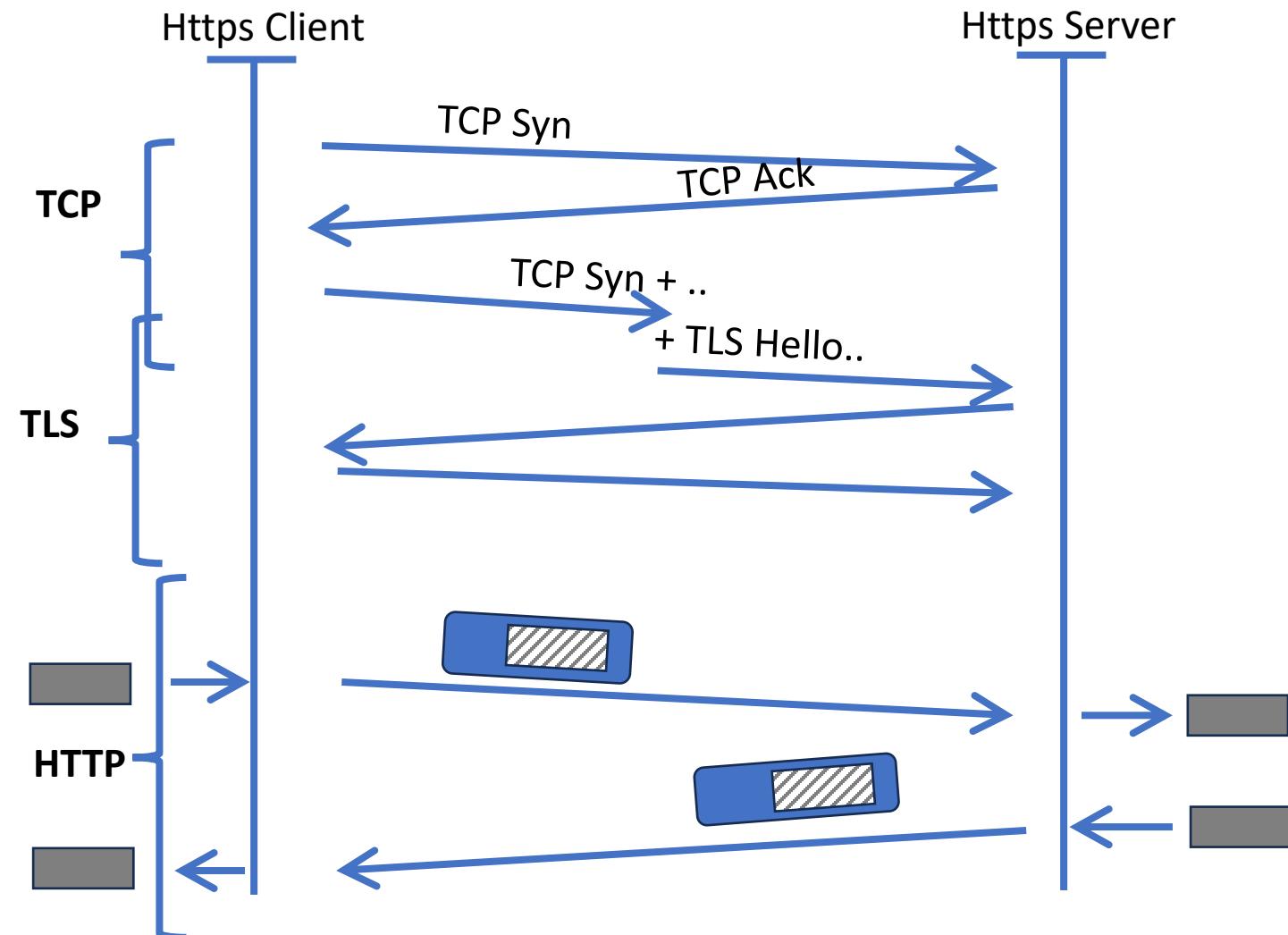
# Https TLS-Socket Handshake over TCP/IP Packets

step 1:  
establish **TCP socket** over IP

step 2:  
generate Temp Pub/Private Key  
exchange Pub Key  
exchange AES session Key  
= establish **TLS socket** over TCP/IP

step 3,4..\*: establish **Https** over TLS  
wrap each sent message data within TLS packet  
encrypt using AES key

unwrap each received TLS packet,  
decrypt payload data using AES key



# (Https) TLS has 2 phases / use 2 Cryptographic Algorithms

## **Phase 1 : TLS HandShake (slow, once)**

Check Server Certificate

Generate Public-Private Key Pair,  
then Exchange PubKey

... using **Asymmetric Key Cryptographic Protocols**

**typically RSA, or ECC : Elliptic Curve (Diffie-Hellman)**

RSA or ECC

**Asymmetric Keys Pair**

Public      Private  
Key      Key



Generate then exchange temporary Symmetric Key, exchange once  
(may regenerate from time to time)

**AES-256**

Symmetric Key

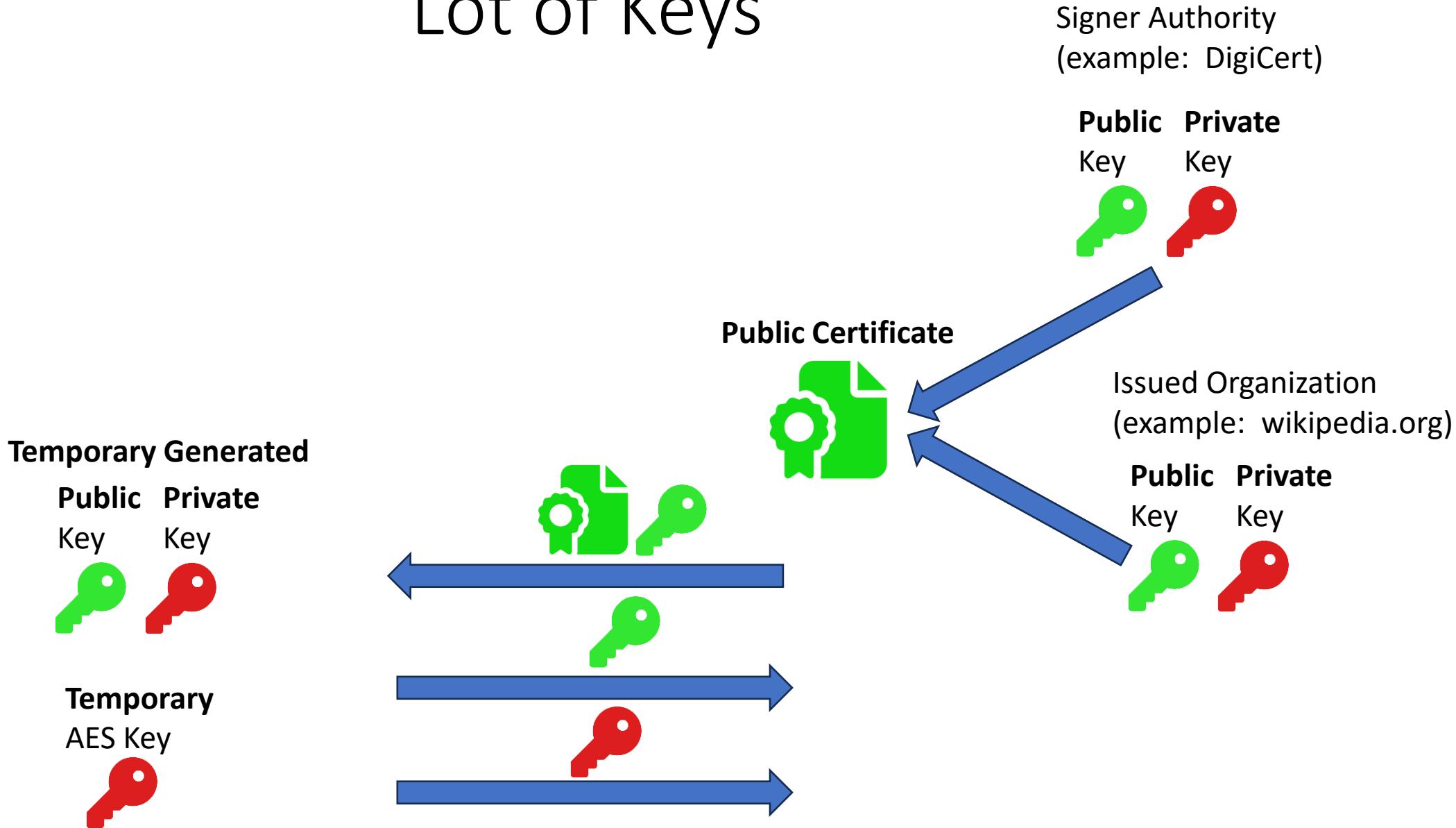
## **Phase 2 : Data encryption (fast, repeated)**

... using **Symmetric Key Cryptographic Protocols**

**typically AES-256**



# Lot of Keys



# Click on Site Information, check Certificate

The screenshot shows a web browser window for Wikipedia over HTTPS. A red arrow points from the top-left of the browser window to the padlock icon in the address bar. Another red arrow points from the bottom-left of the browser window to the "Connection is secure" message in the site information dropdown. A third red arrow points from the bottom-right of the browser window to the "Protocol Secure (HTTPS)" link in the page content. A fourth red arrow points from the right side of the browser window to the "Security" panel on the right, which displays the "Certificate is valid" status.

W HTTPS - Wikipedia

en.wikipedia.org/wiki/HTTPS

en.wikipedia.org

Connection is secure

Cookies and site data

Site settings

About this page

Protocol Secure (HTTPS)

Protocol (HTTP). It uses encryption for secure co

Security

en.wikipedia.org

Connection is secure

Your information (for example, passwords or credit card numbers) is private when it is sent to this site. [Learn more](#)

Certificate is valid

What is a "Certificate" ?

Can it be not valid.. yes

# Certificate (& Details)

**Certificate Viewer: \*.wikipedia.org**

**General** **Details** 

**Issued To**

Common Name (CN)	*.wikipedia.org
Organization (O)	Wikimedia Foundation, Inc.
Organizational Unit (OU)	<Not Part Of Certificate>

**Issued By**

Common Name (CN)	DigiCert TLS Hybrid ECC SHA384 2020 CA1
Organization (O)	DigiCert Inc
Organizational Unit (OU)	<Not Part Of Certificate>

**Validity Period**

Issued On	Wednesday, October 18, 2023 at 2:00:00 AM
Expires On	Thursday, October 17, 2024 at 1:59:59 AM

**SHA-256 Fingerprints**

Certificate	07bcde69c024ab9a2591b92fb55107408e927fe775cd825e7f4b4b3eac8 0f026
Public Key	075ecc2685ba06ef4797e252b4b4f8830fa09b79d20d0de5c6e8fac4a42 b8e65

**Certificate Viewer: \*.wikipedia.org**

**General** **Details**

**Certificate Hierarchy**

- ▼ DigiCert Global Root CA
  - ▼ DigiCert TLS Hybrid ECC SHA384 2020 CA1
    - \*.wikipedia.org

**Certificate Fields**

- ▼ \*.wikipedia.org
  - ▼ Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
  - Issuer
  - ▼ Validity
    - Not Before



# More on TLS & Certificates in next Lessons

## Asymmetric Cryptography with Public-Private Key

How to write to someone (website) you have never seen before  
and still be sure of its identity ??

Which secret key to choose to talk to him,  
as you never exchanged before a key in privacy with him ?

How do you know you are really talking to him, not to a "Man in the Middle" ?  
<=> How do you trust that nobody can decode your conversation

# Outline

Http Authorization

Transport: Need For confidentiality



**Backend-end: storing password in Database ?**

Cryptographic "Hash" functions, Salt

Who is "John The Ripper" ?

2FA & MFA (Multi Factor Authentication)

# Do you need to store passwords in Databases ?

NO

Use OAuth2 redirections to other safe"r"(?) sites

Use 2FA, MFA (MultiFactor)

Example with SmartCards, IdCard, SMS

Already Yes? Connect to your LDAP server

Connect to your Cloud Provider Identity Realm

By the way ... Are you RGPD compliant ?

# Personnal Password Database File : "KeePass"

Do not store your password un-encrypted on your PC !

Do not share your screen with passwords in clear-text

=> Use KeePass application



using MasterPassword



practical, but UNSAFE  
(No Master Password)

you are never prompted to enter password  
on registered websites ! ...  
BUT password are transmitted automatically  
.. it relies on your Windows log-in session !

# KeePass

← → ⌂ keepass.info



**KeePass**  
Password Safe

This is the official website of KeePass, the free, open source, light-weight and easy-to-use password manager.

[Awards] [RSS Feed]

**Latest News**

**KeePass 2.56 released**  
2024-02-04 13:35. [Read More »](#)

**KeePass 1.42 released**  
2024-02-01 14:10. [Read More »](#)

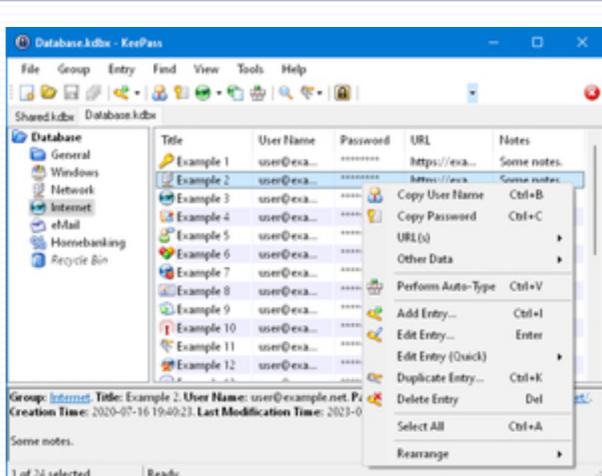
**KeePass 2.55 released**  
2023-10-12 15:31. [Read More »](#)

**KeePass 2.54 released**  
2023-06-03 15:28. [Read More »](#)

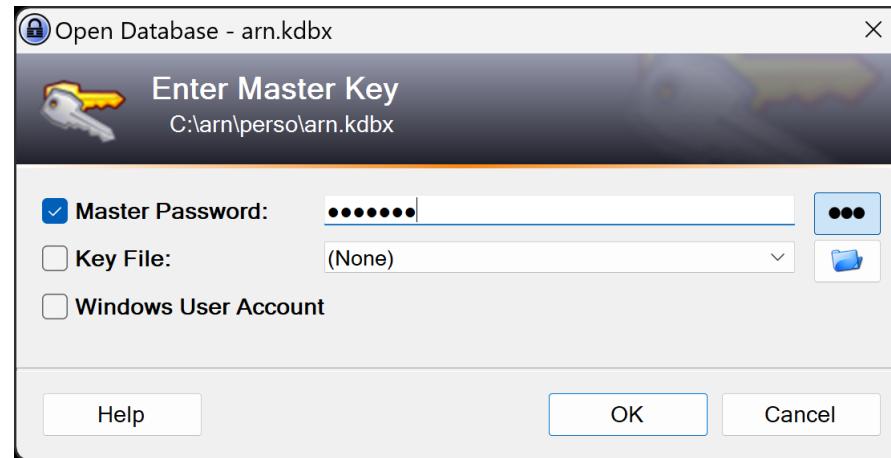
[\[News Archive\]](#)

**Why KeePass?**  
Today, you have to remember many passwords. You need a password for a lot of websites, your e-mail account, your webserver, network logins, etc. The list is endless. Also, you should use a different password for each account, because if you would use only one password everywhere and someone gets this password, you would have a problem: the thief would have access to *all* of your accounts.

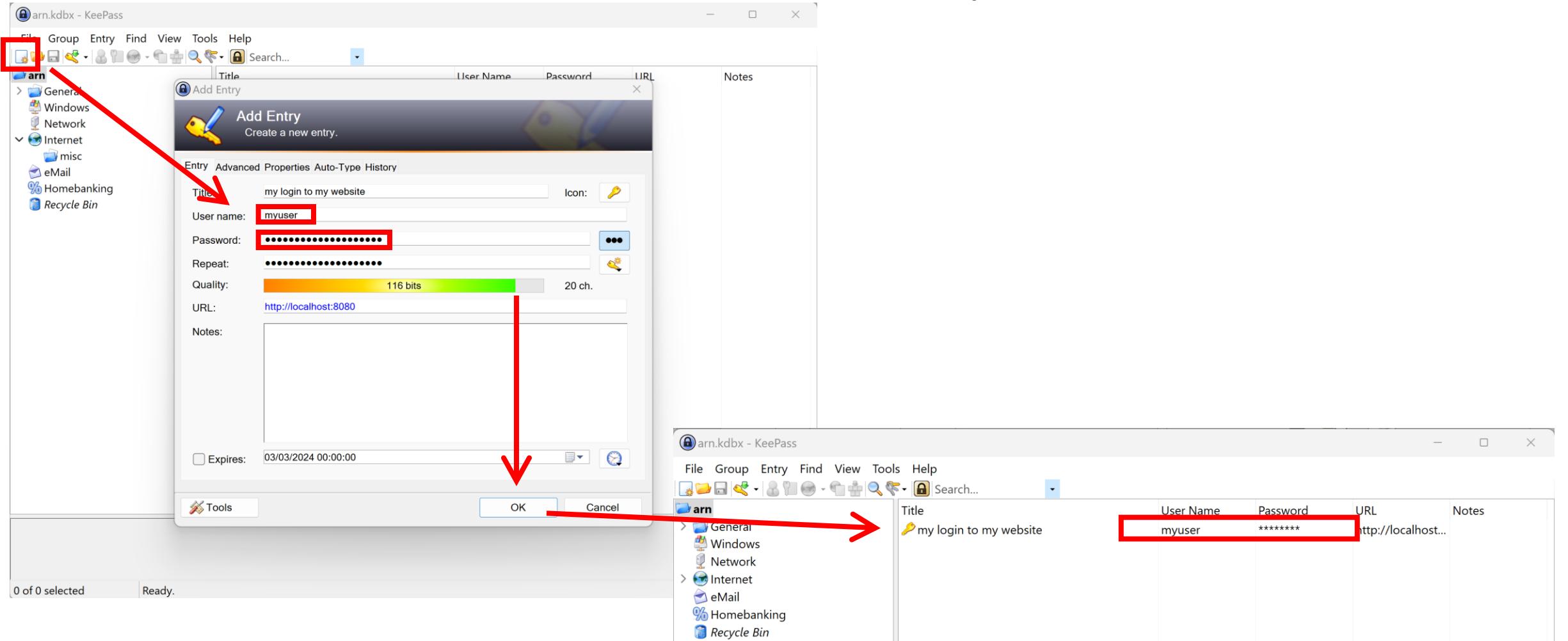
KeePass is a free open source password manager, which helps you to manage your passwords in a secure way. You can store all your passwords in one database, which is locked with a master key. So you only have to remember one single master key to unlock the whole database. Database files are encrypted using the best and most secure encryption algorithms currently known (AES-256, ChaCha20 and Twofish). For more information, see the [features](#) page.



KeePass = your.kdx file  
+ your MasterPassword



# Add an Entry

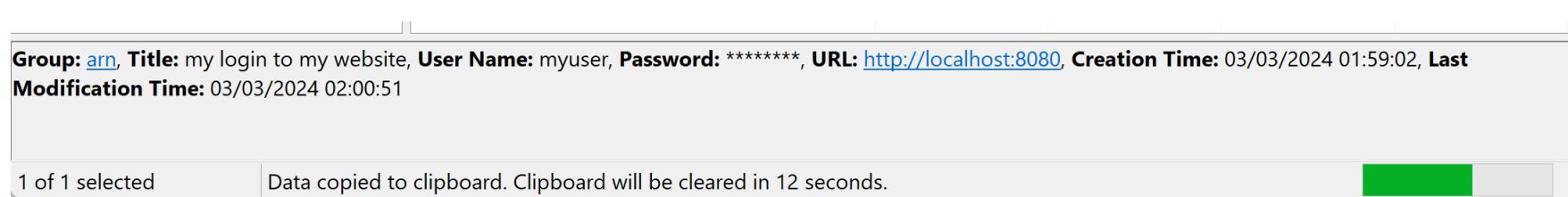


# KeePass

**Ctrl+B** => copy username entry to clipboard

**Ctrl+C** => copy password entry to clipboard (for 10 seconds)  
without showing

**Ctrl+V** => autotype field username, then field password in last page



# What About Passwords Databases, used by Web App servers ?

Remember the Security proverb :

There are only 2 kinds of app servers/softwares in the world :

- those that have been **already HACKED**
- those that will be **HACKED soon**

# Example of Scandals Hacked Users Databases



scandals of company stolen password database

Videos

News

Images

2021

Books

Finance

About 2,070,000 results (0.33 seconds)

<https://www.csoonline.com/article/534628/the-biggest-data-breaches-of-the-21st-century.html>

## 1. Yahoo

**Date:** August 2013

**Impact:** 3 billion accounts

Securing the number one spot – almost seven years after the initial breach and four since the true number of records exposed was revealed – is the attack on Yahoo. The company first publicly announced the incident – which it said took place in 2013 – in December 2016. At the time, it was in the process of being acquired

## 2. Aadhaar [tie with Alibaba]

**Date:** January 2018

**Impact:** 1.1 billion Indian citizens' identity/biometric information exposed

## 2. Alibaba [tie with Aadhaar]

**Date:** November 2019

**Impact:** 1.1 billion pieces of user data

# Scandals 4..9

## 4. LinkedIn

**Date:** June 2021

**Impact:** 700 million users

## 5. Sina Weibo

**Date:** March 2020

**Impact:** 538 million accounts

## 6. Facebook

**Date:** April 2019

**Impact:** 533 million users

## 7. Marriott International (Starwood)

**Date:** September 2018

**Impact:** 500 million customers

## 8. Yahoo

**Date:** 2014

**Impact:** 500 million accounts

Making its second appearance in this list is Yahoo, which suffered an attack in 2014 separate to the one in 2013 cited above. On this occasion, state-sponsored actors stole data from 500 million accounts including names, email addresses, phone numbers, **hashed passwords**, and dates of birth. The company took initial

## 9. Adult Friend Finder

**Date:** October 2016

**Impact:** 412.2 million accounts

What's more, the vast majority of the exposed **passwords** were hashed via the notoriously **weak algorithm SHA-1**, with an estimated 99% of them cracked by the time LeakedSource.com published its analysis of the data set on November 14, 2016.

# Scandals 10..15

## 10. MySpace

**Date:** 2013

**Impact:** 360 million user accounts

According to [the company](#), lost data included email addresses, **passwords** and usernames for “a portion of accounts that were

## 11. NetEase

**Date:** October 2015

**Impact:** 235 million user accounts

email addresses and **plaintext passwords** relating to 235 million

## 12. Court Ventures (Experian)

**Date:** October 2013

**Impact:** 200 million personal records

## 13. LinkedIn

**Date:** June 2012

**Impact:** 165 million users

With its second appearance on this list is LinkedIn, this time in reference to a breach it suffered in 2012 when it announced that 6.5 million **unassociated passwords (unsalted SHA-1 hashes)** had been

## 14. Dubsmash

**Date:** December 2018

**Impact:** 162 million user accounts

In December 2018, New York-based video messaging service Dubsmash had 162 million email addresses, usernames, **PBKDF2 password hashes**, and other personal data such as dates of birth

## 15. Adobe

**Date:** October 2013

**Impact:** 153 million user records

more than 150 million username and **hashed password** pairs taken

# Keywords in Problem descriptions

**hashed** passwords

password hashed with weak algorithm **SHA-1**

**plaintext** passwords

**unsalted** SHA-1 hashes

DBKDF2 password hashes

# What is "Salted Hashed Password" ?

A You

explain salted hash passwords



ChatGPT

Salted hash passwords are a method used to store passwords securely in databases. It involves adding a random piece of data, called a "salt", to each password before hashing it. The salt is then stored along with the hashed password in the database.

Here's how the process works:

1. **Generating a Salt:** When a user creates or updates their password, a random salt is generated. This salt should be unique for each user and sufficiently long to ensure its randomness. Commonly, salts are generated using cryptographic random number generators.
2. **Combining Salt and Password:** The salt is then concatenated with the user's password. This ensures that even if two users have the same password, their hashed passwords will be different due to the unique salts.
3. **Hashing:** The combined string (password + salt) is then hashed using a cryptographic hash function, such as SHA-256 or bcrypt. This produces a fixed-size string of characters, known as the hash.
4. **Storing in the Database:** The hashed password and the corresponding salt are stored in the database. The salt is typically stored alongside the hashed password, either in the same database column or in a separate column.

# Outline

Http Authorization

Transport: Need For confidentiality

Backend-end: storing password in Database ?



Cryptographic "Hash" functions, Salt

Who is "John The Ripper" ?

2FA & MFA (Multi Factor Authentication)

# Hash Functions

One-Way ... non invertible

hopefully getting different hashes for different inputs



# Mathematical Properties

hashFunction  $X \rightarrow Y$   
 $x \mapsto y = \text{hash}(x)$

Not invertible:

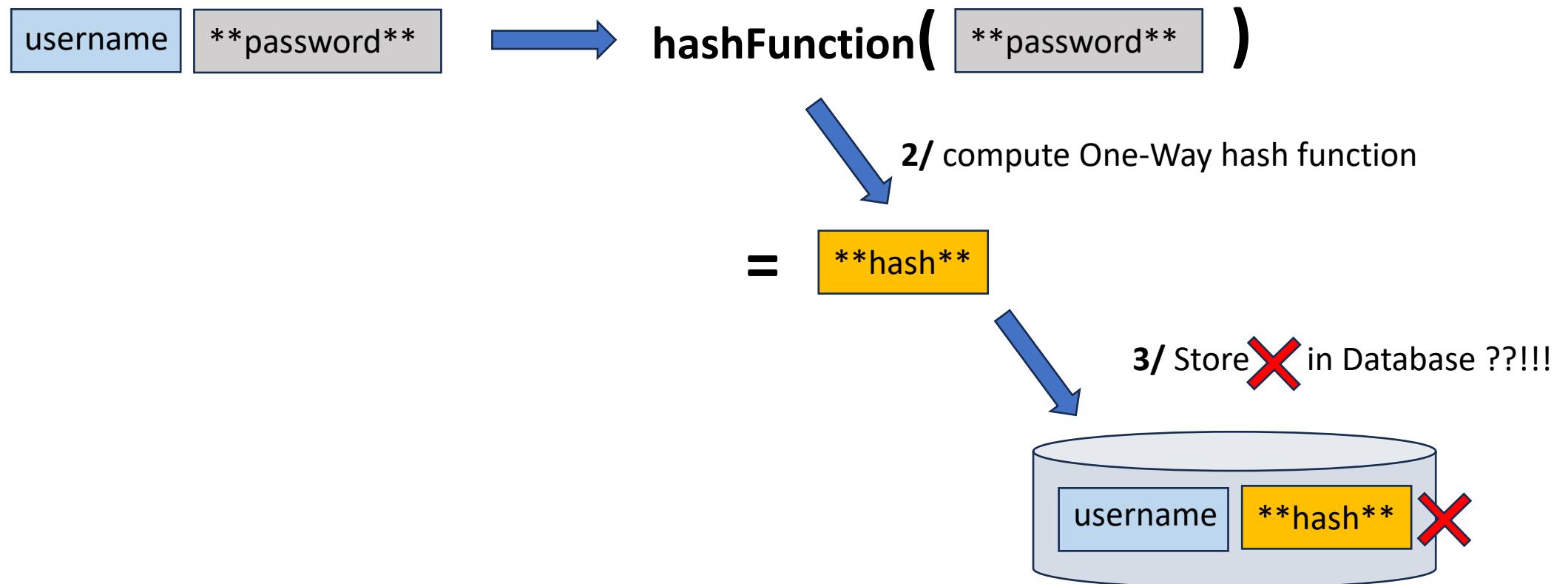
given  $y$  in  $Y$ , it is impossible to simply deduce  $x$  in  $X$ , such that  $f(x)=y$

In general, for  $x \neq y$ , then  $f(x) \neq f(y)$

BUT there can be "rare" collisions, where  $x \neq y$ , and still  $f(x)=f(y)$

In general, if  $f(x) = f(y)$ , then  $x=y$  with very high "probability"

# (UnSalted) Hash Passwords

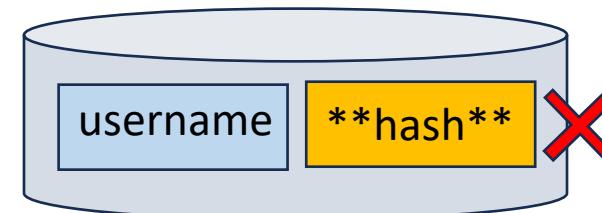
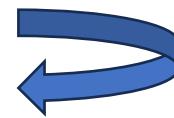


# Checking (Unsalted) HashPassword

username    \*\*check-password\*\*



1/ **find** by username



2/ **compute**    hashFunction(\*\*check-password\*\*) = checkHash



3/ **check equality**    if (\*\*hash\*\* == checkHash) ...



4/ force slowness: sleep **wait**



# Why Adding Salt ?

it tastes different



it keeps longer



# Why Adding Salt ?

same password with different salt are completely different

it is slower to compute hash(password+salt) than just hash(password)  
For every password value, all the computation must be re-evaluated

# $\text{hash}(\text{password+salt})$ OR $\text{hash}(\text{salt+password})$ ?

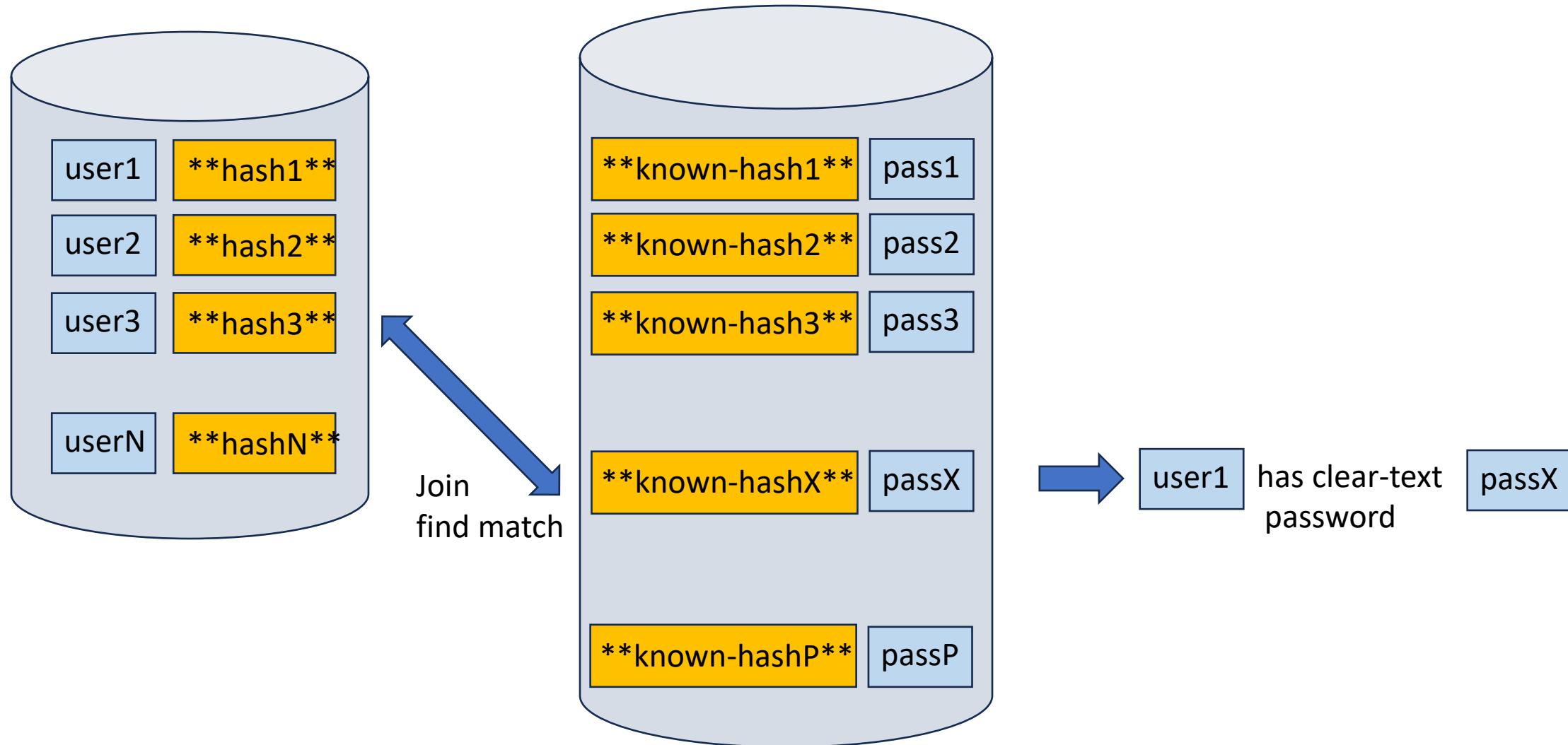
Suppose an attacker may know the resultHash AND the salt

For any password attempt (iterating billions of times),

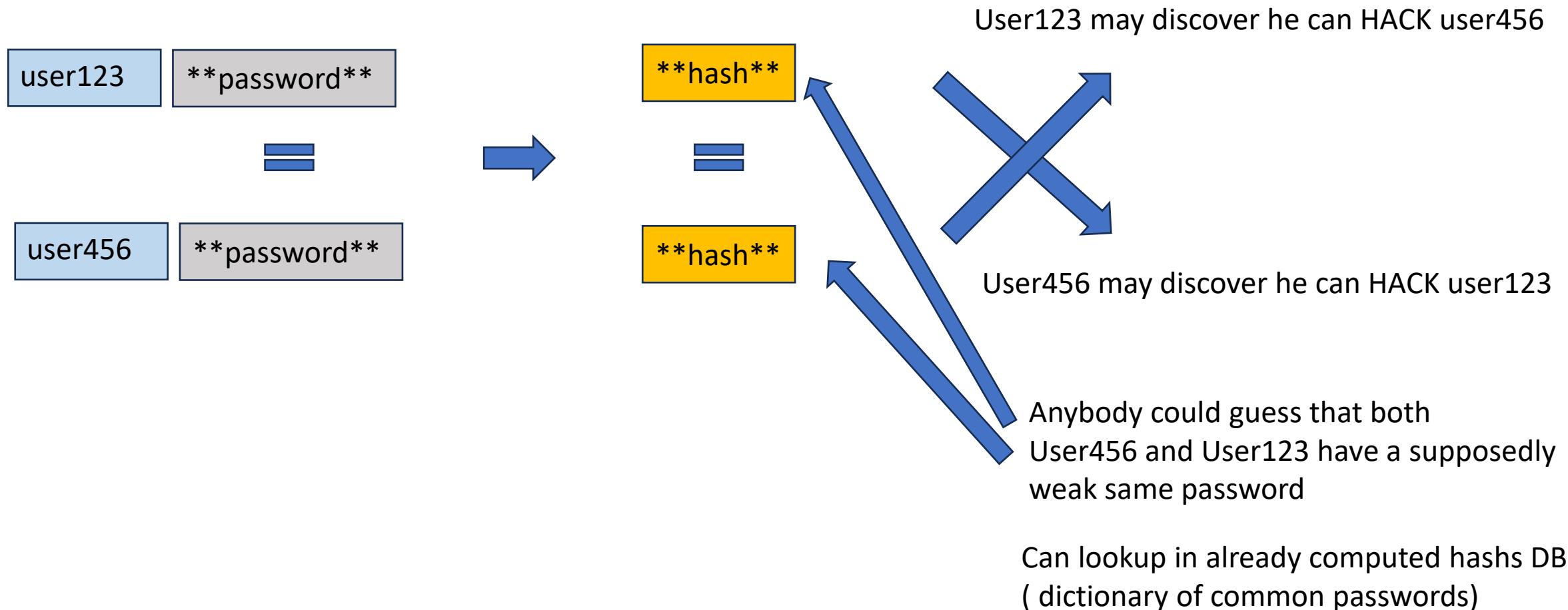
computing **hash(password + salt)** => safer (slower): need to recompute everything

computing **hash(salt + password)** => unsafe (faster): could pre-compute internal state for "hash(salt)"

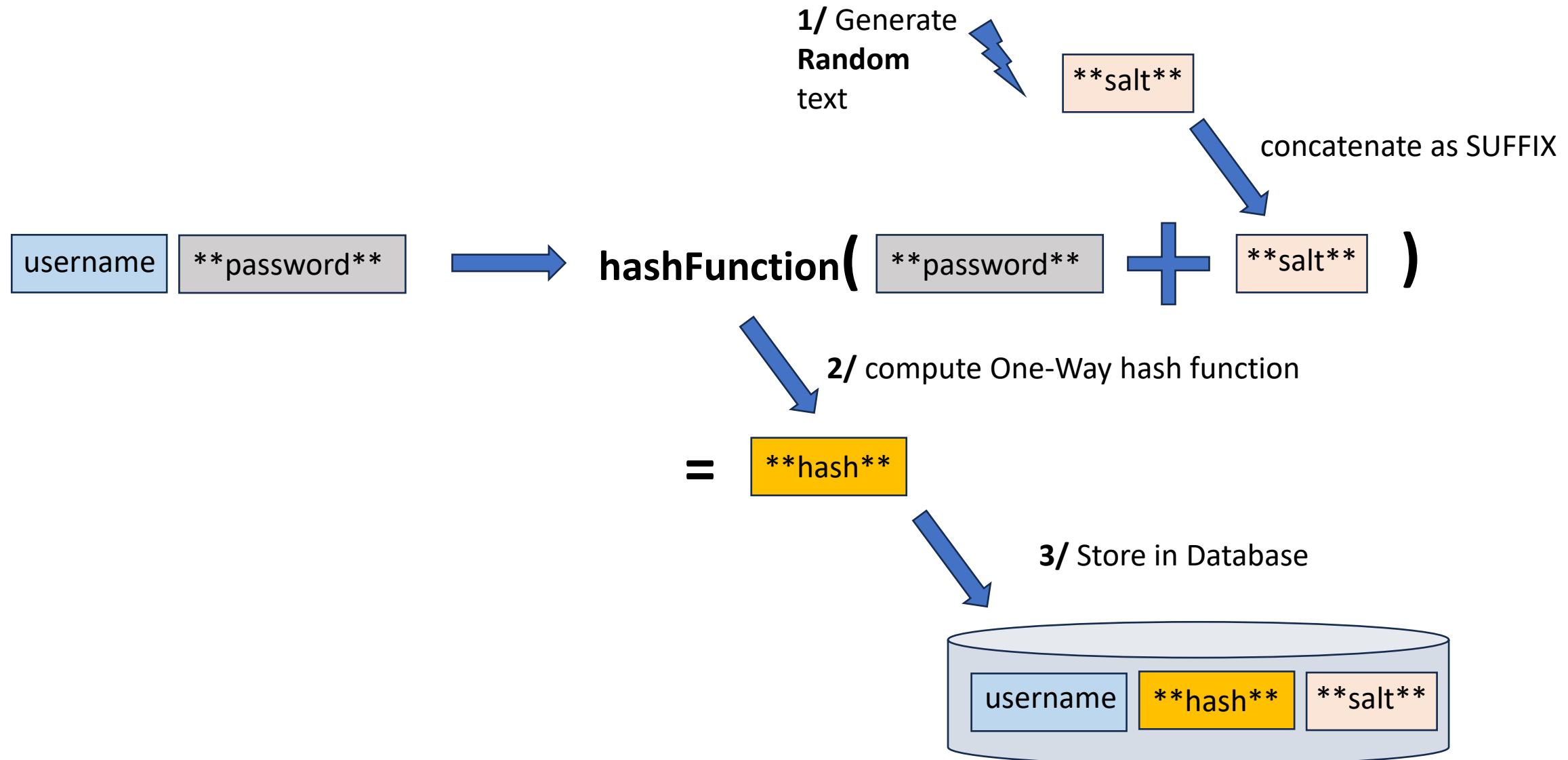
# UnSalted Hashes Database join with Known Common Hashes



# Different Users can have same passwords



# Salted Hash Passwords

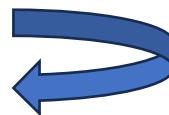


# Checking Password

username    \*\*check-password\*\*



1/ **find** by username



2/ **compute**    `hashFunction(**check-password** + salt) = checkHash`



3/ **check equality**    `if (**hash** == checkHash) ...`



4/ **force slowness: sleep wait**



# \$ sha256sum

```
 Cmder
$ echo "password:salt" | sha256sum
377e6a1f09f801c4c983f6ca95019afbe10eb6ee7d64674dd75d14efdaa0875e *-
```

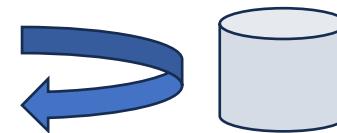
# Checking Hash

```
Cmder
$ saltedHash="377e6a1f09f801c4c983f6ca95019afbe10eb6ee7d64674dd75d14efdaa0875e *-";
$ hashAttempt=$( echo "bad-password-attempt:salt" | sha256sum );
$ if [ "$hashAttempt" = "$saltedHash" ]; then echo "OK"; else echo "FAILED"; fi
FAILED
$
$ hashAttempt=$( echo "password:salt" | sha256sum );
$ if [ "$hashAttempt" = "$saltedHash" ]; then echo "OK"; else echo "FAILED"; fi
OK
$
```

# Can you Spot the Security Problems ?

1/ **find by username**

```
if (notFound) {  
    return false;  
}
```



3/ **check equality**    if (\*\*hash\*\* == checkHash) ...

```
boolean equals(byte[] left, byte[] right) {  
    int len = min(left.length, right.length);  
    for(int i = 0; i < len; i++) {  
        if (left[i] != right[i])  
            return false;  
    }  
    return true;  
}
```

# Notice "Force Slowness"



4/ force slowness: sleep **wait**



## 2 Reasons

1/ block brute force trials

```
if (unsucessfull check) { sleep (5 seconds) }  
if (3rd consecutive attempts) { sleep (30 seconds) }
```

2/ Avoid Side-Channel information

adjust "Time to Find & Compare", so that time does not leak any information  
min time should be always example 5ms +/-0.0001ms

even if User not found,

even if 1st char of hash does not match

# Side-Channel : different response times => avoid optimizations

```
boolean unsafeFastEquals(byte[] left, byte[] right) {  
    int len = min(left.length, right.length);  
    for(int i = 0; i < len; i++) {  
        if (left[i] != right[i]) {  
            return false; // stop loop immediate  
        }  
    }  
    return true;  
}
```

```
boolean cryptographicSlowEquals(byte[] left, byte[] right) {  
    int len = min(left.length, right.length);  
    boolean res = true;  
    for(int i = 0; i < len; i++) {  
        boolean alwaysCheck = (left[i] == right[i]);  
        res &= alwaysCheck;  
    }  
    return res;  
}
```

# Side-Channel : response time

examples:

left = 377e6a1f09f801c4c983f6ca95019afbe10eb6ee7d64674dd75d14efdaa0875e  
right = 1



if optimized,  
only ~2 cpu cycle  
=> ~100 nano-seconds for 2GHz CPU

left = 377e6a1f09f801c4c983f6ca95019afbe10eb6ee7d64674dd75d14efdaa0875e  
right = 377e6a1f09f801c4c983f6ca95019afbe10eb6ee7d64674dd75d14efdaa0875e



30 correct => ~60 cpu cycles => 6000 nanos = 6 micros-sec

# Time Side-Channel attack

Assume already guessed N left chars

Trying to guess the (N+1) char is possible with chrono precision of ~200 nano seconds

Repeat several times trial, to eliminate measurements noise  
by averaging time (and eliminate outliers slows/fast answers)

Trying to guess the (N+1,N+2,N+3,N+4) next chars is possible  
with chrono precision of ~4x200 nano sec  
with 256^4 trials to iterate

# Even Stranger Side-Channels

Suppose you have access to the Hardware

=> Electric consumption (in WATT) depends of executed instructions sequence

=> Thermal Heat (infra-red) depends of Electric Power

**Measuring heat ... leak information to internal values statistics/ordering**

## Side-channel attack

文 A 17 languages ▾

Article Talk

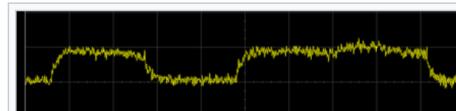
Read Edit View history Tools ▾

From Wikipedia, the free encyclopedia

"Side channels" redirects here. For the yoga concept, see [Nadi \(yoga\)](#) § *Side channels*. For interactive television services, see [Red Button \(digital television\)](#).

In [computer security](#), a **side-channel attack** is any attack based on extra information that can be gathered because of the fundamental way a [computer protocol](#) or [algorithm](#) is [implemented](#), rather than flaws in the design of the protocol or algorithm itself (e.g. flaws found in a [cryptanalysis](#) of a [cryptographic algorithm](#)) or minor, but potentially devastating, [mistakes](#) or [oversights in the implementation](#). (Cryptanalysis also includes searching for side-channel attacks.) Timing information, power consumption, [electromagnetic](#) leaks, and [sound](#) are examples of extra information which could be exploited to facilitate side-channel attacks.

Some side-channel attacks require technical knowledge of the internal operation of the system,



An attempt to decode RSA key bits using [power analysis](#). The left peak represents the CPU power variations during the step of the [algorithm](#) without multiplication, the right (broader) peak – step with multiplication, allowing an attacker to read bits 0, 1.

# Outline

Http Authorization

Transport: Need For confidentiality

Backend-end: storing password in Database ?

Cryptographic "Hash" functions, Salt



**Who is "John The Ripper" ?**

2FA & MFA (Multi Factor Authentication)

# a Password Cracker App

Google search results for "john the ripper":

About 13,200,000 results (0.23 seconds)

**Openwall**  
<https://www.openwall.com/john> ::

**John the Ripper password cracker**

John the Ripper password cracker. John the Ripper is an Open Source password security auditing and password recovery tool available for many operating systems.

Browse the documentation for... · John the Ripper Pro · Pro for Linux · In the cloud

**GitHub**  
<https://github.com/openwall/john> ::

**John the Ripper jumbo - advanced offline password ...**

John the Ripper jumbo - advanced offline password cracker, which supports hundreds of hash and cipher types, and runs on many operating systems, CPUs, GPUs, ...

[README.md](#) · [Issues 513](#) · [Pull requests 2](#) · [Actions](#)



**John the Ripper**  
Software ::

John the Ripper is a free password cracker tool. Originally developed for the Unix operating system, it can run on fifteen different platforms.

The screenshot shows a web browser window with the URL [openwall.com/john/](http://openwall.com/john/) in the address bar. The page header features the Openwall logo and navigation links for Products, Services, Publications, Resources, and What's new. The main content area is titled "John the Ripper password cracker" and describes the tool's capabilities across various platforms and formats.

## John the Ripper password cracker

John the Ripper is an Open Source password security auditing and password recovery tool available for many operating systems. **John the Ripper jumbo** supports hundreds of hash and cipher types, including for: user passwords of Unix flavors (Linux, \*BSD, Solaris, AIX, QNX, etc.), macOS, Windows, "web apps" (e.g., WordPress), groupware (e.g., Notes/Domino), and database servers (SQL, LDAP, etc.); network traffic captures (Windows network authentication, WiFi WPA-PSK, etc.); encrypted private keys (SSH, GnuPG, cryptocurrency wallets, etc.), filesystems and disks (macOS .dmg files and "sparse bundles", Windows BitLocker, etc.), archives (ZIP, RAR, 7z), and document files (PDF, Microsoft Office's, etc.) These are just some of the examples - there are many more.

### [Openwall wordlists collection for password cracking \(20+ languages\)](#)

John the Ripper is free and Open Source software, distributed primarily in source code form. If you would rather use a commercial product, please consider [John the Ripper Pro](#), which is distributed primarily in the form of "native" packages for the target operating systems and in general is meant to be easier to install and use while delivering optimal performance.

Proceed to [John the Ripper Pro](#) homepage for your OS:

- [John the Ripper Pro for Linux](#)
- [John the Ripper Pro for macOS](#)
- **On Windows, consider Hash Suite** (developed by a contributor to John the Ripper)
- On Android, consider [Hash Suite Droid](#)

Get John the Ripper apparel at 0-Day Clothing and support the project

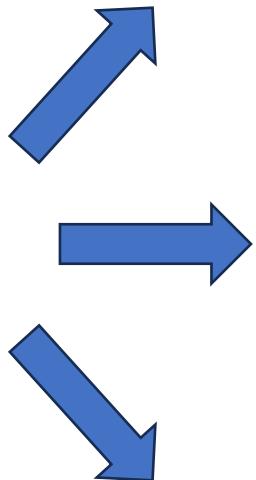
Download the latest John the Ripper jumbo release ([release notes](#)) or development snapshot:

- 1.9.0-jumbo-1 sources in [tar.xz](#), 33 MB ([signature](#)) or [tar.gz](#), 43 MB ([signature](#))
- **1.9.0-jumbo-1 64-bit Windows binaries in [7z](#), 22 MB ([signature](#)) or [zip](#), 63 MB ([signature](#))**
- **1.9.0-jumbo-1 32-bit Windows binaries in [7z](#), 21 MB ([signature](#)) or [zip](#), 61 MB ([signature](#))**
- Development source code in [GitHub repository](#) (download as [tar.gz](#) or [zip](#))



# a Security Auditing and Password Recovery Tool

targeted users :



YOU, ME, everybody SHOULD feel concerned

others ?



# { White | Grey | Black } Hat

## White, gray and black hat comparison



### WHITE HAT

Considered the good guys because they follow the rules when it comes to hacking into systems without permission and obeying responsible disclosure laws



### GRAY HAT

May have good intentions, but might not disclose flaws for immediate fixes

\*\*\*\*\*  
Prioritize their own perception of right versus wrong over what the law might say



### BLACK HAT

Considered cybercriminals; they don't lose sleep over whether or not something is illegal or wrong

\*\*\*\*\*  
Exploit security flaws for personal or political gain—or for fun

# Why me [1/2] ?

Maybe you have an old PC, and forget a password

On day-to-day, you do not use the "Admin" password.

On old PC, you had an old password, now unused.

You manage your family PCs.

# Why me [2/2] ?

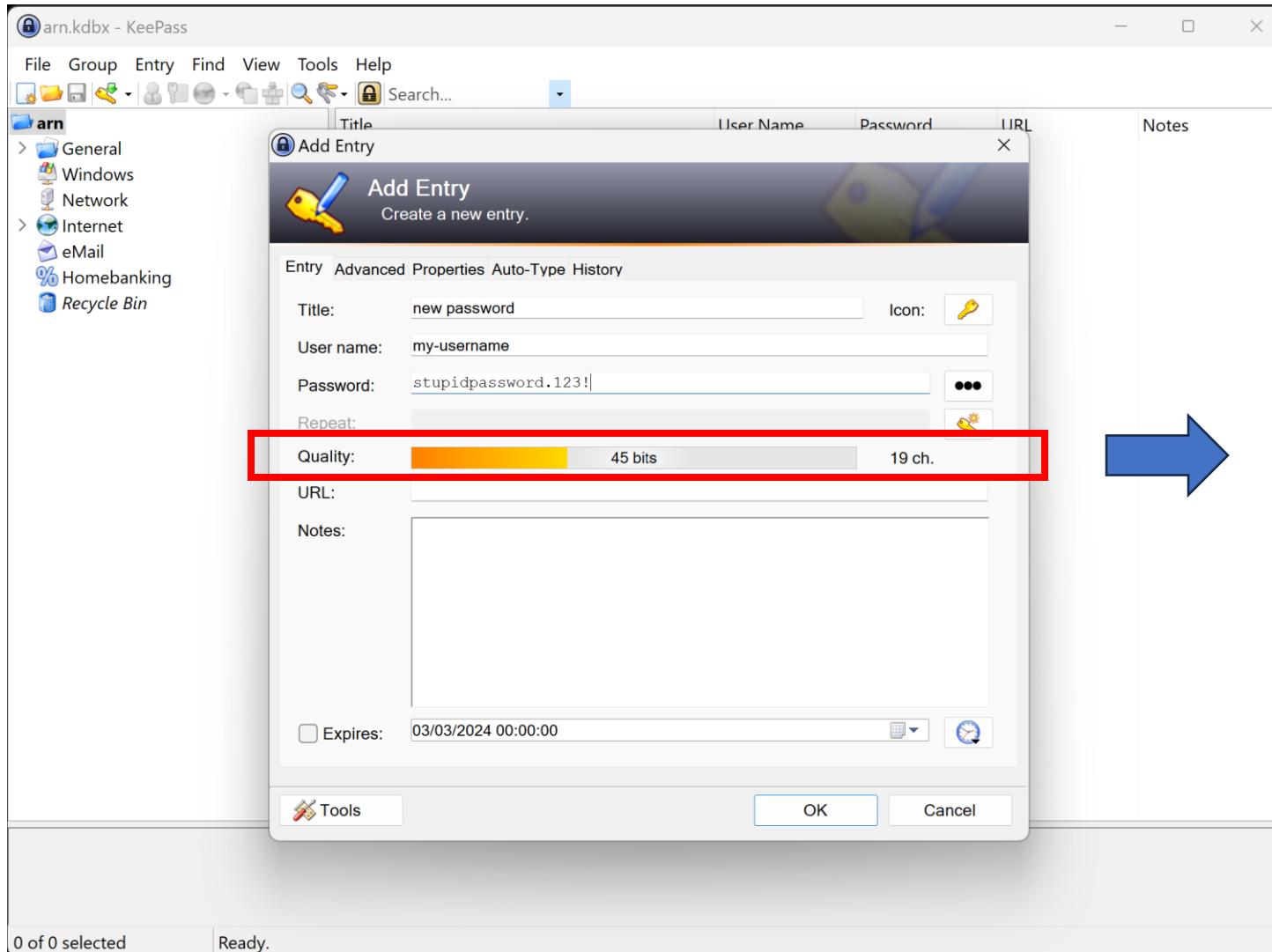
Check your passwords are "resistant" to a simple brute force attack

Without trusting anybody to do it for you

Anyway, hackers will check your passwords some day.

Don't trust "Password Strength" indications

# (Badly Over-)Estimated Password Strength / Quality



Stupid & Small passwords  
have extremely BAD quality

**BUT Estimated High quality does NOT mean OK !**

( You should always GENERATE them )

# Generate Passwords

The screenshot shows the KeePass application interface. The main window title is "arn.kdbx - KeePass". The menu bar includes File, Group, Entry, Find, View, Tools, and Help. The toolbar contains icons for New, Open, Save, Import, Export, Find, and Tools. A search bar is also present.

The main pane displays a list of entries under a group named "arn". One entry titled "Add Entry" is currently selected. A context menu is open over this entry, listing several options for generating passwords:

- Open Password Generator...
- (Derive from previous password)
- (Automatically generated passwords for new entries)
- Hex Key - 40-Bit (built-in) (selected)
- Hex Key - 128-Bit (built-in)
- Hex Key - 256-Bit (built-in)
- Random MAC Address (built-in)

The "Add Entry" dialog box is open, showing the following fields:

- Title: new password
- User name: my-username
- Password: 36a118196ba33e81c238487d5833587e
- Repeat: (empty)
- Quality: 123 bits (32 ch.)
- URL: (empty)
- Notes: (empty)
- Expires: 03/03/2024 00:00:00

At the bottom of the dialog are "Tools", "OK", and "Cancel" buttons.

- John the Ripper Pro for Linux
- John the Ripper Pro for macOS
- On Windows, consider Hash Suite (developed by Openwall)
- On Android, consider Hash Suite Droid

# On Windows, consider HashSuite

hashsuite.openwall.net ★

## Hash Suite

Follow @HashSuite on Twitter for new release announcements

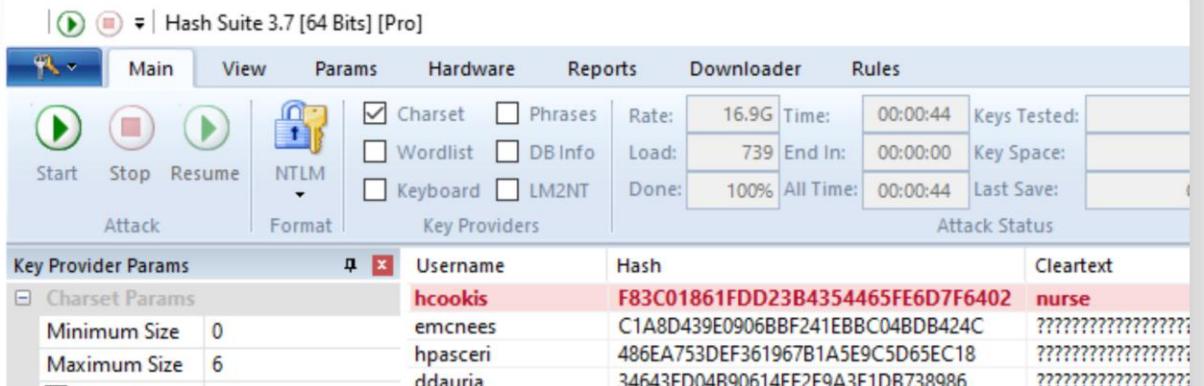
Home FAQ Tutorial Performance Comparison Download Android Changelog License Contact

Fast, powerful, simple

**Home**

Hash Suite is a Windows program to test security of password hashes. Main objectives are:

- **Fast:** We offer a program with very high performance.
- **Simple and modern:** We use a simple GUI with features offered by modern Windows (fig 1).
- **Smart:** Reports with statistics, easy download of quality wordlists, easily fix weak passwords.
- **Powerful:** All common features of modern crackers and many unique to Hash Suite.
- **Scalable:** Works fast even with a huge number of hashes.



**News**

**Hash Suite Droid 1.6**  
October 28, 2021  
Hash Suite Droid 1.6 has just been released, adding support for Android 11 scoped storage enforcement and no longer requiring the READ\_EXTERNAL\_STORAGE and WRITE\_EXTERNAL\_STORAGE permissions.

**Hash Suite 3.7, Droid 1.5.1**  
March 19, 2021  
Hash Suite 3.7 and Hash Suite Droid 1.5.1 are available for download, with SHA256CRYPT and SHA512CRYPT support. Hash Suite Droid 1.5.1 is also on Google Play.

# Example Use-Case

Since 2 years, I always logged-in with my "fingerprint",  
Then with my "PIN" code

... I forgot my old password  
I can not do admin commands anymore



Hash Suite 3.7 [64 Bits] [Free]

Main View Params Hardware Reports Downloader Rules

Import hashes for use in the application.

**Local accounts.**  
Import NTLM/LM hashes from local machine (requires administrator privilege).

**Remote accounts.**  
Import NTLM/LM hashes from remote machine (requires admin privilege on remote machine).

**From Windows Registry.**  
Import Windows hashes (LM/NTLM/DCC/DCC2) from local registry or backup files.

**From Memory.**  
Import already authenticated users (LM/NTLM) from memory.

**From file.**  
Import hashes from a file automatically detecting hash type.

**Wifi captures.**  
Import WPA hashes from a Pcap, Wireshark or Aircrack capture file.

Keys Tested: None  
Key Space: Current  
Last Save: Batch Attacks

Attack Status C...

Options X Exit

First Words Loaded 1000000  
Fingerprint

Phrases Params

Ready. 1 / 1

A screenshot of the Hash Suite 3.7 software interface. The main menu bar includes Main, View, Params, Hardware, Reports, Downloader, and Rules. The 'Main' tab is selected. A dropdown menu under 'Import' is open, showing options like 'Import', 'Export', 'Feedback', 'Wizard', 'Benchmark', 'Hash Calculator', 'Fix Accounts', and 'Delete All Accounts'. The 'From Memory' option is highlighted with a yellow background. The right side of the interface shows fields for 'Keys Tested' (None), 'Key Space' (Current), and 'Last Save' (Batch Attacks). Below these are sections for 'Attack Status' and 'First Words Loaded' (1000000). At the bottom, there's a 'Phrases Params' section and a status bar indicating 'Ready.' and page navigation buttons.

# Imported Password Hashes

The screenshot shows the Hash Suite 3.7 software interface. The title bar reads "Hash Suite 3.7 [64 Bits] [Free]" and the window title is "Importer". The menu bar includes Main, View, Params, Hardware, Reports, Downloader, Rules, and Statistics. The Statistics tab is selected. Below the menu, there are summary statistics: Users Added: 5, Lines Skipped: 0, Completion: 100%, Added: 0, Disable: 5, Exist: 0, Added: 3, Disable: 0, Exist: 2. There are three tabs at the bottom: General, LM, and NTLM. The LM tab is selected. On the left, there is a "Key Provider Params" section with a "Charset Params" panel expanded, showing settings for Minimum Size (0), Maximum Size (6), and checkboxes for Use rules, Lower, Upper, Digit, and Symbol. To the right is a table with columns: Username, Hash, and Cleartext. The table contains five rows of data:

Username	Hash	Cleartext
Administrateur	31:0	??????????????????????...
arnaud	49:6	??????????????????????...
DefaultAccount	31:0	??????????????????????...
Invité	31:0	??????????????????????...
WDAGUtilityAcc...	26:4	??????????????????????...

Other collapsed panels include Wordlist Params, Keyboard Params, Phrases Params, DB Info Params, LM2NT Params, and Rules.

# Maybe "Use Rules" to add known parts Configure known size, wordlist, ..

Restrict searches  
with hypothesis

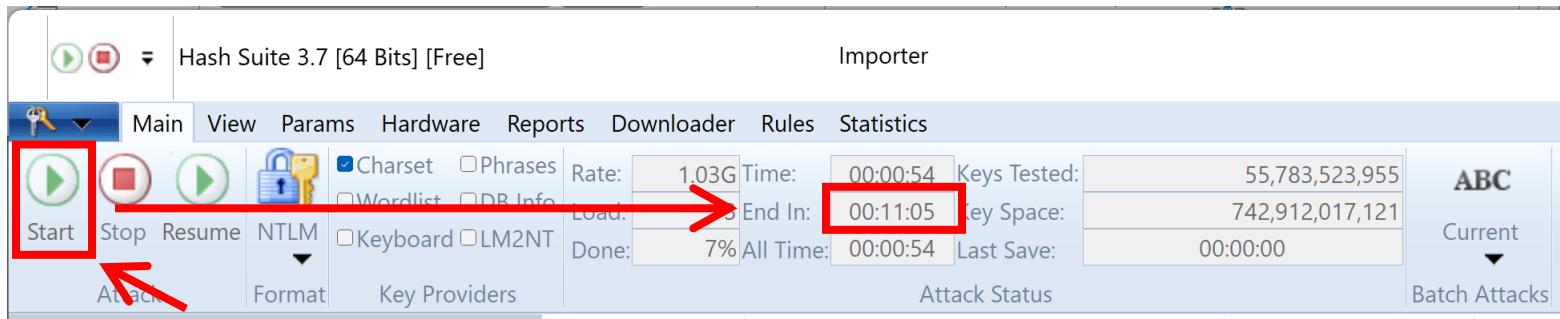
Add known words  
in dictionary

The screenshot shows the Hash Suite 3.7 [64 Bits] [Free] application window. The main interface includes a toolbar with a play/pause button, a progress bar, and a status message 'Hash Suite 3.7 [64 Bits] [Free]'. Below the toolbar is a menu bar with Main, View, Params, Hardware, Reports, Downloader, Rules, and Statistics. The Statistics tab is selected.

On the left, there's a sidebar with sections: General, LM, NTLM, Key Provider Params, Charset Params, Wordlist Params, Keyboard Params, Phrases Params, DB Info Params, LM2NT Params, and Rules. The 'Wordlist Params' section is highlighted with a red box and has a red arrow pointing to its configuration in the main panel.

The main panel displays a table with columns: Username, Hash, and Cleartext. The Hash column for several entries is highlighted in yellow. To the left of the table, under 'Key Provider Params', there are 'Charset Params' settings for Minimum Size (0) and Maximum Size (6), and a checked 'Use rules' checkbox. Below these are checkboxes for Lower, Upper, Digit, and Symbol, each with a corresponding character set example. A red box highlights the 'Wordlist Params' section in the sidebar, which is also shown expanded in the main panel. In the main panel, the 'Wordlist Params' section shows Minimum Size (1), Maximum Size (6), a checked 'Use rules' checkbox, and a dropdown menu for 'Wordlist' containing 'wordlist\_small.lst (21.8 KB)'.

# Sample Compute with assumptions Rules => 11 minutes only



11 minutes is FAST !

it actually took me ~3minutes, knowing "<wordlist>"

on a single, modest PC

... HACKERS don't have assumptions, but they do have databases of wordlist, and CPUs

# Outline

Http Authorization

Transport: Need For confidentiality

Backend-end: storing password in Database ?

Cryptographic "Hash" functions, Salt

Who is "John The Ripper" ?



**2FA & MFA (Multi Factor Authentication)**

# 2FA / MFA = 2/Multi Factor Authentication

A

You

principles of 2FA or MFA security ?

ChatGPT

Two-factor authentication (2FA) and multi-factor authentication (MFA) are both methods used to enhance the security of user accounts by requiring more than one form of verification before granting access. Here are the principles behind these security measures:

## 1. Authentication Factors:

- **Something You Know:** This factor typically involves a password, PIN, or passphrase. It's information that only the user should know.
- **Something You Have:** This factor involves a physical device or token, such as a smartphone, security key, or smart card. It's something the user possesses.
- **Something You Are:** This factor refers to biometric characteristics, such as fingerprints, facial recognition, or iris scans. It's based on the user's unique physical traits.

## 2. Combining Factors:

- 2FA requires two of the above factors for authentication.
- MFA requires two or more of these factors for authentication.

## 3. Defense Against Credential Theft:

- Even if an attacker manages to steal or guess a user's password (the "something you know" factor), they would still need access to the other factor (e.g., the user's smartphone) to successfully authenticate.

## 4. Reducing the Impact of Single Points of Failure:

- If one factor is compromised (e.g., a password is stolen), having an additional factor makes it significantly harder for attackers to gain unauthorized access.

## 5. Adaptability and Scalability:

- MFA solutions can incorporate various authentication methods based on the organization's security requirements and user preferences.
- Methods such as SMS-based codes, authenticator apps (like Google Authenticator), hardware tokens, biometric scanners, and push notifications can be used.

## 6. User Experience and Convenience:

- Balancing security with user convenience is crucial. MFA methods should be user-friendly and not overly burdensome, while still providing effective protection.

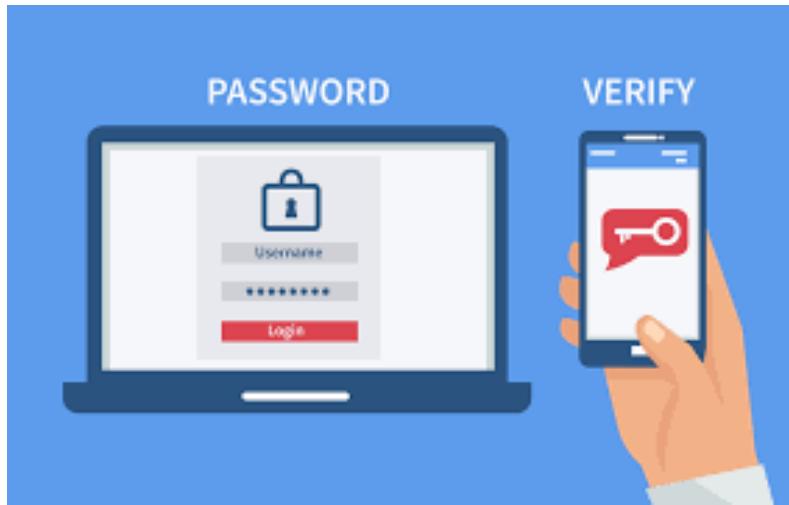
## 7. Continuous Improvement and Risk Assessment:

- Organizations should regularly evaluate their MFA implementations, considering emerging threats, advancements in authentication technology, and user feedback.
- Risk-based authentication can dynamically adjust authentication requirements based on the perceived risk level of the login attempt.

By implementing 2FA or MFA, organizations can significantly enhance the security posture of their systems, protecting against various forms of cyber threats, including password theft, phishing attacks, and credential stuffing.

# Example 2FA

Password + SMS confirmation code



Password + QR code to scan  
with a installed+configured App



Password + Fingerprint

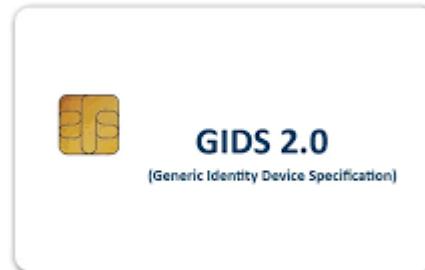


# Example 2FA (or simple Auth when no password)

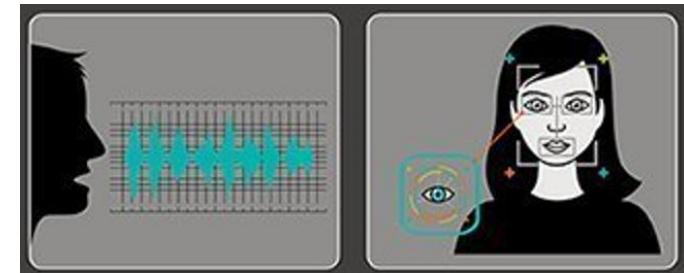
... + RSA generated Ids  
(validity time ~1 minute)



SmartCard + PIN-CODE



!! Voice now possible to break with Generative AI  
(~midjourney, chatgpt)



# Outline

Http Authorization

Transport: Need For confidentiality

Backend-end: storing password in Database ?

Cryptographic "Hash" functions, Salt

Who is "John The Ripper" ?

2FA & MFA (Multi Factor Authentication)

Enough for today



there's more...

# Questions ?

arnaud.nauwynck@gmail.com