Near collisons and their impact on biometric security (code)

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Namespace Documentation

2.1 Approach_stat Namespace Reference

Functions

- def d_k (a, b, k)
- def Make_partition (Indice_list)
- def Compute_I (V)
- def Compute_A (I, D, a)
- def get_epsilon_0 (epsilon, lenD)
- def get_da (a, D)
- def Compute_N (p, a, I)
- def simulated_annealing (D, epsilon)
- def Back_to_F2 (solution, D, I)
- def get_cost (state, D, epsilon, I)
- def get_neighbors (state, epsilon, D, I)
- def delete_i_in_list (X, i, label)
- def clustering (X, seuil)
- def Partition (X, epsilon)
- def test_partition (n, epsil, nbr_client, sead=123)
- def several_iter (n, epsilon, nbr_cli, sed, repet)
- def main_test (nbr_thread, L, rep=1000)
- def several_iter_marche (n, epsilon, nbr_cli, sed, repet)
- def main_test_marche_alea (nbr_thread, L, rep=1000)

Variables

- **thread** = multiprocessing.cpu_count()

2.1.1 Detailed Description

```
@package Approach_stat
  This module provide the algorithm for the simulated annealing.
```

2.1.2 Function Documentation

2.1.2.1 Back_to_F2()

```
def Approach_stat.Back_to_F2 (
              solution,
              D,
              I )
```

```
This function throw back a solution of the simulated annealing in F_2.
@param solution The solution.
@param D The database.
@param I The partition of N for the database.
@return A vector of lenD coordinate with the value epsilon.
```

2.1.2.2 clustering()

```
def Approach_stat.clustering (
              X,
              seuil )
```

```
This function compute the clusters for a given database and a given threshold.
@param X The database.
@param seuil The threshold.
@return The labels of the clusters and the number of clusters.
```

2.1.2.3 Compute_A()

```
def Approach_stat.Compute_A (
              I,
              D,
              a )
```

```
This function compute the matrix A decribe in the paper.
@param I The partition.
@param D The database.
@param a The reference point.
@return The matrix A.
```

2.1.2.4 Compute_I()

```
def Approach_stat.Compute_I (
              V )
```

```
This function returns I a partition of N = {1,..,n}.
@param V The database.
@return A partition of N.
```

2.1.2.5 Compute_N()

```
def Approach_stat.Compute_N (
              p,
              a,
              I )
```

```
This function compute the vector distance between a point and another point on I.
@param a The point.
@param p The other point.
@param I the partition.
@return The vector distance between the point p and a on I.
```

2.1.2.6 d_k()

```
def Approach_stat.d_k (
              a,
              b,
              k )
```

```
This function compute the distance between a and b on the coordinate indexed by k.
@param a The first vector.
@param b The second vector.
@return The distance.
```

2.1.2.7 delete_i_in_list()

```
def Approach_stat.delete_i_in_list (
              X,
              i,
              label )
```

```
This function delete the element indexed by i in the database and in the clusters.
@param X The database.
@param i The targeted index.
@param label The clusters labels of X.
@return A new list and new labels without the targeted element.
```

2.1.2.8 get_cost()

def Approach_stat.get_cost (
state,
D,
epsilon,
I)

```
Calculates cost of the argument state for your solution.
@param state The current state of the solution.
@param D The database.
@param epsilon The threshold.
@param I The partition.
@return The cost of the argument state.
```

2.1.2.9 get_da()

def Approach_stat.get_da (
a,
D)

```
This function compute the vector distance between a point and a database.
@param a The point.
@param D The Database.
@return The vector distance between the point a and the database D.
```

2.1.2.10 get_epsilon_0()

def Approach_stat.get_epsilon_0 (
epsilon,
lenD)

```
This function compute the vector of lenD coordinate with the value epsilon.
@param epsilon The threshold.
@param lenD The size of D.
@return A vector of lenD coordinate with the value epsilon.
```

2.1.2.11 get_neighbors()

def Approach_stat.get_neighbors (
state,
epsilon,
D,
I)

```
This function returns neighbors of the argument state for your solution.
@param state The solution current state.
@param D The database.
@param I The partition of N for the database.
@param epsilon The threshold.
@return The neighbors of the argument state for your solution.
```

2.1.2.12 main_test()

```
def Approach_stat.main_test (
              nbr_thread,
              L,
              rep = 1000 )
```

```
"Run and test the whole attack with the given parameters.
@param nbr_thread The number of thread to use for the tests.
@param L The vector of parameters.
```

2.1.2.13 main_test_marche_alea()

```
def Approach_stat.main_test_marche_alea (
              nbr_thread,
              L,
              rep = 1000 )
```

```
"Run and test the master-template search with the given parameters.
@param nbr_thread The number of thread to use for the tests.
@param L The vector of parameters.
```

2.1.2.14 Make_partition()

```
def Approach_stat.Make_partition (
              Indice_list )
```

```
A function for Compute_I.
```

2.1.2.15 Partition()

```
def Approach_stat.Partition (
              X,
              epsilon )
```

```
This function run the whole attack.
@param X The database.
@param seuil The threshold.
@return The partion of X.
```

2.1.2.16 several_iter()

def Approach_stat.several_iter (
n,
epsilon,
nbr_cli,
sed,
repet)

```
Set up experiments
```

2.1.2.17 several_iter_marche()

def Approach_stat.several_iter_marche (
n,
epsilon,
nbr_cli,
sed,
repet)

```
Set up for test but only for the master-template
```

2.1.2.18 simulated_annealing()

def Approach_stat.simulated_annealing (
D,
epsilon)

```
Peforms simulated annealing to find a solution
@param D The database.
@param epsilon The threshold.
@return The epsilon master template of D if it has been found.
```

2.1.2.19 test_partition()

def Approach_stat.test_partition (
n,
epsil,
nbr_client,
sead = 123)

```
Set up experiments.
```

2.2 center Namespace Reference

Functions

- def I_0 (template)
- def Find_eps_cov_temp (template, tau)
- def several_iter2 (n, epsilon, nbr_cli, sed, repet)
- def main_test_mt (L, rep=10000)

Variables

- list **Res_thread** = [ ]

2.2.1 Detailed Description

```
@package Center
  This module provide the algorithm for the IP approach.
```

2.2.2 Function Documentation

2.2.2.1 Find_eps_cov_temp()

```
def center.Find_eps_cov_temp (
              template,
              tau )
```

```
This function compute the master-template of a database.
@param template The database.
@param tau The threshold.
@return The master-template of the database.
```

2.2.2.2 I_0()

```
def center.I_0 (
              template )
```

```
Return the index where all the vectors have the same value.
@param template The database.
@return The list of index where all vectors has the same value.
```

2.2.2.3 main_test_mt()

```
def center.main_test_mt (
              L,
              rep = 10000 )
```

```
Launch tests to find an epsilon master template.
@param L The vector of parameters
```

2.2.2.4 several_iter2()

```
def center.several_iter2 (
              n,
              epsilon,
              nbr_cli,
              sed,
              repet )
```

```
Set up for experiments
```

2.3 Partition Namespace Reference

Functions

- def delete_i_in_matrix (X, i)
- def print_partition (L)
- def gluttony (L, epsilon)
- def test_gouton (n, epsilon, nbr_cli, sezd, repet)
- def partition2 (X, epsilon)
- def test_partition2 (n, epsil, nbr_client, sead=123)
- def several_iter2 (n, epsilon, nbr_cli, sed, repet)
- def main_test_mt (L, rep=10000)

Variables

- **thread** = multiprocessing.cpu_count()
- list **Res_thread** = [ ]
- list **Res_glout** = [ ]

2.3.1 Detailed Description

```
@package Partition
  This module provide the algorithm tests of the whole attack for the IP approach.
```

2.3.2 Function Documentation

2.3.2.1 delete_i_in_matrix()

```
def Partition.delete_i_in_matrix (
              X,
              i )
```

```
"Remove the i-th element of the dissimililarity matrix.
@param X The dissimilarity matrix.
@param i The index to remove.
@return The matrix without i.
```

2.3.2.2 gluttony()

```
def Partition.gluttony (
              L,
              epsilon )
```

```
"The gluttony algorithm to do the partitionning.
@param L The database.
@param epsilon The threshold.
@return The partition.
```

2.3.2.3 main_test_mt()

```
def Partition.main_test_mt (
              L,
              rep = 10000 )
```

```
Run the main experimentation using some parameters.
@param L A list of parameters
```

2.3.2.4 partition2()

```
def Partition.partition2 (
              X,
              epsilon )
```

```
The whole attack using the IP algorithm
@param X The database.
@param epsilon The threshold
@return The new database.
```

2.3.2.5 print_partition()

```
def Partition.print_partition (
              L )
```

```
"Debug Function print a partition to check if it works.
@param L The partition.
```

2.3.2.6 several_iter2()

```
def Partition.several_iter2 (
              n,
              epsilon,
              nbr_cli,
              sed,
              repet )
```

```
Set up for the experimentations
```

2.3.2.7 test_gouton()

```
def Partition.test_gouton (
              n,
              epsilon,
              nbr_cli,
              sezd,
              repet )
```

```
Set up to test the gluttony algorithm.
```

2.3.2.8 test_partition2()

```
def Partition.test_partition2 (
              n,
              epsil,
              nbr_client,
              sead = 123 )
```

```
Set up for the experimentations
```

2.4 utils Namespace Reference

Functions

- def start_timer ()
- def stop_timer ()
- def get_time ()
- def gen_template (size_of_template, sed=123)
- def gen_cli (size_of_template, n, sed=123)
- def gen_DB (size_of_template, n, sed=123)
- def add_one_template (size_of_template, template_list, sed=123)
- def pt_in_b (r, n)
- def compute_HammingDistance_matrix (X)
- def distance_hamming (A, B)
- def opti_ham_dist_tau (A, B, T)
- def Add (A, B)
- def gen_mask (alpha, n)
- def gen_near_templates (n, nbr_client, tau, sed)
- def compute_cluster (Liste_vecteur, cluster_label, cluster_index)
- def compute_max_distance (L, moyen)
- def compute_center_in_R (template)
- def compute_vecteur_moyen (L)
- def negpart (x)
- def list_cmp (A, B)
- def verification (res, template, tau)
- def sample (Liste, indice)
- def mod_gen_cli (size_of_template, n, sed=123)

Variables

- int **tmps1** = 0
- int **tmps2** = 0

2.4.1 Detailed Description

```
@package utils
  This module provide some global functions.
  The needed packages are math, numpy, time and random.
```

2.4.2 Function Documentation

2.4.2.1 Add()

```
def utils.Add (
              A,
              B )
```

```
This function return the XOR of two vectors.
@param A The first template.
@param B The second template.
@return  The XOR vector.
```

2.4.2.2 add_one_template()

```
def utils.add_one_template (
              size_of_template,
              template_list,
              sed = 123 )
```

```
This function add a random template in a template list.
@param size_of_template The size of the wanted template.
@param template_list The current template list.
@param sed This is the seed for reproduce some executions.
@return The inputed list with one more template.
```

2.4.2.3 compute_center_in_R()

```
def utils.compute_center_in_R (
              template )
```

```
This function compute the mean vector in the reel field of a database and return it.
@param template The database.
@return The mean vector in the reel field of a database.
```

2.4.2.4 compute_cluster()

```
def utils.compute_cluster (
              Liste_vecteur,
              cluster_label,
              cluster_index )
```

```
This function return the elements in the cluster cluster_index.
@param Liste_vecteur The database.
@param cluster_label The result of the clustering algorithm.
@param cluster_index The cluster we want to get.
@return A list with all the element in the cluster cluster_index.
```

2.4.2.5 compute_HammingDistance_matrix()

```
def utils.compute_HammingDistance_matrix (
              X )
```

```
This function compute the dissimilarity matrix of a template database using the Hamming distance.
@param X The template database.
@return The dissimilarity matrix.
```

```
This function compute the dissimilarity matrix for the hamming distance of a database.
@param X A database.
@return The matrix of dissimilarity.
```

2.4.2.6 compute_max_distance()

```
def utils.compute_max_distance (
              L,
              moyen )
```

```
This function return the maximum pairwise distance between a database and a vector.
@param L The database.
@param moyen The vector.
@return The maximum pairwise distance between L and moyen.
```

2.4.2.7 compute_vecteur_moyen()

```
def utils.compute_vecteur_moyen (
              L )
```

```
This function compute the mean vector of a database and return it.
@param L The database.
@return The mean vector of a database.
```

2.4.2.8 distance_hamming()

```
def utils.distance_hamming (
              A,
              B )
```

```
This function compute the Hamming distance between two templates of same size.
@param A The first template.
@param B The second template.
@return The distance between both template.
```

2.4.2.9 gen_cli()

```
def utils.gen_cli (
              size_of_template,
              n,
              sed = 123 )
```

```
This function allow to generate a whole database: n binary templates of size size_of_template. At the end, eac
@param size_of_template The size of the wanted template.
@param n The number of templates wanted in the database.
@param sed This is the seed for reproduce some executions.
@return n vector in {0,1} of lenght size_of_template.
```

2.4.2.10 gen_DB()

```
def utils.gen_DB (
              size_of_template,
              n,
              sed = 123 )
```

```
This function allow to generate a whole database: n binary templates of size size_of_template. At the end, eac
@param size_of_template The size of the wanted template.
@param n The number of templates wanted in the database.
@param sed This is the seed for reproduce some executions.
@return n vector in {0,1} of lenght size_of_template.
```

2.4.2.11 gen_mask()

```
def utils.gen_mask (
              alpha,
              n )
```

```
An helping function for the gen_near_templates method.
```

2.4.2.12 gen_near_templates()

```
def utils.gen_near_templates (
              n,
              nbr_client,
              tau,
              sed )
```

```
This function return list of several close templates of size n (under tau).
@param n The size of templates we want.
@param nbr_client The number of templates we want to create.
@param tau The thresold for close templates.
@param sed This is the seed for reproduce some executions.
@return  A list of several close templates (under tau).
```

2.4.2.13 gen_template()

```
def utils.gen_template (
              size_of_template,
              sed = 123 )
```

```
This function allow to generate a binary template of size size_of_template.
@param size_of_template The size of the wanted template.
@param sed This is the seed for reproduce some executions.
@return A vector in {0,1} of lenght size_of_template.
```

2.4.2.14 get_time()

```
def utils.get_time ( )
```

```
Return the elapsed times between start_timer() and stop_timer().
   @return (tmps2 - tmps1).
```

2.4.2.15 list_cmp()

```
def utils.list_cmp (
              A,
              B )
```

```
This function said if two lists A and B are equal or not.
@param A The first list.
@param B The second list.
@return True if the two lists are equal and false otherwise.
```

2.4.2.16 mod_gen_cli()

```
def utils.mod_gen_cli (
              size_of_template,
              n,
              sed = 123 )
```

```
This function create a database.
@param size_of_template The size of templates.
@param n The number of templates.
@return A list of n templates.
```

2.4.2.17 negpart()

```
def utils.negpart (
              x )
```

```
This function compute the negative part of x.
@param x An integer.
@return 0 if x > 0 and -x otherwise.
```

2.4.2.18 opti_ham_dist_tau()

```
def utils.opti_ham_dist_tau (
              A,
              B,
              T )
```

```
This function say if distance between two template is under a thresold.
@param A The first template.
@param B The second template.
@param T The thresold.
@return True or False.
```

2.4.2.19 pt_in_b()

```
def utils.pt_in_b (
              r,
              n )
```

```
This function count the number of template in a ball of radius r in a space of size n.
@param r The radius of the ball.
@param n The space size (2**n).
@return The cardinality of a ball in F_2^n of radius r.
```

2.4.2.20 sample()

```
def utils.sample (
              Liste,
              indice )
```

```
This function return a sublist of Liste which contain the elements at the index indice.
@param Liste The list of elements.
@param indice The list of wanted index.
@return A sublist of Liste which contain the elements at the index indice.
```

2.4.2.21 start_timer()

```
def utils.start_timer ( )
```

```
"Start the timer: it store the current time in tmps1.
    @return True.
```

2.4.2.22 stop_timer()

```
def utils.stop_timer ( )
```

```
"End the timer: it store the current time in tmps2.
    @return False.
```

2.4.2.23 verification()

```
def utils.verification (
              res,
              template,
              tau )
```

```
This function said if res is a master-template of template database.
@param res A vector.
@param template The database.
@param tau The threshold.
@return True res is a master-template and false otherwise.
```

Index