

06 Cryptographie

Les mots de passes

Corinne HENIN
www.arsouyes.org

Quel est le problème

Accès frauduleux à la BDD

Aucun système n'est sûr

Mauvaise conf, vulnérabilité, insider, ...

Défense en profondeur

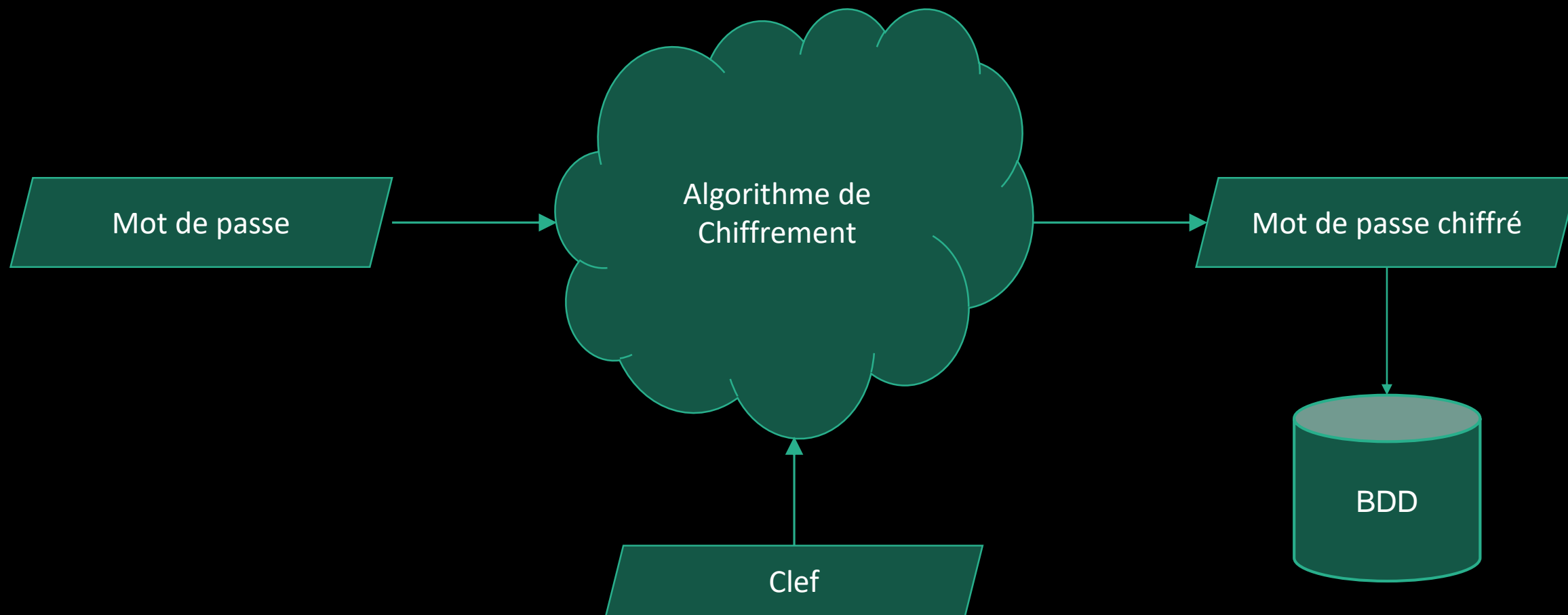
Défendre à tous les niveaux

Protéger le contenu de la BDD

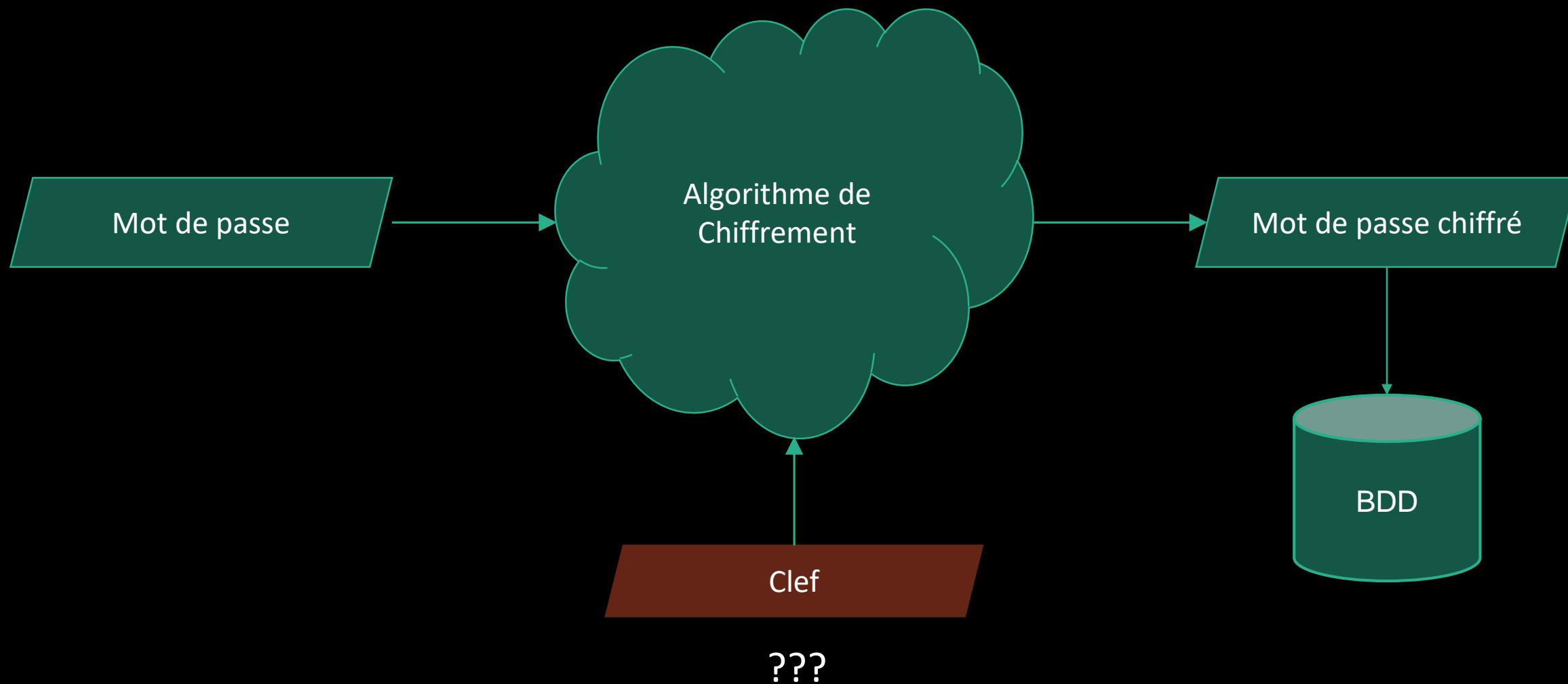
Intuitivement

Solution un peu lourde

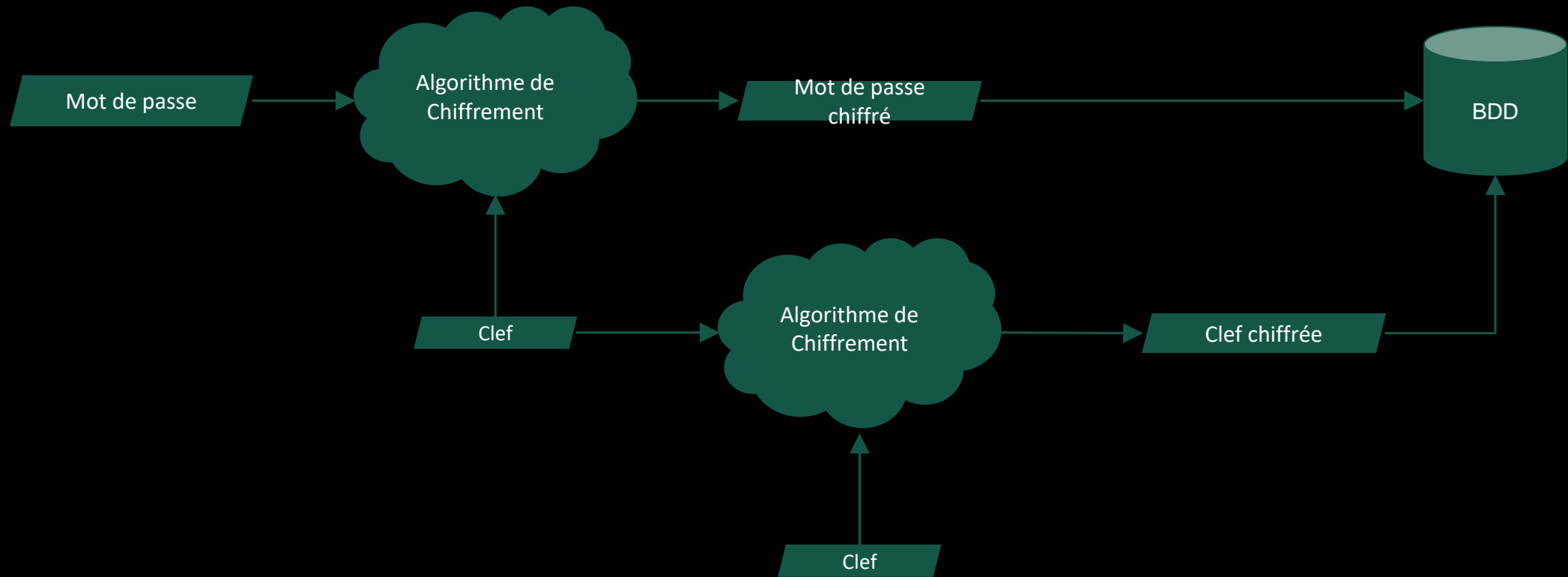
Chiffrer



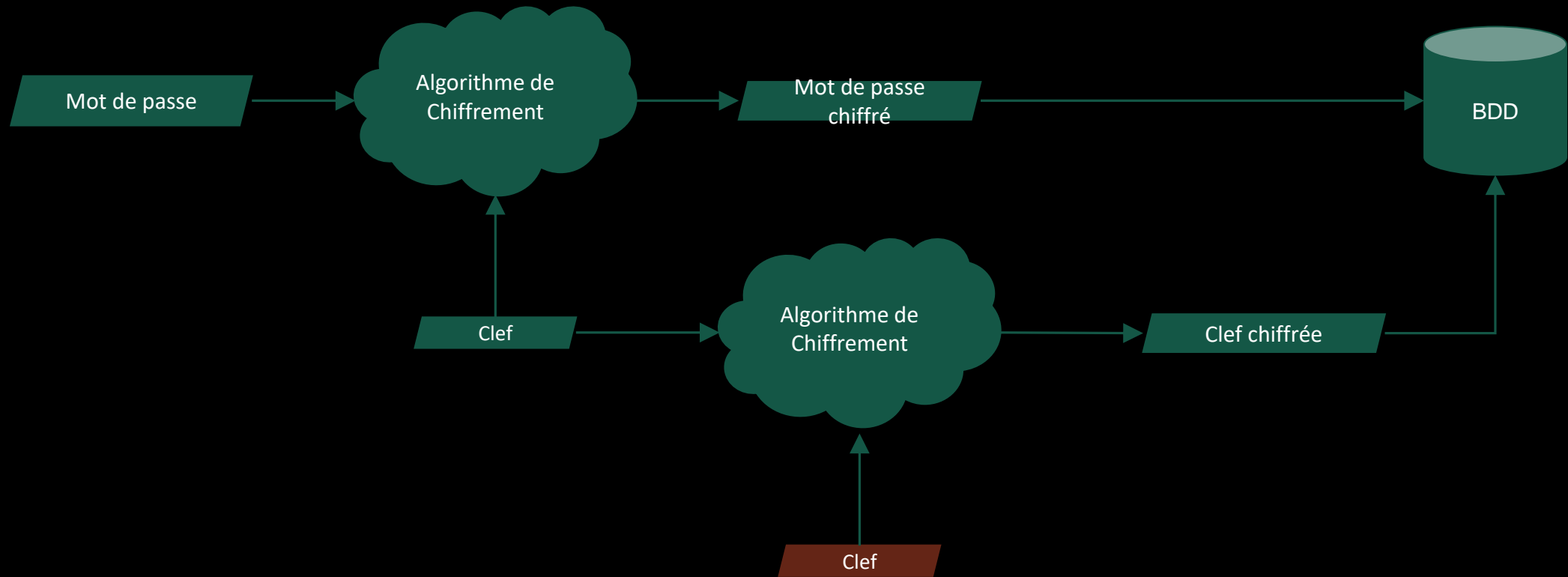
Chiffrer



Chiffrer



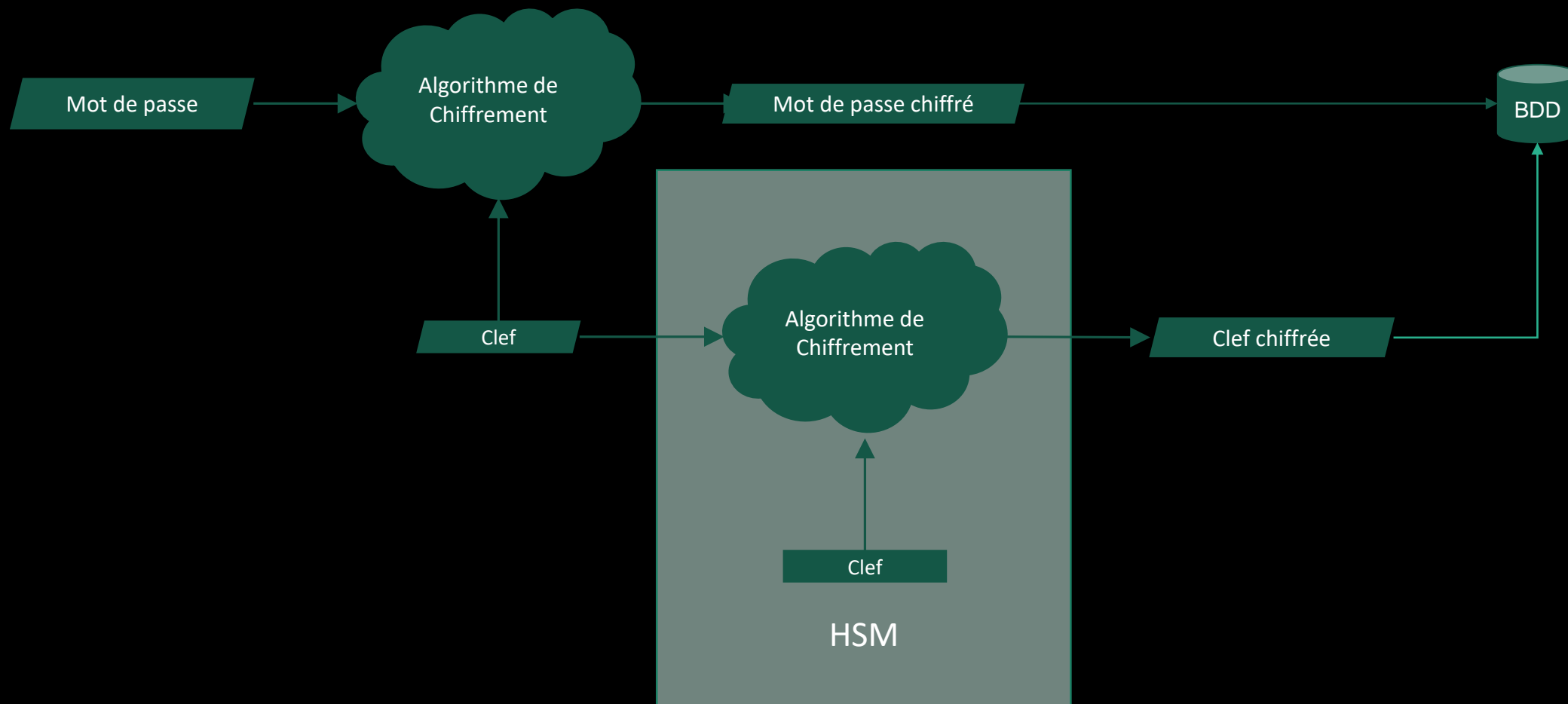
Chiffrer



Chiffrer



Chiffrer



Limitations

C'est lourd

Code plus volumineux, achat de matériel

C'est complexe

Changement de clefs

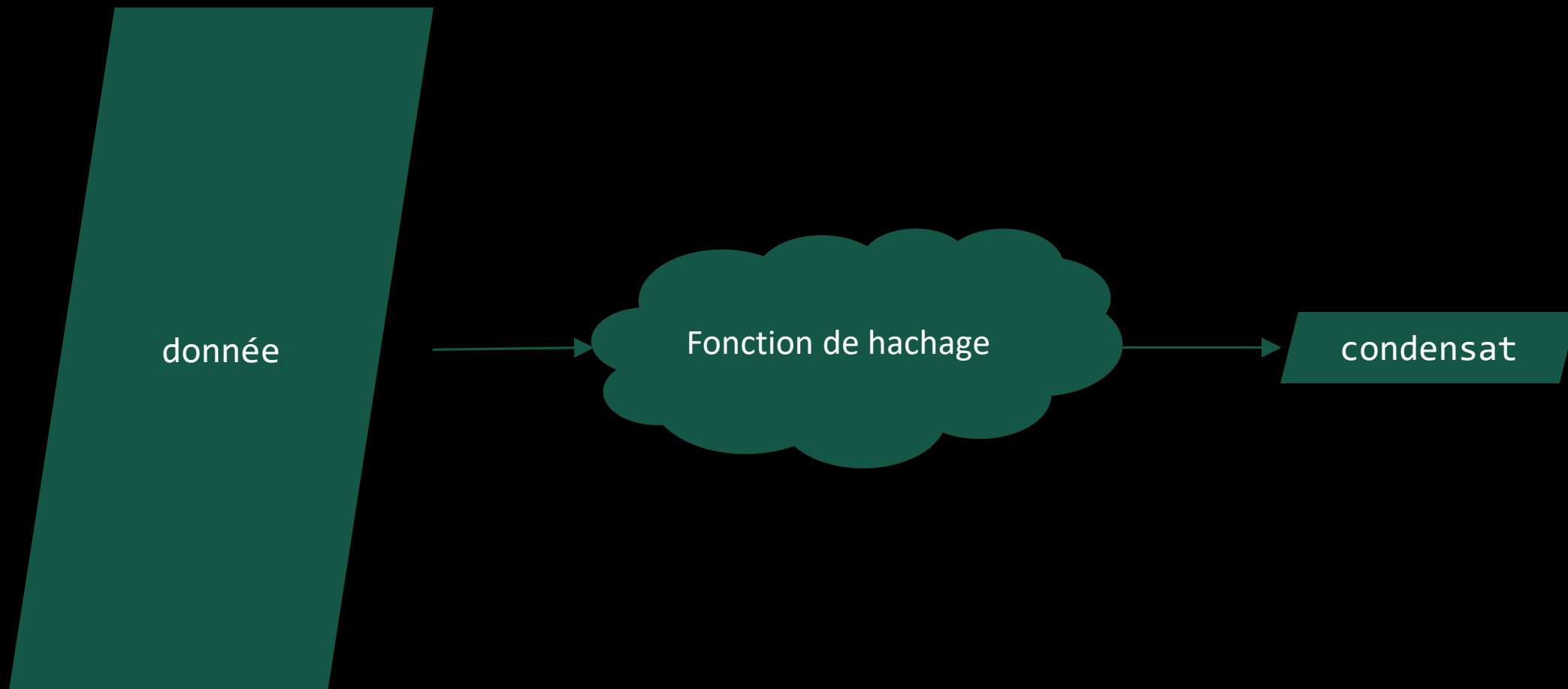
Sauvegarde obligatoire

Destruction des données si ouverture

Penser autrement

Autre solution ?!

Fonction de hachage



Critères de sécurité

Irréversible

ne pas trouver la donnée

Infalsifiable

ne pas trouver une autre donnée

Résistante aux collisions

Sert aux preuves formelles

Exemples de fonctions publiées

Obsolètes

MD5

(depuis 1993)

SHA1

(depuis 2012)

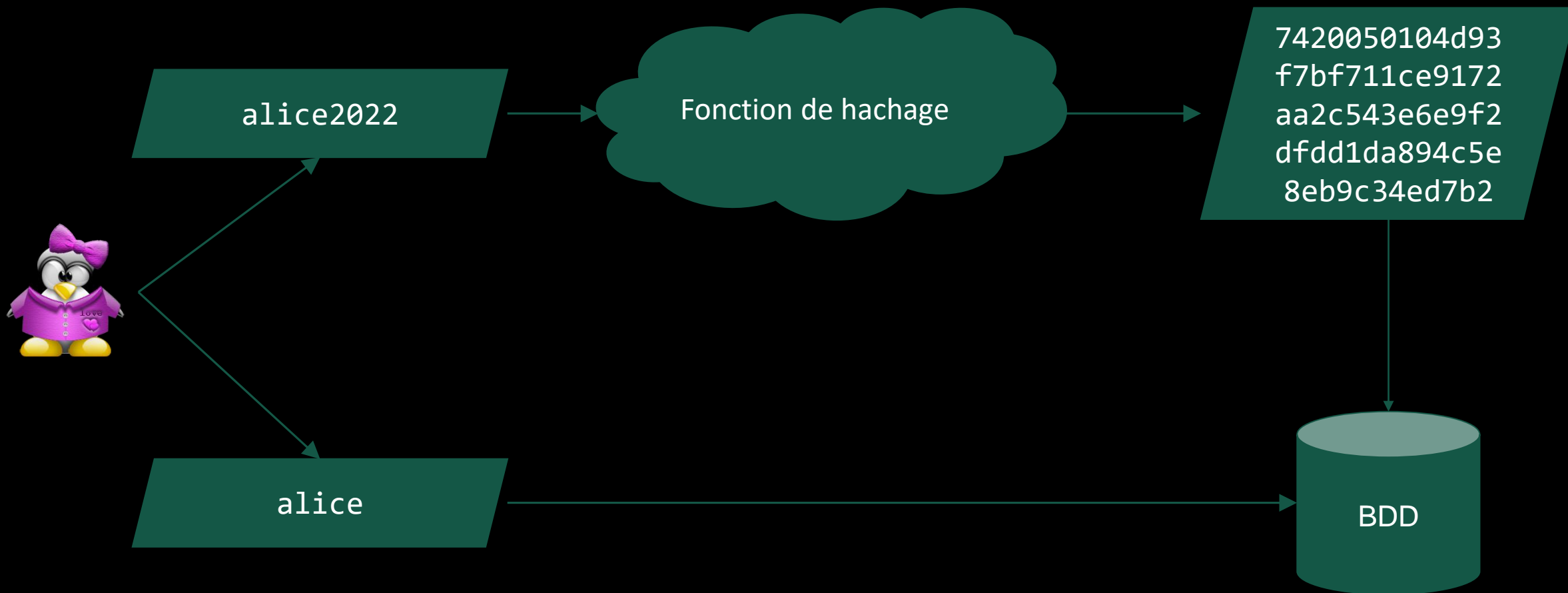
A jour

SHA2

(SHA256 et SHA512)

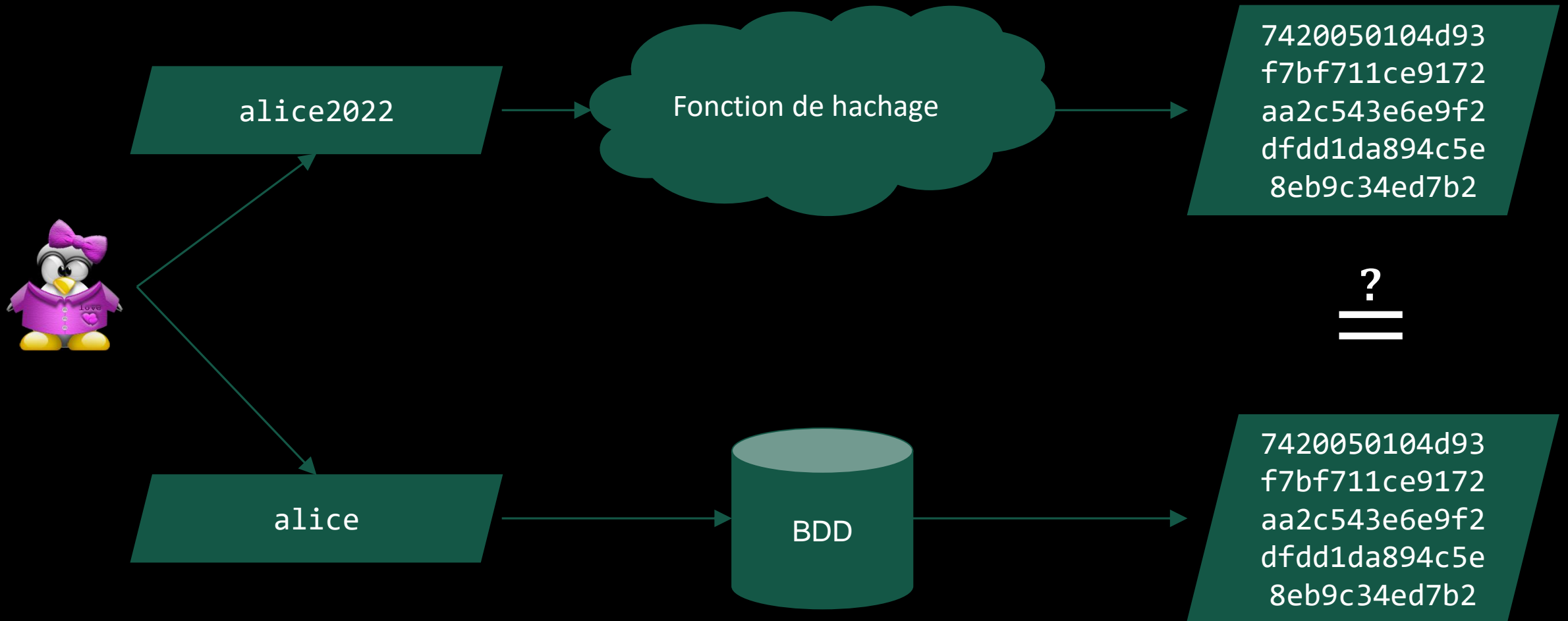
Nouvel utilisateur

Stockage du mot de passe



Authentification

Vérification du mot de passe



Php – Stockage

```
function store($password) {  
    return hash("sha512", $password) ;  
}
```

```
function verify($password, $stored) {  
    return hash_equals(store($password), $stored) ;  
}
```

Php – Vérification

```
function store($password) {  
    return hash("sha512", $password) ;  
}
```

```
function verify($password, $stored) {  
    return hash_equals(store($password), $stored) ;  
}
```

Limitations

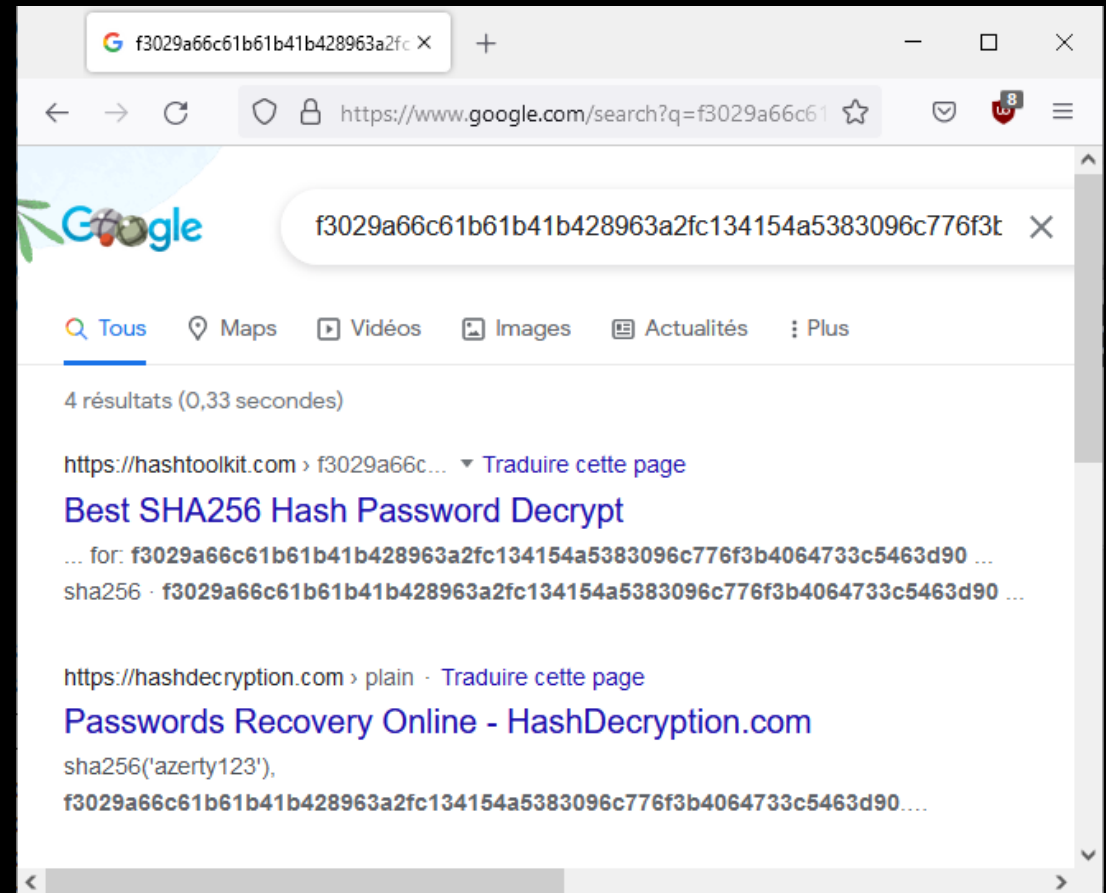
Hachage déterministe

Dictionnaire

(pré-calcul)

Rainbowtable

(dictionnaire compressé)



Ajouter du sel

Pour ajouter du piment

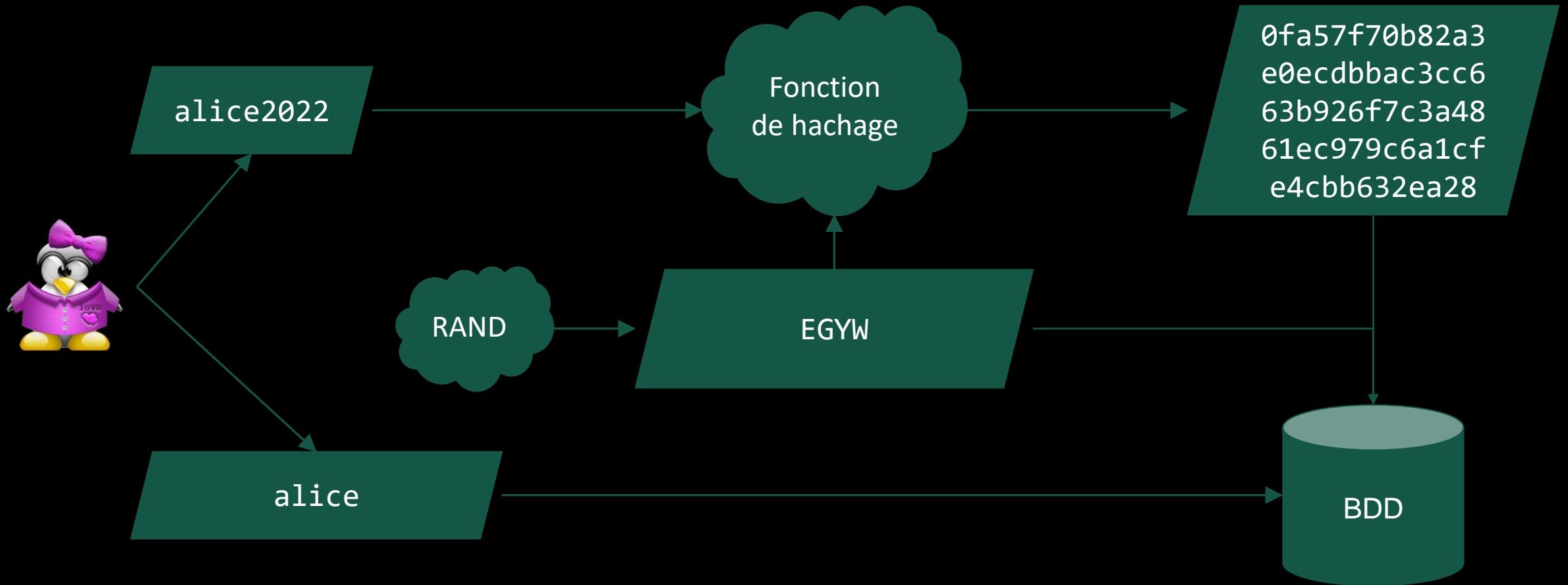
Salage

Augmenter artificiellement la taille

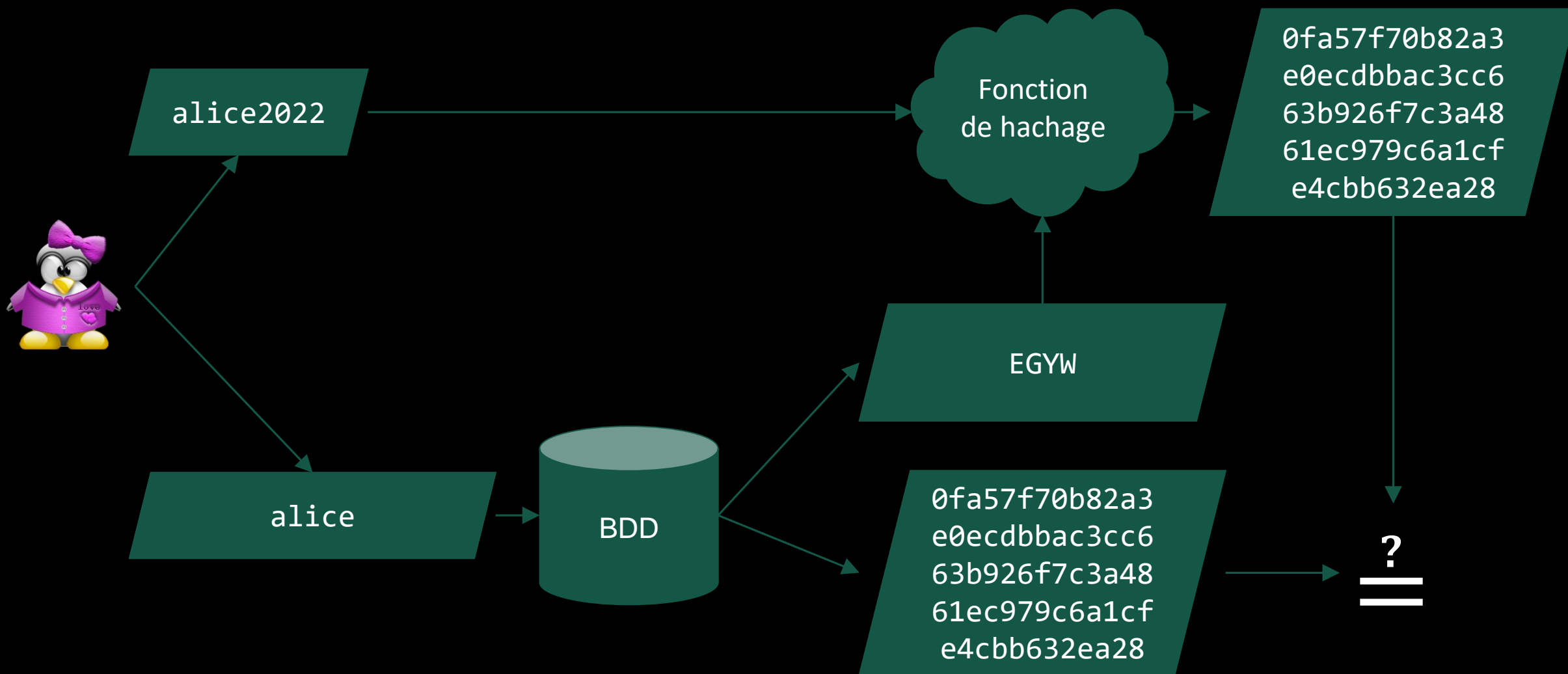
Ajout de caractères aléatoires

Sel stocké dans la BDD

Stockage du mot de passe



Vérification d'un mot de passe



Php – Stockage

```
function store($password) {
    $salt = bin2hex(random_bytes(16)) ; // 128 bits
    $hash = hash("sha512", $password . $salt) ;
    return "$salt.$hash" ;
}

function verify($password, $stored) {
    list($salt, $hash) = explode(".", $stored) ;
    return hash_equals(hash("sha512", $password . $salt), $hash) ;
}
```

Php – Vérification

```
function store($password) {
    $salt = bin2hex(random_bytes(16)) ; // 128 bits
    $hash = hash("sha512", $password . $salt) ;
    return "$salt.$hash" ;
}

function verify($password, $stored) {
    list($salt, $hash) = explode(".", $stored) ;
    return hash_equals(hash("sha512", $password . $salt), $hash) ;
}
```

Avantage

Bruteforce nécessaire

(dictionnaires plus stockables)

Limitations

GPU efficaces
(i.e. Hashcat efficace)

```
Invite de commandes

83ceaf3a99d23039130abe304a9d4b30b6ad89d656414712badab90b45260246505f9b78155b4cb91
e31f2b63a9d331ec97f4d6e0809b1730cc4028a3bce668b.60985c1aeb17dba7674691475faa834a.
aze123

Session.....: hashcat
Status.....: Cracked
Hash.Type.....: sha512($pass.$salt)
Hash.Target.....: 83ceaf3a99d23039130abe304a9d4b30b6ad89d656414712bad...aa834a
Time.Started.....: Wed Dec 18 18:00:39 2019 (2 secs)
Time.Estimated...: Wed Dec 18 18:00:41 2019 (0 secs)
Guess.Mask.....: ?1?1?1?1?1?1 [6]
Guess.Charset....: -1 ?l?u?d, -2 Undefined, -3 Undefined, -4 Undefined
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 36867.5 kH/s (84.49ms) @ Accel:64 Loops:32 Thr:256 Vec:1
Recovered.....: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.....: 94371840/56800235584 (0.17%)
Rejected.....: 0/94371840 (0.00%)
Restore.Point....: 0/14776336 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:928-960 Iteration:0-32
Candidates.#1...: UBnier -> ybz8he
Hardware.Mon.#1..: Temp: 42c Fan: 46% Util: 99% Core:1290MHz Mem:3504MHz Bus:16

Started: Wed Dec 18 18:00:20 2019
Stopped: Wed Dec 18 18:00:42 2019

C:\Users\corin\Downloads\hashcat-5.1.0>
```

Ralentir

Pour ne plus afficher trop vite

Principe

Augmenter le temps

Répétition des opérations

Augmenter les ressources

Vecteur d'initialisation _très_ grand

Parallélisme

Fonctions adaptatives

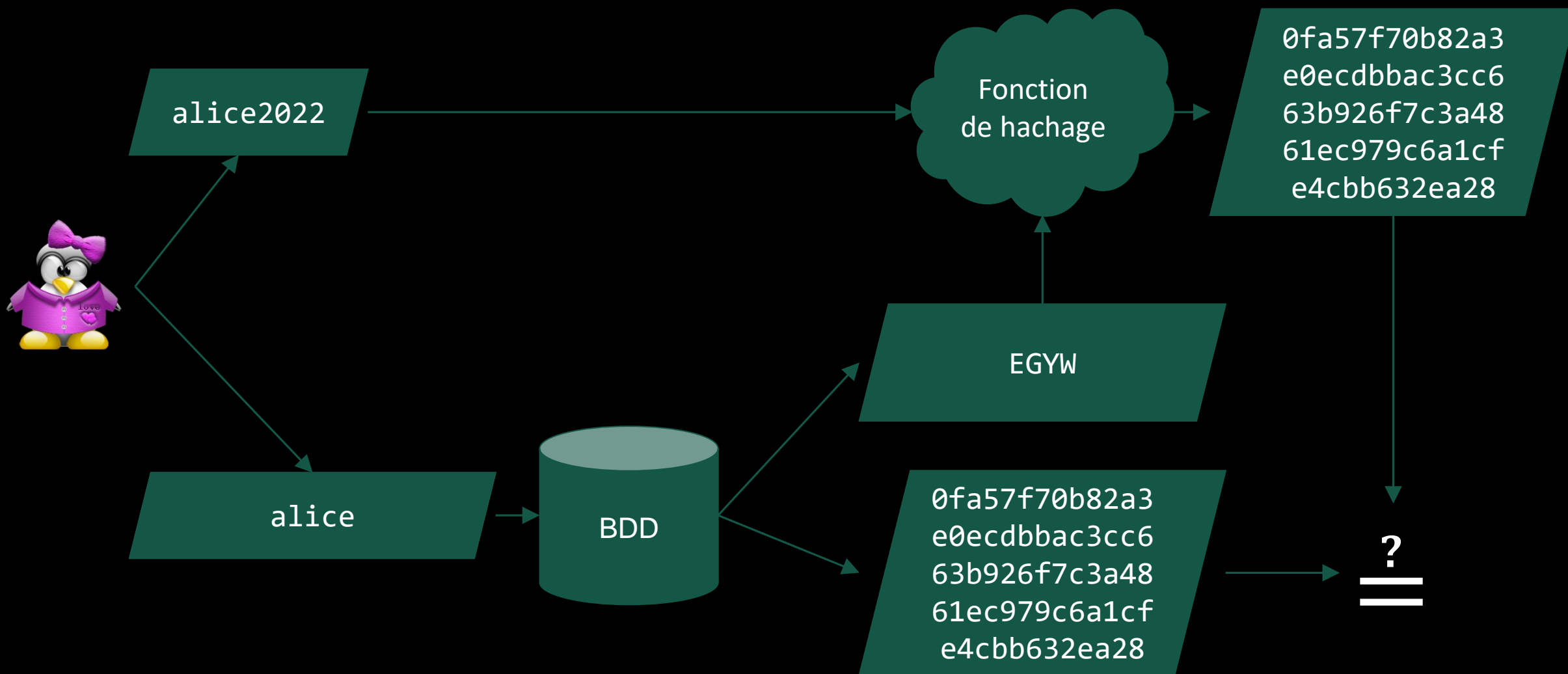
Bcrypt (1999)

Scrypt (2012)

Argon2 (2015)

PBKDF2 (2015)

Vérification d'un mot de passe



Php – Hachage

```
function myHash($password, $salt, $cost) {
    $options = sprintf('$2a$%\'.02d$%\'.22s$', $cost, $salt) ;
    return crypt($password, $options) ;
}

function store($password) {
    $salt = bin2hex(random_bytes(16)) ;
    $hash = myHash($password, $salt, 10) ;
    return $hash ;
}

function verify($password, $stored) {
    $hash = crypt($password, $stored) ;
    return hash_equals($stored, $hash) ;
}
```

Php – Stockage

```
function myHash($password, $salt, $cost) {
    $options = sprintf('$2a$%\'.02d$%\'.22s$', $cost, $salt) ;
    return crypt($password, $options) ;
}

function store($password) {
    $salt = bin2hex(random_bytes(16)) ;
    $hash = myHash($password, $salt, 10) ;
    return $hash ;
}

function verify($password, $stored) {
    $hash = crypt($password, $stored) ;
    return hash_equals($stored, $hash) ;
}
```

Php – Vérification

```
function myHash($password, $salt, $cost) {
    $options = sprintf('$2a$%\'.02d$%\'.22s$', $cost, $salt) ;
    return crypt($password, $options) ;
}

function store($password) {
    $salt = bin2hex(random_bytes(16)) ;
    $hash = myHash($password, $salt, 10) ;
    return $hash ;
}

function verify($password, $stored) {
    $hash = crypt($password, $stored) ;
    return hash_equals($stored, $hash) ;
}
```

Ne pas réinventer la roue

D'autres y ont déjà pensé pour vous

Php – Stockage

```
function store($password) {  
    return password_hash($password, PASSWORD_DEFAULT) ;  
}
```

```
function verify($password, $stored) {  
    return password_verify($password, $stored) ;  
}
```

Php – Stockage

```
function store($password) {  
    return password_hash($password, PASSWORD_DEFAULT) ;  
}
```

```
function verify($password, $stored) {  
    return password_verify($password, $stored) ;  
}
```