A2 – Steganography

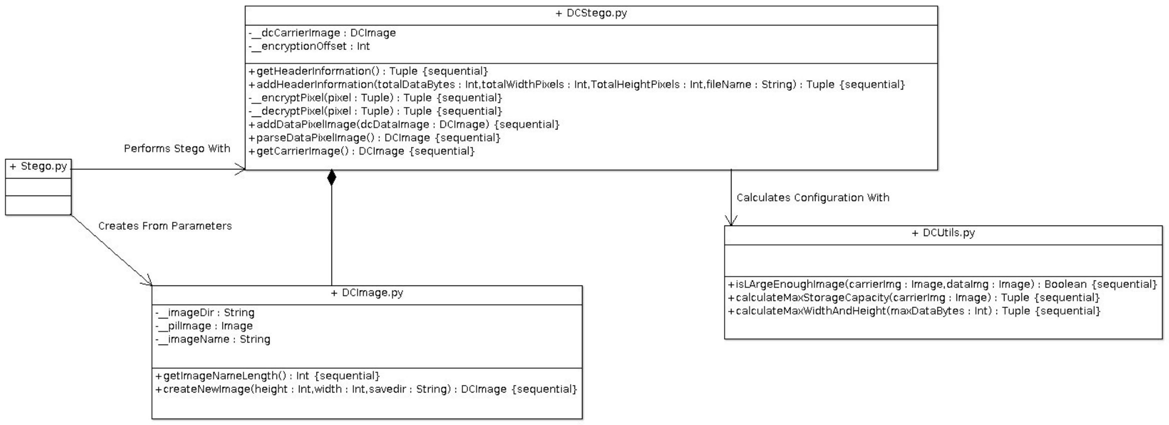
Ben Soer

A00843110

Table of Contents

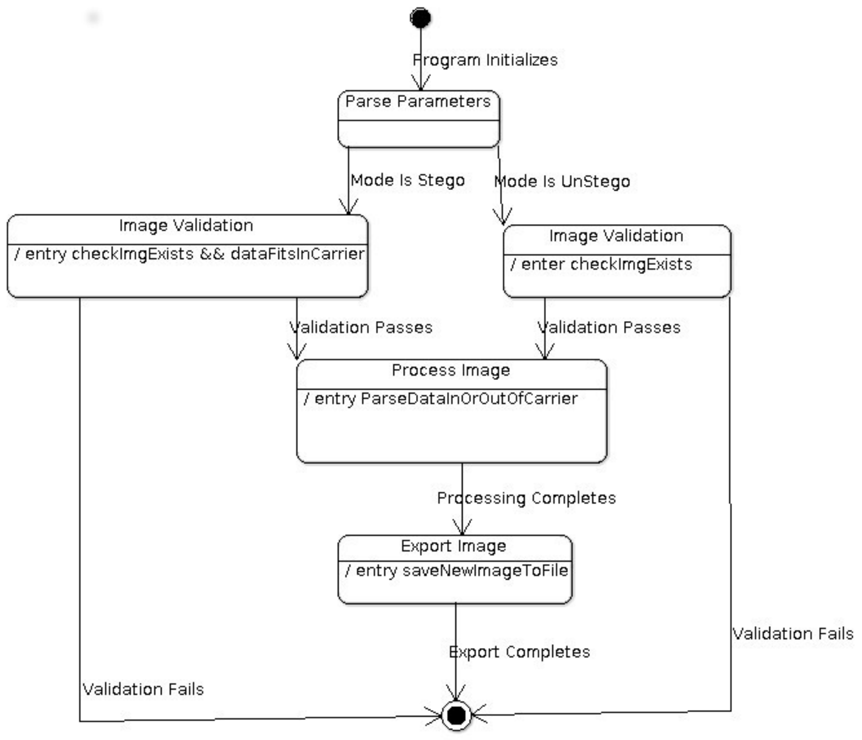
Class Diagrams

Original image can be found under imgs/SteganographyClassDiagram.jpg


+ DCStego.py
-__dcCarrierImage : DCImage
-__encryptionOffset : Int
+getHeaderInformation() : Tuple {sequential}
+addHeaderInformation(totalDataBytes : Int,totalWidthPixels : Int,TotalHeightPixels : Int,fileName : String) : Tuple {sequential}
-__encryptPixel(pixel : Tuple) : Tuple {sequential}
-__decryptPixel(pixel : Tuple) : Tuple {sequential}
+addDataPixelImage(dcDataImage : DCImage) {sequential}
+parseDataPixelImage() : DCImage {sequential}
+getCarrierImage() : DCImage {sequential}
Performs Stego With
+ Stego.py
Creates From Parameters
Calculates Configuration With
+ DCUtils.py
+isLArgeEnoughImage(carrierImg : Image,dataImg : Image) : Boolean {sequential}
+calculateMaxStorageCapacity(carrierImg : Image) : Tuple {sequential}
+calculateMaxWidthAndHeight(maxDataBytes : Int) : Tuple {sequential}
+ DCImage.py
-__imageDir : String
-__pilImage : Image
-__imageName : String
+getImageNameLength() : Int {sequential}
+createNewImage(height : Int,width : Int,savedir : String) : DCImage {sequential}

Finite State Machines

Original image can be found under imgs/SteganographyStateMachine.jpg



Pseudocode

stego.py (Program Main Entry)

main()

```
1. Parse Command Line Arguments. If none or –HELP call printHelp()
2. Determine Mode. If Stego
    1. Configure Logging For Stego Procedure
    2. Fetch Image. Test If Data Fits Inside Carrier. If not, Terminate
    3. Process Image
    4. Save Processed image to the file system
    5. Cleanup imaging library resources
```

printHelp()

```
1. Print Help Information
```

dcimage.py

getImageNameLength()

```
1. Fetch Length of Image
```

canHoldImage()

```
1. Call DCUtils.isLargeEnough and return results
```

getPixelAccess()

```
1. Get PixelAccess object from Image object
```

getPilImage()

```
1. Return the pil image stored as class attribute
```

getPilWidth()

```
1. Get width of image stored as class attribute
```

getPilHeight()

```
1. Get height of image stored as class attribute
```

getImageName()

1. Return the file name of the image as stored as the class attribute.

createNewImage(height, width, savedir)

1. Create a plain new image using passed height and width
2. Save in savedir location
3. Close resources
4. Create a DCImage object passing savedir to the constructor
5. Return the DCImage

dcutils.py

isLargeEnoughImg(carrierImg, dataImg)

1. Call DCUtils.calculateMaxStorageCapacity passing the carrierImg
2. determine if the maxBytes is larger then the dataImg size in bytes
3. Return true or false

calculateMaxStorageCapactiy(carrierImg)

1. Calculate max storage capacity of the carrier image in bytes and how many bits it will take to represent that number. Round up as a precaution to give extra space

calculateMaxWidthAndHeight(maxDataBytes)

1. Calculate max possible pixels that the image could be based on the maxDataBytes and how many bits it would take to represent that number. Round up as a precaution to give extra space

dcstego.py

getHeaderInformation()

1. Determine max storage capacity of the carrier image
2. calculate space needed for max storage, width, height and name headers
3. parse out max bytes
4. parse out height
5. parse out width
6. parse out file name

7. Return index of first data pixel along with all parsed information in a tuple

addHeaderInformation(totalDataBytes, totalWidthPixels, totalHeightPixels, fileName)

1. Calculate max storage capacity of carrier image and bits needed to represent that value
2. encode max byte size of data image into calculated max space
3. calculate max width and number of bits needed to represent that value
4. encode width of data image into calculated max width space
5. calculate max height and number of bits needed to represent that value
6. encode height of data image into calculated max height space
7. encode static 30 characters into the image
8. Return indexes of where first data bit will go

encryptPixel(pixel)

1. Parse out red, blue green values from pixel
2. Apply ceasar cipher using offset amount. Correct pixels values if they go out of range
3. Put red,blue,green values back into pixel
4. return pixel

decryptPixel(pixel)

1. Parse out red, blue green values from pixel
2. Apply ceasar cipher using offset amount. Correct pixels values if they go out of range
3. Put red,blue,green values back into pixel
4. return pixel

addDataPixelImage(dcDataImage)

1. Calculate width and height of dcDataImage
2. Get PixelAccess object of data image
3. call addHeaderInformation() passing calculated header data
4. loop through data image pixels
    1. fetch data pixel
    2. parse into red, green, blue

   3. loop over red green and blue planes
      1. grab 3 carrier pixels. Loop over each red, green, blue plane in carrier pixels
         1. grad lsb from data image. Right shifting 0-7 times to get each bit
         2. alter carriers lsb to match currently fetched lsb from data image
      2. put altered palnes back into appropriate carrier pixels
      3. put altered carrier pixels into appropriate carrier pixel position
      4. increment by 3 so that next loop will get the next 3 carrier pixels in the carrier image

parseDataImage()

1. Calculate width and height of dcDataImage
2. Get PixelAccess object of data image
3. call getHeaderInformation() returning the data start posiitons, totalDataBytes, width, height and filename
4. loop through data pixels height and width
   1. parse data pixel
   2. create list of all planes in r,g,b order. Loop over all planes
      1. get 3 carrier pixel
      2. parse out RGB planes of each carrier pixel
      3. place each plane in reverse order (pixel3 – pixel1, GRB order) into a list
      4. loop through the pixelList
         1. left bit shift data int from 0-7 through loop
         2. grab lsb from each plane in the list and place into data int.
      5. replace plane being used in loop from step 2 with newly generated one in loop
      6. increment by 3 to get next 3 carrier image pixels
   3. Create new data pixel using updated planes from loop in step 2
   4. Insert data pixel into data image being built
5. Return data image

getCarrierImage()

1. Return the carrier image stored as a class attribute