EE 685 Final Report: Tree structure based feature mapping algorithm

Zaoyi Chi

May 8, 2019

Abstract

This report will mainly discuss the feature mapping algorithm and the implementation based on this method. The PUF(physical unclonable function) is a technology which been widely used in informatics security. The algorithm I proposed is going to base on the image dendrite PUF and make two PUF matched up robustly, even the image in client port has some noise or problem of image photoing.

In this research report, the most widely used feature mapping algorithms are going to be discussed. Here, I also proposed a new feature mapping algorithm is more reliable on dendrite PUF feature mapping, which can ignore most of the noise beside the significant dendrite. The method includes three parts: image pre-processing, the feature extraction, and feature mapping. Authenticating the user, the client image needs to check every model in the database, to find out that the image has the most significant similarity. Moreover, the method is highly based on graph theory, so the database will only store the tree data.

Index Terms: PUF, feature mapping, image processing, graph theory, authentication.

1 Introduction

PUF is a security technique that is based on a random physical feature. This random technology can make the hacker impossible to regenerate the same PUF because of the true random number generation of image PUF. Image-based PUFs represents a category of PUFs based on random visual features of physical objects evaluated by an imaging method [2]. The dendrite is the most significant part of each image PUF cell. This image dendrite has some common characteristics. They all have a circle in the middle of the dendrite, and multiple random branches without cycle.

There are lots of feature mapping algorithms, such as Harris, SIFT, SURF, Min Eigen, etc. They are the most widely used methods, which can also suit for most of the case in feature extraction and mapping. The most difficulty of feature mapping is that the issue of scale invariant. Ideally, two images with the same item and different scale, the method should robust to calculate the scale

ratio between two images. The SURF method is an update and accelerated version of SIFT, but SIFT is more accurate than SURF since the calculation is based on the local float feature.

Graph theory is a mathematical way to express the relationship between node, tree, and graph. It is also the principle of some data structure in today's computer science region, such as various tree and link list. Notably, the tree structure has been widely implemented into the search engine, routing and UAV network.

Feature mapping algorithm relies on the image processing techniques, such as hough transformation. Here, I will introduce that the method also uses hough technique, and more depends on the theory of graph.

2 Methodology

2.1 Image processing

Usually, the image is hard to get the same high quality as the enrollment in the step of authentication of client port. It can be the problem of scratch, different noise making on the image sample, environment issue or the skewed image when taking a photo of the sample. Various noise and effects can make the image, which even has the same object as the reference image, hard to detect and identify. Therefore, a couple of steps of image processing can make the test image has some accurate information.

Since the primary dendrite has very different color compared to the background of the entire image, therefore, the K-means clustering color-based segmentation can extract the image structure of the dendrite and get rid of the most unnecessary noise of the picture. After the operation of the rough image, some noise may still be kept since a similar color with the main dendrite object. The next step is to convert the retained object to the monochrome image by the predefined threshold. After the particle filtering, most of the noise has been removed, and the dendrite structure is kept. The thinning operation can make the main structure to be the skeleton, which could be much easier to detect main features.

2.2 Feature extraction

The main idea of extracting main features depend on the Depth-first principle [3]. The algorithm starts with the root of the skeleton, which is the middle circle of the dendrite object. It goes through the structure of image pixel by pixel, and use a 3×3 frame box to scan along with the skeleton. In each iteration, the frame detects how many new pixels in the following, to determine what type of current pixel. In general, the leaf node has no new pixel; bifurcation node has two or more sub-branches. The pixel, which has one new pixel, will not be considered inside of the structure, but the information of node to node is necessary to mark down. Moreover, the algorithm can not only just using DFS

[3], but also some tricks to make sure there is no overlap or any structure change because of the scanning order.

2.3 Feature mapping

2.3.1 Feature information pre-procssing

The feature extraction gives me the information in a more efficient way to store the information of dendrite PUF image into the database without memory-wasting. Since the different images have varying information, even for the same object with different photo taking method, the extracted feature data may vary slightly. To make the mapping more efficient and easier, the Algorithm. 1 shows the detail to normalize the data between the different reference object and test object. Here,i and j is the index information of test and reference object respectively. Notation $p(i)$ is the parent index of test image. The distance and angle between two node index are denoted as $d_{i,j}$ and $\theta_{i,j}$.

Algorithm 1: Normalization of test and reference trees

Result: The normalized $d^t_{i,p(i)}$ and $\theta^t_{i,p(i)}$, $d^r_{i,p(i)}$ and $\theta^r_{i,p(i)}$.

Inputs: $d^t_{i,p(i)}$ and $\theta^t_{i,p(i)}$, $d^r_{i,p(i)}$ and $\theta^r_{i,p(i)}$;

Compute $u^\theta_t := \frac{\sum \theta^t_{i,p(i)}}{n_t}$, $u^d_t := \frac{\sum d^t_{i,p(i)}}{n_t}$;

Compute $\sigma^d_t := \sqrt{\frac{\sum_{i=1}^{N}\left(d_{i,p(i)}-u^d_t\right)^2}{N}}$, $\sigma^\theta_t := \sqrt{\frac{\sum_{i=1}^{N}\left(\theta_{i,p(i)}-u^\theta_t\right)^2}{N}}$;

Normalize $d^t_{i,p(i)} := \frac{d^t_{i,p(i)}-u^d_t}{\sigma^d_t}$, $\theta^t_{i,p(i)} := \frac{\theta^t_{i,p(i)}-u^\theta_t}{\sigma^\theta_t}$;

Normalize $d^r_{i,p(i)} := \frac{d^r_{i,p(i)}-u^d_t}{\sigma^d_t}$, $\theta^r_{i,p(i)} := \frac{\theta^r_{i,p(i)}-u^\theta_t}{\sigma^\theta_t}$;

2.3.2 Node to node distance generation

The key mapping factor is the relative distance between the node in test and node in reference. The proposed Algorithm.2 Use the normalized distance d and θ and a tree structure to find out the test parents to reference parents distance. In each iteration, the relevant distance will give a penalty α, and parameter m decides the importance of angle difference or distance difference. The ϵ denotes the weight of strong relative distance and absolute distance between a node to root.

2.3.3 Consistent score computation

Algorithm.3 computes how accurate between two nodes in the tree structure. Based on the results from Algorithm.2, the rough mapping is using Munkres algorithm [1]. It will give rough results of linked test nodes and reference nodes pairs. The method checks each mapped node between its parent node, children

Algorithm 2: Level based distances

Result: The strong distance $D_{i,j}^{total}$ of two random nodes of different tree.

Inputs: node N_t^i, node N_r^j, reference tree R, test tree T , penalty α, ratio parameter ϵ factor m; $D_{i,j}^{total} := 0$;

Level $l := 0$;

while $N_i^{temp} \neq null$ *or* $N_j^{temp} \neq null$ **do**

$D_{i,j}^{total} :=$
$D_{i,j}^{total} + \alpha^l \; norm[m(|d_{i,p(i)} - d_{j,p(j)}) + (1-m)(|\theta_{i,p(i)} - \theta_{j,p(j)})|]$;

$i \leftarrow P(i)$;

$j \leftarrow P(j)$;

$l := l + 1$;

end

$D_{i,j}^{total} := \epsilon D_{i,j}^{total} + (1-\epsilon)(D_{i,root} - D_{j,root})$;

nodes, and sibling nodes, to see if they all matched up, and give a more precise consistency score of $\mathcal{S}$.

2.3.4 Mapping between tree and tree

The Algorithm.4 computes the similarity between two trees. This score will be kept in the database and compare the score of each pair, to get the most similar one in the whole database. Between tree and tree, the method set up the criterion to decide if the consistency score $\mathcal{S}$ is satisfied. The matched pairs of the node, which does not meet the requirement, need to go back to inconsistent node set and go through the whole method again with dropped penalty value λ, and different weight value in the case. After multiple iterations, the feature matching rate is going to convergence to a specific amount, which is considered as the final similarity score of $\mathcal{H}$.

3 Results

The experiment testing is based on 50 dendrite image samples. Also, the trial goes through the image in the ideal case, which the test image is the enrolled image in the whole database without any kind noise. The results show me 100% accuracy of entire 50 samples matched up with their references.

The testing also makes the Gaussian and salt & pepper noise on a test image. The range of Gaussian noise is between 27.2dB to 26.6dB, salt & pepper noise in the range of 26.86dB to 26.65dB. The accuracy of noised image matched up correct one is from 100% to 50%.

On the testing of different popular feature mapping algorithm. The tests are based on the rotated image. Here, I covered Harris, Fast, SURF, Min Eigen, Brisk and the algorithm I proposed. To decide which one is more robust in a

different image, matched points should be less than 2 pixels distance in this case. The Brisk has 75% average accuracy, and Fast has around 76% accuracy. Harris, Min Eigen, and Surf have about 85%, 90% and 70% accuracy. The method I proposed on tree structure technique has 95% accuracy, which performs best in most image testing.

The skewed image can simulate that people have some trouble in the capture photo image. The skewed angle is in the range of from 4 to 43. The algorithm works well before 15 degrees skewed; the average matching rate is still higher than 50 %. After that, the matching rate starts dropping in the average slope of 1.5 %/ deg.

Algorithm 3: Consistency score computation $\mathcal{C}(N_t^i, N_t^j)$

Result: Consistency score between two nodes $\mathcal{S}$
Inputs: node N_t^i, node N_r^j, test linked node set $\mathcal{L}_t$, reference linked node set $\mathcal{L}_r$, parameters α, β, γ;
Precondition: N_t^i and N_r^j must be matched pairs.
$\alpha \in [0, \frac{1}{2}], \beta \in [0, 1], \gamma \in [0, 1]$;
$\mathcal{S} := 0$;
if $N_t^{P(i)}$ *and* $N_r^{P(j)}$ *are linked,* $N_t^{P(i)} \leftrightarrow N_r^{P(j)}$ **then**
  $\mathcal{S} := \mathcal{S} + \alpha^2$
end
Compute the number of children of two nodes n_t^c, n_r^c;
Counter $Ct := 0$;
for $m \leftarrow 1$ **to** n_t^c **do**
  Linked pair index of $R : I_r := \mathcal{L}(N_t^{C^i(m)})$;
  if $I_r \in C^j$ **then**
    $Ct := Ct + 1$;
  end
end
$\mathcal{S} := \mathcal{S} + \beta \frac{Ct}{Max(n_t^c, n_r^c)}$;
Compute the number of siblings of two nodes n_t^s, n_r^s;
Counter $Ct := 0$;
for $m \leftarrow 1$ **to** n_t^s **do**
  Linked pair index of $R : I_r := \mathcal{L}(N_t^{S^i(m)})$;
  if $I_r \in S^j$ **then**
    $Ct := Ct + 1$;
  end
end
$\mathcal{S} := \mathcal{S} + \gamma \frac{Ct}{Max(n_t^s, n_r^s)}$;
$\mathcal{S} := \frac{\mathcal{S}}{3}$;

Algorithm 4: Mapping $\mathcal{M}(T_u, R_u, T_m, R_m, T, R, \mathcal{H})$

Result: Similarity score $\mathcal{H}$ between two full trees

Inputs: Full test tree T, full reference tree R, partial unmatched test tree T_u, partial unmatched reference tree R_u, matched partial test tree T_m, matched partial reference tree R_m, parameters α, β, γ, penalty λ, max iterations t_{max};

Criterion $\mathcal{A} := \frac{1}{3}(\alpha + \alpha^2 + \beta + \gamma)$;

Criterion $\mathcal{B} := \frac{1}{6}(\alpha + \alpha^2 + \beta + \gamma)$;

Temporary linked test nodes and reference nodes $(L^t, L^r) := Munkres(T_i, R_i)$;

Computer number n_t of nodes of L^t ;

for $i \leftarrow$ **to** n_t **do**

  test node index $x := L^t(i)$;

  reference node index $y := L^r(i)$;

  $\mathcal{S} := \mathcal{C}(x, y)$;

  if $\mathcal{S} \in [\mathcal{A}, \mathcal{B}]$ **then**

    $T_m := T_m + x$;

    $R_m := R_m + y$;

    $T_u = T_u - x$;

    $R_u := R_u - y$;

  end

end

Update $n_t \leftarrow$ number of the nodes of T_m ;

$N_t \leftarrow$ the number of nodes of full test Tree T ;

$N_r \leftarrow$ the number of nodes of full reference Tree R ;

$\mathcal{H}_{diff} := |\mathcal{H} - \frac{2n_t}{N_t + N_r}|$;

$\mathcal{H} := \mathcal{H} + \lambda^t \mathcal{H}_{diff}$;

$t := t + 1$;

if $t \neq t_{max}$ *or* $\mathcal{H} \neq 1$ **then**

  return;

else

  $\mathcal{H} := \mathcal{M}(T_u, R_u, T_m, R_m, T, R, \mathcal{H})$;

end

References

[1] James Munkres. *Topology.* Pearson Education, 2014.

[2] Saloomeh Shariati et al. *Image-based physical unclonable functions for anti-counterfeiting.* PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2013.

[3] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.