



Les mots de passe

Franck LAMY – BTS SIO1

► **Partie théorique** ◀





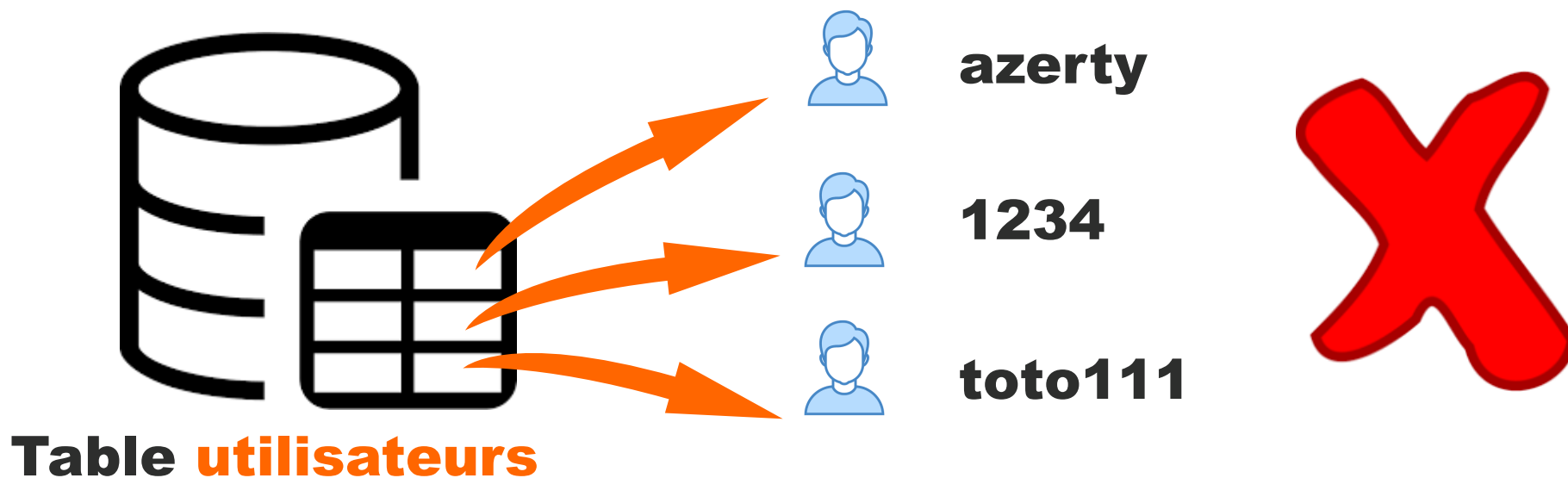
Gestion des utilisateurs



**comment stockés les
mots de passe**



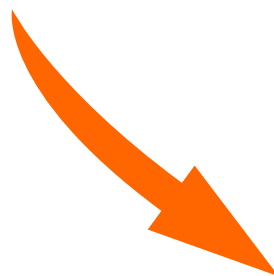
Mots de passe en clair



► **Les mots de passe
ne doivent jamais
être stockés en
clair dans la base
de données !** ◀



► **Pour des besoins de** ◀
sécurité, les mots
de passe doivent
être hachés



**Fonction de
hachage**

► **Une fonction de hachage** ◀
est un algorithme
permettant, à partir d'un
message, de générer une
valeur de longueur fixe
appelée hash

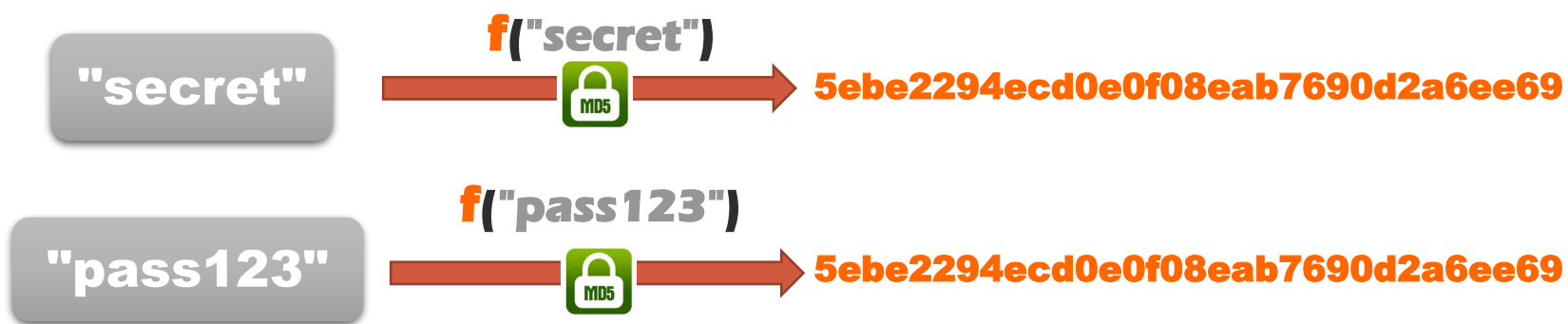


► **Une fonction de hachage** ◀
est irréversible



Obsolètes pour les mots de passe

► Dans la théorie, 2 messages différents ne doivent pas avoir le même hash ◀



Collision

► **Une bonne fonction de hachage ne doit pas permettre les collisions** ◀



► Problème n°1 : le même hash ◀



johndoe@symfony.com

secret ->md5(secret)=

5ebe2294ecd0e0f08eab7690d2a6ee69



janedoe@symfony.com

secret ->md5(secret)=

5ebe2294ecd0e0f08eab7690d2a6ee69



**Le hacker sait
donc que johndoe
et janedoe ont le
même mot de
passe**

► Problème n°2 : mots de passe courants ◀



john.doe@symfony.com

secret ->md5(secret)=

5ebe2294ecd0e0f08eab7690d2a6ee69



janedoe@symfony.com

pass123 ->md5(secret)=

32250170a0dca92d53ec9624f336ca24



Le hacker va utiliser
des bases de
données contenant
les hash des mots
de passe les plus
courants

► Utilisation du **sel cryptographique** (salt) afin de générer un **hash** qui soit **difficile à cracker** ◀



**Valeur
aléatoire**



johndoe@symfony.com
password -> secret
sel -> 098Dy



md5(secret + sel)
559426a2960c56a69ceacd4be394049a



janedoe@symfony.com
password -> secret
sel -> 943FB



md5(secret + sel)
1004c5d097f2bf27eefcbd237ce96f10



Casse-tête pour notre ami !

▶ **La plupart des algorithmes de hachage modernes génèrent un sel aléatoirement et le stocke dans le hash** ◀



Partie pratique en PHP



PRAT!QUE



Fonction permettant de **hacher** un **mot de passe**

▶ **password_hash(...)** ◀

**Plusieurs algos
de hachage
possible**



Hacher un mot de passe

```
$password = "secret";  
$hash = password_hash($password, PASSWORD_DEFAULT);  
echo $hash;
```

\$2y\$10\$vxni3l.jMgDPBpCVXeJ7POJIOGllqPXV6pRXQzJc/FPVZ6HpO.EbG

```
$password = "secret";  
$hash = password_hash($password, PASSWORD_ARGON2I);  
echo $hash;
```

\$argon2i\$v=19\$m=65536,t=4,p=1\$eGFEUFRMLzl3Qlpma3hLcw\$r/jvge
l5hzaSRzUoJLsQlqj3ftgwBZsGu4ebrGk6Qlk



Fonction permettant de **vérifier** un **mot de passe**

▶ **password_verify(...)** ◀

Vérifier un mot de passe

```
$password = "secret";  
$hash = password_hash($password, PASSWORD_ARGON2I);  
  
if (password_verify($password, $hash)) {  
    echo 'Le mot de passe est valide !';  
} else {  
    echo 'Le mot de passe est invalide !';  
}
```

IRREVERSIBLE



▶ Principe ◀

```
password_verify($password, $hash)
```

1
Extrait le sel de **\$hash**

2
Hachage de **\$password** avec le même sel

3
Compare les 2 hash