SAÉ 3.01	Version : 4.0
Document : Dossier de tests	Date : 16/02/2024
Responsables de la rédaction :	Pierre JAUFFRES, Cyril TILAN

Dossier de tests des fonctions de l'application

1. Introduction

On a réalisé un dossier de tests pour chacune des fonctions des fichiers *Crypto.php* et *functions.php*. Ce dossier de test comporte 6 tableaux, un pour chacune des fonctions testées

2. Description de la procédure de test

Pour la campagne de tests, nous avons utilisé des tests unitaires en boîte blanche. Les tests unitaires en boîte blanche impliquent la création de tests basés sur la structure interne du code source. Les développeurs écrivent des tests qui visent à couvrir toutes les branches de décision, les boucles et les chemins logiques du code. Ensuite, ces tests sont exécutés sur le code source pour s'assurer qu'il fonctionne correctement selon sa structure interne.

3. Description des informations à enregistrer pour les tests

1. Campagne de test

a. Test de Crypto.php

Produit testé : Crypto.php	
Configuration logicielle : Windows 10 ou 11 ; PhpStorm avec PHP 5.6	
Configuration matérielle : PC sous Windows	
Date de début : 16/02/2024	Date de finalisation :
Tests à appliquer : test sur les fonctions cryptographiques du fichier *Crypto.php*	
Responsable de la campagne de test Pierre JAUFFRES	

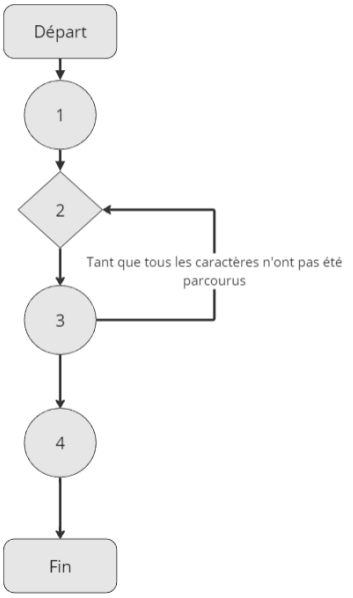
Identification du test :	chiffrement_RC4_key	Version : 1
Description du test : chiffrement_RC4_key est une fonction de hachage cryptographique qui prend en paramètre un message, une clé et renvoie une version chiffrée du message		
Ressources requises : PhpStorm et le module PHPUnit		
Responsable : Pierre JAUFFRES		

fonction testée :

```php
function chiffrement_RC4_key($password,$key) {
    #1
   $key_bytes = $key;
   echo "Clé utilisée : ". $key_bytes ."\n";
   $password_bytes = $password;
   $S = KSA($key_bytes);
   $keystream = PRGA($S, strlen($password_bytes));
   $encrypted_password = '';

   foreach (str_split($password_bytes) as $index => $char) { #2
       #3
       $encrypted_password .= sprintf('%02x', ord($char) ^ $keystream[$index]);
   }

   return $encrypted_password; #4 // Retourne le texte chiffré
}
```



Lors de l'étape 1 on utilise la fonction *KSA* et *PRGA* qui seront traitées dans la suite du dossier de tests.

On remarque que cette fonction n'a qu'un seul chemin possible qui est C1 : {1;2;3;4}. Dans ce cas pour vérifier le fonctionnement de la fonction *chiffrement_RC4_key*, on a juste besoin de réaliser un seul test.

Référence du test appliqué : chiffrement_RC4_key
Responsable :
Date de réalisation du test :
Résultat du test : (OK, KO, non fait, dérogé)

Chemin	$password	$key	Résultat attendu	Résultat du test
C1	pedia	wiki	34a8a217e4	OK

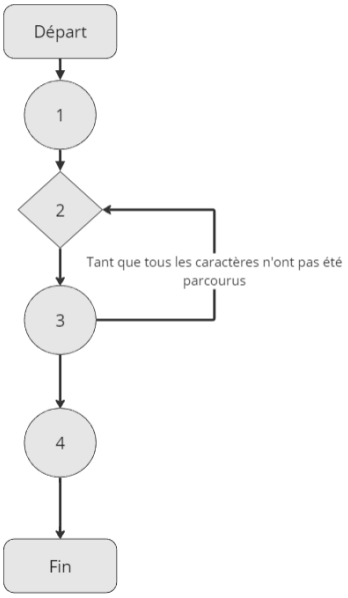
Identification du test :	dechiffrement_RC4	Version : 1
Description du test : dechiffrement_RC4 est une fonction de hachage cryptographique qui prend en paramètre un message chiffré, une clé et renvoie une version déchiffrée du message		
Ressources requises : PhpStorm et le module PHPUnit		
Responsable : Pierre JAUFFRES		

fonction testée :

```php
function dechiffrement_RC4($encrypted_password, $key) {
    #1
   $key_bytes = $key;
   $S = KSA($key_bytes);
   $keystream = PRGA($S, strlen($encrypted_password) / 2);
   $encrypted_bytes = pack('H*', $encrypted_password);
   $decrypted_password = '';

   foreach (str_split($encrypted_bytes) as $index => $byte) { #2
        #3
       $decrypted_password .= chr(ord($byte) ^ $keystream[$index]);
   }

   return $decrypted_password; #4 // Retourne le texte déchiffré
}
```



Lors de l'étape 1 on utilise la fonction *KSA* et *PRGA* qui seront traitées dans la suite du dossier de tests.

On remarque que cette fonction n'a qu'un seul chemin possible qui est C1 : {1;2;3;4}. Dans ce cas pour vérifier le fonctionnement de la fonction *dechiffrement_RC4* , on a juste besoin de réaliser un seul test.

Référence du test appliqué : dechiffrement_RC4
Responsable : Maxime BOGNON
Date de réalisation du test : 01/03/2024
Résultat du test : (OK, KO, non fait, dérogé)

Chemin	$encrypted_password	$key	Résultat attendu	Résultat du test
C1	45A01F645FC35B383552544B9BF5	Secret	Attack at dawn	OK

Identification du test :	PRGA	Version : 1
Description du test : PRGA (Pseudo Random Generation Algorithm) est un algorithme utilisé dans la cryptographie pour générer une séquence pseudo-aléatoire de bits ou d'octets,		
Ressources requises : PhpStorm et le module PHPUnit		
Responsable : Maaz Norat		

fonction testée :

```php
function PRGA(&$S, $n) {
    #1
   $i = 0;
   $j = 0;
   $keystream = array();

   while (count($keystream) < $n) {#2

        #3
       $i = ($i + 1) % 256;
       $j = ($j + $S[$i]) % 256;
       $temp = $S[$i];
       $S[$i] = $S[$j];
       $S[$j] = $temp;
       $zi = $S[($S[$i] + $S[$j]) % 256];
       array_push($keystream, $zi);
   }

   return $keystream; #4 // Retourne le flux de clés généré
}
```

On remarque que cette fonction n'a qu'un seul chemin possible qui est C1 : {1;2;3;4}. Dans ce cas pour vérifier le fonctionnement de la fonction *dechiffrement_RC4* , on a juste besoin de réaliser un seul test.

Référence du test appliqué : testPRGA
Responsable : Maaz Norat
Date de réalisation du test :
Résultat du test : (OK, KO, non fait, dérogé)

Chemin	$S	$keystream	Résultat attendu	Résultat du test
C1	range(0,255)	taille de 10 PRGA=(10, $S)	Le keystream devrait avoir une longueur de 10 et toutes ses valeurs doivent être des entiers dans la plage de 0 à 255.	OK

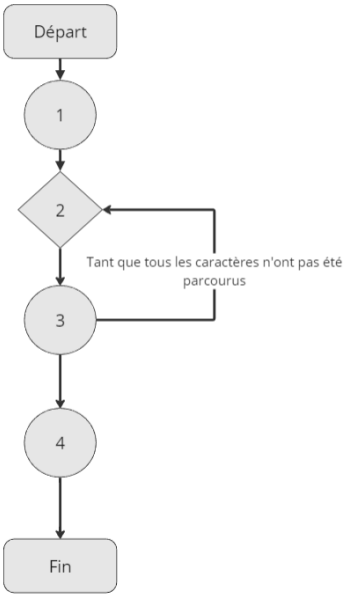
Identification du test :	KSA	Version : 1
Description du test : KSA (Key Scheduling Algorithm) est un algorithme utilisé dans le chiffrement RC4 pour initialiser le tableau d'état, également connu sous le nom de "S-box"		
Ressources requises : PhpStorm et le module PHPUnit		
Responsable :Maaz Norat		

fonction testée :

```php
function KSA($key) {
    $S = range(0, 255);
    $j = 0;
    $key_length = strlen($key);

    for ($i = 0; $i < 256; $i++) {
        $j = ($j + $S[$i] + ord($key[$i % $key_length])) % 256;
        $temp = $S[$i];
        $S[$i] = $S[$j];
        $S[$j] = $temp;
    }

    return $S; // Retourne le tableau d'état initialisé
}
```



On remarque que cette fonction n'a qu'un seul chemin possible qui est C1 : {1;2;3;4}. Dans ce cas pour vérifier le fonctionnement de la fonction *dechiffrement_RC4* , on a juste besoin de réaliser un seul test.

Référence du test appliqué : testKSA
Responsable : Maaz Norat
Date de réalisation du test :
Résultat du test : (OK, KO, non fait, dérogé)

Chemin	$ksaResult	Vérifcation	Résultat attendu	Résultat du test
C1	Initialisation d’un tableau à 256	longueur $ksaResult = 256 type int	Un tableau d'état de 256 éléments et resul == $ksaResult	OK