Chapter 6

Secure Image Databases

This chapter presents a model for the development of a secure image database and its surrounding components. An entity defines specified parts of an image as objects, and each object is given a classification (this is done for example by the image's owner or by a database administrator). The original image is stored in an image repository (e.g. a server), and its associated objects are recorded in an image (relational) database. Any entity wishing to retrieve an image will only receive those parts of the image corresponding to that entity's classification or below. The whole process concerning secure image databases is divided into three related phases: the input phase, the storage phase, and the output phase. If these three phases are correctly implemented, an image database with the required level of security should result from it. A variation of this model will then be discussed which is more efficient, but not as dynamic.

6.1 Introduction

The importance of security is increasing continuously within most application domains, especially within business and government institutions. Sensitive information is often stored in electronic databases, and it may need to be classified according to a number of security levels (usually defined in the security policy). Multimedia data is also becoming increasingly popular, and so too is the need to securely store and retrieve this data according to security classifications.

Some terminology is necessary before presenting the model. An *image object* consists of a region, or number of regions, that represent something distinct within the image. Each image object is assigned a security classification. The database records the required information for each image object, including what methods where used to select the region(s) that make up that particular image object and all method specific information.

An *image database* is an electronic repository that allows information on specified parts of an image (the image objects) to be stored in an organised manner. A *secure image database* allows these image objects to be stored according to a defined classification. The image objects contain references to specified areas of the original image. This will allow the owner of an image (or another authorised entity) to specify different levels of confidentiality for different parts of the image. As a simple example, a military organisation may want to prevent certain individuals (such as generals) from appearing in certain photos unless an entity with a specified classification level (such as top-secret) requests the image.

A *background image* must be provided, which (ideally) is equivalent to the original image without its defined (worldly) image objects i.e. the background image should exclude as many potentially defined image objects as possible. The background image has the lowest classification (e.g. public in the case of the military model). Whenever an entity wants to retrieve an image from the secure image database, the entity's classification is taken into consideration. The authorised parts of the original image, and the relevant parts of the background image (for areas of the image that are above the entity's classification) are assembled and transmitted to the requesting entity. This ensures that the entity does not receive any part of the image that is above its classification.

At least one region selection method must be available that will allow two overlapping images to be separated. One of the methods discussed in the next section, the automated method, is especially suited for these cases. When an image is requested, the relevant parts of the image are put together before the image is transmitted across a communications medium to the requesting entity. The image must however be encrypted using a good encryption algorithm before it is transmitted, which will prevent any unauthorised entities from intercepting and viewing the image. The resulting image (after the classified parts have been removed) should also not look suspicious. A number of similar issues concerning secure image databases must be addressed. These issues will be discussed in more detail in the next chapter, and possible solutions will be provided.

6.2 The secure image database model

This section will present my model for the development of a secure image database and its surrounding components. The model is divided into three phases. The first phase (input phase) concerns activities such as region selection and object classification. These activities are performed through a graphics interface that has access to the image database. The second phase (storage phase) involves writing the relevant information from the input phase into the image database. The original image and background image are stored securely in an image repository (separate from the database), and the relevant references to these images are recorded in the image database. The third phase (output phase) concerns all activities that are performed when an image must be retrieved. The most important function for this phase is to ensure that only the authorised parts of the original image are assembled and transferred to the requesting entity.

The secure image database model is illustrated in Figure 6.1 below. The symbols n, m1, m2, ... , mn are variables. The variable n is equal to the number of image objects that are defined for that particular image. The value of the variables m1, m2, ... , mn represent the number of regions defined for image objects 1, 2, ... , n respectively. The solid and dashed arrows represent the flow of information between the components of the model. Dashed lines are used for the special cases where a request is being made between two components.

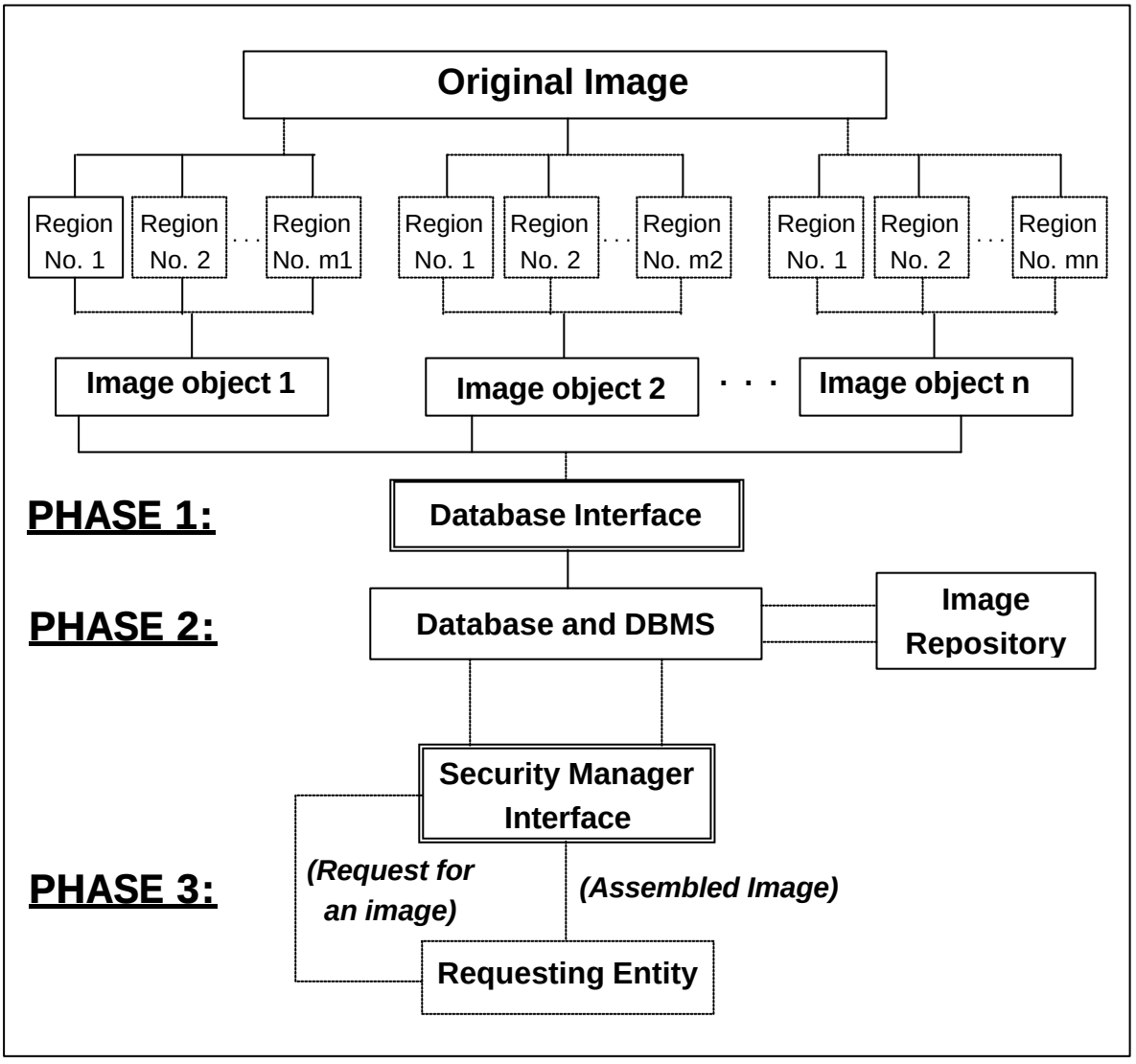

Figure 6.1: The secure image database model

This model assumes that a security classification system has been defined. Each image object that is created must be classified according to this classification system. Any entity that may need to make a request for an image must also have its own security classification, which will be used to decide what parts of the image the requesting entity is authorised to access. The security policy should also specify whether a DAC policy, MAC policy, or a combination of the two is to

be enforced when creating objects. The image database administrator(s) could for example be exclusively responsible for assigning security classifications to objects (if a MAC policy is used).

6.2.1 Phase one: The Input Phase

The purpose of this phase is to 'prepare' an image to be stored in the secure image database. The input for this phase is the image in its original form. An entity must now separate the image into its necessary components. This is performed using the *database interface*, which is a program that is allowed to directly communicate with the database. The relevant security procedures must therefore be taken to ensure that image database alterations can only be performed through the database interface.

Images vary in complexity. The composition of an image varies from simple shapes (such as rectangles and circles) with a very low resolution, to complex graphics (such as people and animals) that can have a very high resolution. Selecting the required parts of an image for classification is often not an easy task. Humans are easily able to distinguish between worldly objects such as animals and people, man-made objects, and many other objects. We are also able to distinguish between different parts of an object, such as the difference between a persons head and feet. A software program needs a huge amount of programming effort to be able to distinguish between different worldly objects like we do (see Nilsson (1998) for a discussion on this topic).

To keep the discussion more simple and general, I will assume that a user aids the database interface in specifying which regions of an image are classified, resulting in a semi-computerised selection process. The model can however be extended to allow for a more computerised selection process which makes use of robot vision, for example. The database interface contains a graphics component that will allow the image to be divided into objects that are relevant (as decided by the user). Any number of region selection methods can be used, some of which will be discussed next. It should be noted that any graphics technique capable of selecting specific regions of an image can in general be used to define the image objects. The following methods are mainly used to explain the model and any potential shortcomings, and have also been implemented as part of the secure image database prototype (discussed later in this chapter).

A simple method of selecting a region is to allow the user to 'click and drag' an elastic box over an area (let us call this method the '*box method*'). All the area covered by the rectangular box is counted as a single region. This method will benefit the programmer and the user, since it is easy for the programmer to implement this method, and it is easy for the user to select a region. This method is also efficient and economical, since only two coordinates need to be recorded in the image database, namely the top-left and bottom-right coordinates. The problem with this approach is that if two worldly objects in the image overlap, and only one of the objects needs to be selected, a part of the other object will be selected with the desired object.

This problem is illustrated in Figure 6.2 below. This simple diagram consists of two regions, the letter R and a picture of a smiling face, which overlap each other's rectangular space. The user wants to choose the letter R and uses the box method to select the area covered by this region. The result unfortunately is the letter R together with the top part of the smiling face. This simple example can be extended to more complex images, allowing us to conclude that the box method is limited to only certain cases.

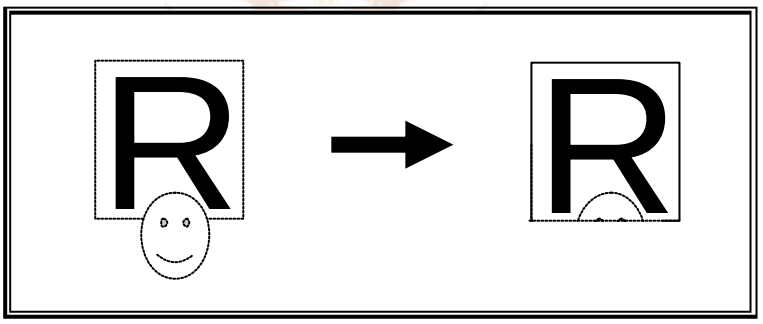

Figure 6.2: The box method problem

Another method of selecting a region is to allow the user to draw a polygon around an object in an image (let us call this method the '*polygon method*'). The start and end coordinates of each line that makes up the polygon are recorded in the database. Once the selection is completed to form a closed polygon, the user selects a reference point, either on the inside or the outside of the polygon, which will be used to reselect the region later (i.e. the user specifies whether the desired selection is everything inside or everything outside the polygon). This method allows more flexibility when selecting a region; the box method problem illustrated in Figure 6.2 can be solved using the polygon method (with some patience). The problem with this method is that detailed regions containing many curves and angles require a lot of effort from the user, and a large number of line coordinates may need to be recorded.

The box method and polygon method share a common problem: it is very difficult to select detailed regions accurately (i.e. without selecting any pixels outside of the desired region). The third method I discuss here (which I call the '*automated method*') helps solve this problem. The user first selects any pixel within the desired region. The graphics software component then scans all the pixels around the selected pixel, works out the boundary of the region, and automatically selects the region (within given parameters) surrounding the selected pixel.

When a user chooses to use the automated method on an image, the graphics component performs an averaging operation over the entire image, which should help to eliminate any irregularities in the pixels for future region selections on that image. The user must specify an *intensity variation value*, which is the maximum absolute value difference in colour intensity that is allowed between two adjacent pixels within the same region. When a pixel is selected, the intensity variation value is used to determine the boundaries of the region surrounding the selected pixel. This value must be specified before a region is selected (unless the previously specified value also applies to this region). If the desired region is not completely selected, or the region selected is too large, the intensity variation value for that region can be raised or lowered respectively.

The graphics software component uses an algorithm that compares the selected pixel to its adjacent pixels, each time calculating the absolute value difference of the intensity values between these adjacent pixels. If the difference is greater than the specified intensity variation value, a boundary value is recorded between these adjacent pixels. This process is recursively repeated for all adjacent pixels until a boundary is reached in all directions (i.e. the region surrounding the selected point has been found). The graphics component will then visually display the boundary for that region using the calculated boundary values.

The split-and-merge method can also be used to find the region surrounding a selected point. It should be noted however that a region selected using these region selection methods often does not represent a worldly object on its own. In this case, it is therefore necessary to combine a number of region selection operations in order to select the regions that make up the desired worldly object.

The next step is to use the database interface to create the image objects. The graphics component of the database interface allows the user to assign the selected regions of the image to image objects. The required information to reproduce each region is associated with the image object, and after the association has been made, the user assigns a security classification to the entire

image object. It is important to note that all the regions associated with a particular image object are given the same security classification. The image object with all its associations is written to the database through the database interface. This process is repeated for each image object.

Various rules must be applied by the database interface (in conjunction with its graphics component). These rules must be enforced before an image object is written to the database, and may vary from one region selection method to the next. For the polygon method, the graphics software component must ensure that the lines form a closed region (i.e. every line must be connected to two other lines). In the case of the automated method, although not essential, the database interface may be required to ensure that multiple regions associated with an image object are connected (i.e. for multiple regions associated with a particular image object, given any region, there exists another region such that at least one boundary value is shared between these two regions).

6.2.2 Phase two: The Storage Phase

This phase concerns writing the images and image objects to the relevant locations. A relational database is used to store the image information, and an image repository is used to store the actual images. The RDBMS for this model performs normal RDBMS activities (storage and retrieval); it does not perform complex image processing such as region selection.

The image database must contain all the information that is needed to be able to assemble an image together according to the required security classification. For each region that forms part of an image object, the region selection method used to create that region must be recorded (e.g. each method could be identified by a unique number). All the information that is needed to reconstruct that region from the original image must be recorded in the image database. If the box method was used, the database must contain the top-left and bottom-right coordinates of the selected region (according to the original image). In the case of the polygon method, the database must include the start and end coordinates of all the lines that make up the polygon, as well as the coordinates of the reference point. The automated method requires the coordinates of the pixel that was selected by the entity for that region, and the intensity variation value for that region.

Once all the regions for an object are properly recorded in the database, they must be linked to their image object. The image object is simply a unique identifier

listed in a database table (along with other unique identifiers) which is used as a reference by its associated regions. After all the relevant regions are linked to the image object, the image object's classification must be recorded. When an image must be retrieved, the list of image objects is scanned to see which parts of the image are above the requesting entity's security classification (i.e. which parts of the image must not be shown). This information will be used by the security manager interface (which will be discussed in section 6.2.3).

Once all the image objects have been created and recorded, they must be linked to the original image. A unique identifier is also assigned to each image, which is used as a reference by its image objects. The image identifiers, image objects, and their associated regions form a hierarchy, and are referenced in this manner. An image repository (e.g. a file-server) securely stores the original image and the background image. The RDBMS has direct access to the image repository. A requesting entity must however not be given direct access to the image repository; all images must be requested through the security manager interface exclusively. The image identifier contains the path and file names of the relevant images within the image repository. Finally, the parts of the image that have not been assigned to an image object are treated as having the lowest possible classification on default.

There may be cases where an entire image must be defined as an image object that forms part of a larger image (the smaller image is considered to be the only region for that image object). In this case, a unique method identifier should be used (as with the other region selection methods), and the smaller image must also be stored in the image repository. The image object must contain the path and file name of the smaller image within the image repository, as well as the top-left coordinate of the smaller image with respect to the larger (original) image.

6.2.3 Phase three: The Output Phase

The purpose of this phase is to ensure that an entity requesting an image receives only the parts of the image it is authorised to access. It is assumed that the entity is properly authenticated to have the claimed security classification (security issues regarding this model will be discussed further in the next chapter).

An entity makes a request for an image through the *security manager interface* (SMI). The SMI securely handles all requests for images and communicates with the database directly; the requesting entity can only retrieve images via the SMI.

The SMI receives the original and background images, and all the relevant information for that image (image objects, etc.) from the RDBMS (which has direct access to the image repository). The SMI also contains a graphics component which assembles the authorised parts of the original image and the remaining parts from the background image together into one single image. The assembled image is then sent to the requesting entity.

The graphics component of the SMI must contain all of the methods supported in the graphics component of the database interface. These methods must have the same implementation details; they must use the same graphic operations and algorithms. This is because the regions that were selected in the input phase will now be recalculated in this phase using the information that was gathered from the image database. This method of storing information regarding previously selected regions is efficient and economical, since only a small amount of information needs to be recorded and retrieved for each region (instead of for example storing and retrieving all the coordinates of the boundary pixels for each region to and from the database).

The SMI receives the method identification for each region from the database, which is used to determine which region selection method was used for any particular region. The SMI graphics component will then reapply that region selection method on the original image, using the information provided by the image database as input parameters. This process will calculate the region that was originally selected. For example, in the case where the automated method was used, the SMI graphics component will reselect the *originally* selected pixel using the *originally* specified intensity variation value on the *original* image, which will result in the required region.

In order to assemble the image, the SMI scans through all of the image objects that are associated with the requested image, and checks to see which of these image objects have a classification higher than the requesting entity's security classification. These unauthorised image objects are dealt with individually. Each unauthorised image object's associated regions are calculated (using the above process), and the area covered by these regions is substituted with the corresponding area in the background image (pixel by pixel). The aim of using a background image is to prevent the output image from looking suspicious. Ideally, the background image should be what the original image would look like without its defined image objects (i.e. the original image's background scenery). If a background image is not available, a default image with a uniform background

colour (white as default) is used instead (issues regarding suspicion are also discussed in more detail in the next chapter).

If the SMI performs all of these activities correctly, the result should be a complete image containing image objects that have a classification equal to or below the requesting entity's security classification. Once the image is correctly assembled, the SMI transmits the resulting image to the requesting entity. It is extremely important to update both graphics components (database interface and SMI), not just one of them. All the important aspects that are used in both the graphics components must be consistent with each other e.g. method identification. Other region selection methods may be added, as long as these changes are made to both graphics components.

This model does allow two image objects with different security classifications to both reference common regions. For example, two people could be holding the same worldly object such as a book, standing next to each other and holding the book in front of them. The person on the *left* is associated with an image object that has a low security classification, while the person on the *right* is associated with an image object that has a high security classification. The shared book is associated with both image objects. There is a potential problem in that if an entity with a low security classification wishes to retrieve the image, the resulting image will contain the person that was on the left with his/her hands in the air, holding nothing (because the book has a high security classification too). The solution to this problem depends on the particular case at hand, and whether certain factors must be taken into consideration (e.g. whether the book must also have a higher classification or whether the shared object is held in such a way that the absence of one of the people would not make gravitational sense). Chapter 7 will discuss these problems, and provide potential solutions for them (if possible).

Management of the database (such as changes to image object classifications) must be performed through the database interface exclusively. An entity must be properly authenticated before any changes can be made to the database. An authenticated entity should only be allowed to modify an image object if the entity has the same or higher classification than the image object. The database interface must ensure that database security and integrity is always maintained.

6.3 A prototype for the secure image database model

This section discusses a prototype that was developed for the above secure image database model. The main aim of the prototype was to discover through

implementation any problems that would need to be dealt with, and improvements that would need to be made in future research work regarding the secure image database model. Some relevant implementation issues will be discussed next. Problems and future improvements that were discovered from the prototype will then be discussed.

6.3.1 Implementation issues

The programming language I used to implement the prototype is Visual Basic, and the secure image database is a Microsoft Access relational database. Visual Basic was chosen because it allows a graphical user interface necessary for this prototype to be easily added, and it also allows a programmer to create an interface to an Access database without too much trouble. More time could therefore be spent on the region selection methods and the actual implementation of the model. Graphics manipulation in Visual Basic isn't as efficient as some other languages, but since this was only a prototype version, speed requirements were not set very high (the end product was adequately efficient).

The prototype consists of two main parts: the database interface component, and the security manager interface (SMI) component. The database interface component implements phase one (the input phase) and phase two (the storage phase) of the secure image database model. This component allows the user to choose an image as input, classify selected areas of the image, and write this information to the image database. The SMI component implements phase three (the output phase) of the secure image database model. This component allows a user to request an image, which is assembled according to the user's security classification (using the secure image database that is currently active).

When a user wishes to classify certain parts of an image according to some predefined classification system, the first task that must be performed by the user is to select the image using the database interface component. The prototype displays a list of all attached drives, and allows the user to select the location of the image that must be classified. Once chosen, the image is displayed. The prototype allows images of any size to be opened, although it naturally takes longer to open and graphically manipulate larger images.

The next step is for the user to specify the location of the secure image database. The database interface component will then verify the validity of the database by making sure that it contains a 'Classification' table with valid entries. This table

contains the names of all the possible classifications that can be given to a selected area of an image, as well as the classification level corresponding to each name. In the prototype, the lowest classification level is assumed to be the most important. In the military model, for example, a general may have a classification level equal to one, whereas a private may have a classification level equal to ten. The security policy for the organisation should be used to define what appears in the 'Classification' table, and it is assumed that only the correct authority has the required access to change values in the 'Classification' table.

The prototype allows the user to select a region by using any of the three methods discussed in the above model, namely: the box method, the polygon method, and the automated method. As discussed in section 6.2.1, the box method allows the user to drag an elastic box over the image to select an area, whereas the polygon method allows the user to draw a polygon over the image to select the area. A small improvement has been made to the automated method. Before the user clicks on a point in the image, two values must be specified. The first value (as described earlier) is the intensity variation value, which is the maximum absolute value difference that is allowed between two adjacent pixels' Red-Green-Blue (RGB) values in that region. In other words, the absolute value difference between the red, green, and blue colour components of these two pixels must all each be within the specified value. The second value that must be specified is the maximum absolute value difference of the RGB total between adjacent pixels in the same region. In other words, the absolute value difference of the total for each adjacent pixels' RGB components must be within this second specified value.

The prototype's automated method showed considerable improvement when trying to select a worldly object after the second value was added to the method. The problem was that in certain cases, a single intensity variation value would not accurately select the desired region. A small value would select a region that was too small, while slightly increasing this value would result in a region that was too large; therefore no single intensity variation value represented the desired region (a larger number of regions had to be selected and associated with that image object). The addition of the second value helped to improve the situation. A slightly larger intensity variation value with a reasonably low maximum RGB total (that is obviously higher than the intensity variation value) usually results in a region that more accurately represents the desired worldly object.

Whenever an image is loaded, a plain white image that has the same size dimensions as the original image is set as the default background image. The prototype allows the user to change this default background image for that

particular original image. When selecting a region, the prototype also gives the user the option to 'preview' the region when replaced with the background image (i.e. to see what the region would look like when an entity that is unauthorised to view that region makes a request for the image).

Once the user is satisfied with the region that has been selected, either a new object is created, or an existing object (in the object list) is selected. The correct classification must then be assigned to the object. The user will then update the object, which will save all relevant information into the object data structure. This operation ensures that all method specific rules are obeyed, such as ensuring that a closed region is formed when using the polygon method. The prototype allows an image object's classification to be changed and additional regions to be added at any time before it is written to the database.

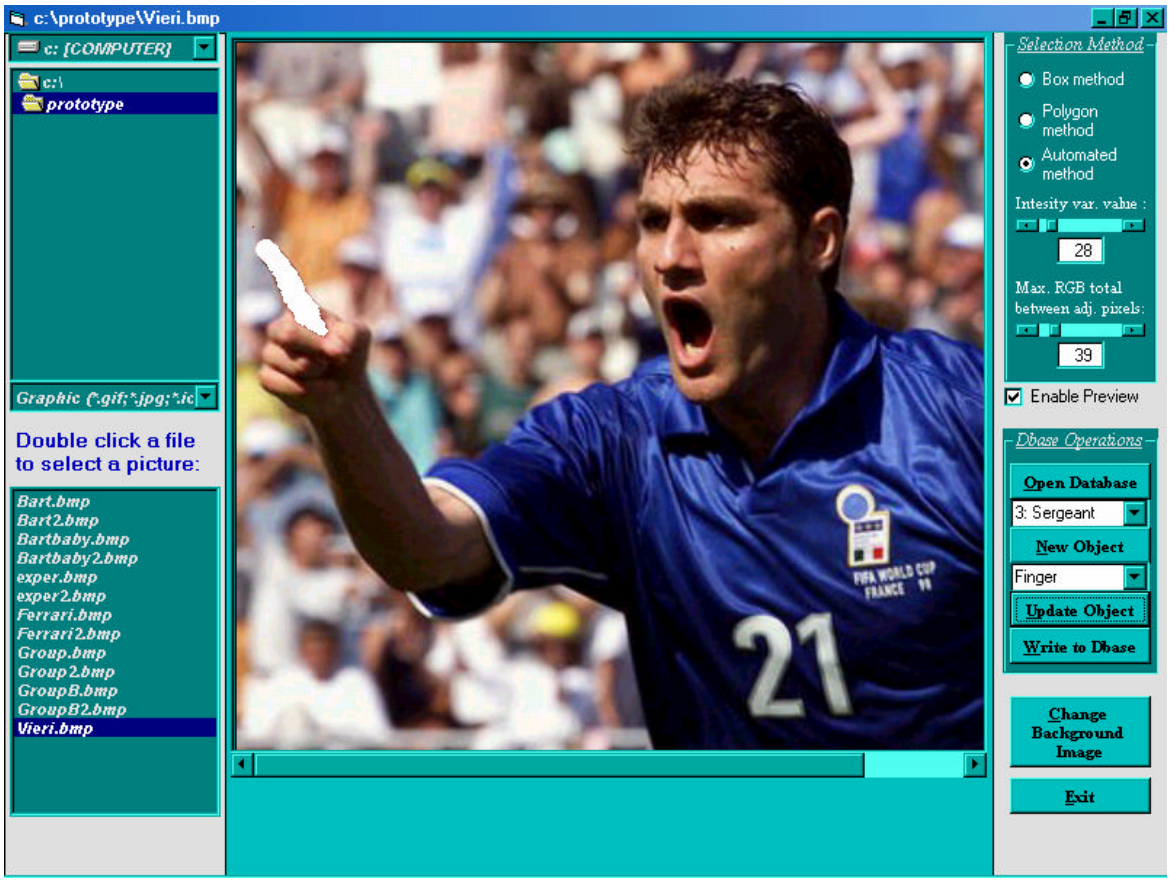

Figure 6.3: A screenshot of the database interface component

Figure 6.3 above shows a screenshot of the database interface component implemented in my prototype. The current image is a photo of Christian Vieri (an Italian soccer international). As can be seen, an image object called "Finger" is created. The region covered by this image object is Vieri's finger, which was selected using the automated method. The intensity variation value for this region

is ‘28’, and the maximum RGB total is ‘39’. The classification assigned to this image object is ‘Sergeant’ (which in this case is classification level 3).

The last task that must be performed when all the objects have been saved to the object data structure is to write all this information to the secure image database. The prototype allows the user to specify where the original and background images are to be stored within the image repository. All the information is then written to the secure image database whose location was specified before any of the objects were created. The unique image identifier containing the original and background images’ paths is written first to the database. All the objects are then written to the ‘Image_Object’ table, with each object containing its own unique identification number, object name, classification, and a reference to the associated image. Finally, each object’s regions are written to the database. The prototype contains at least one separate table for each region selection method, where region specific information is recorded (including object associations). Figure 6.4 illustrates the ER diagram for the prototype (excluding the attributes).

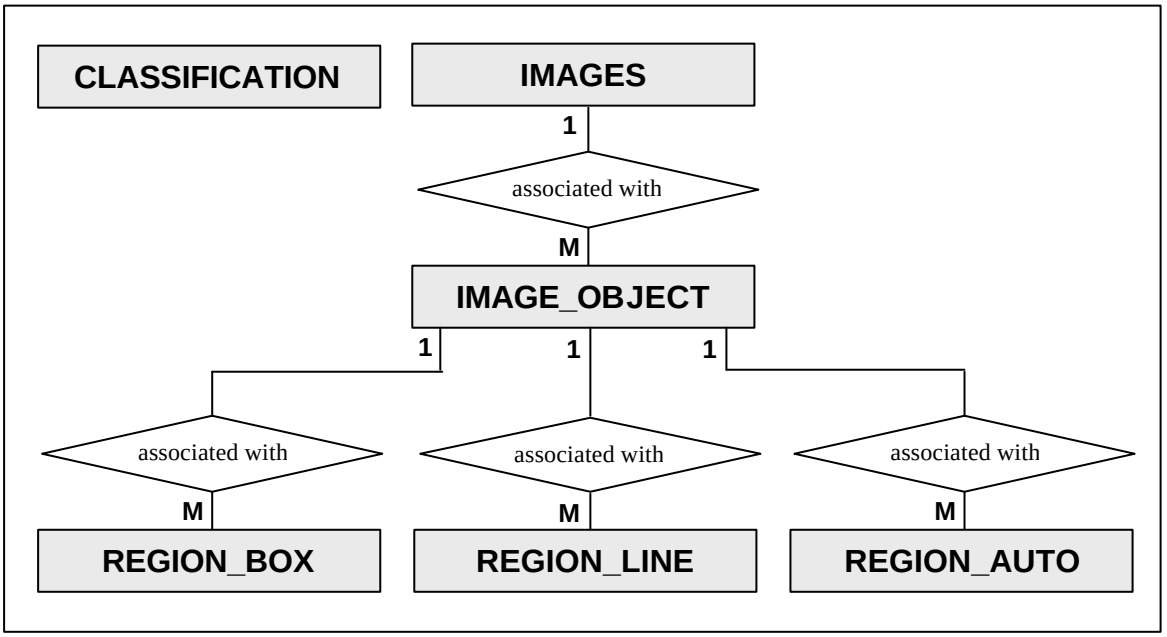

Figure 6.4: ER diagram for the secure image database prototype

The above tasks form part of the database interface and RDBMS activities (phase one and two). The security manager interface component implemented in the prototype is a separate program that allows the user to view how an image is assembled by showing the result of each region selection operation consecutively. The prototype displays the image object information for all objects associated with the requested image that are above the requesting entity’s classification, and also includes which region selection methods were used for each of these objects. The

prototype allows the user to choose which secure image database to open, after which a list of images associated with that image database is displayed. The requesting entity must then choose its classification (for testing purposes), which will be used to assemble an image. Finally, the requesting entity chooses which image must be displayed.

At this point, the SMI receives the original and background images, and uses all relevant information from the secure image database to assemble the image according to the requesting entity's classification. This is done by replacing all regions that are associated with unauthorised image objects (for that image) with the corresponding areas of the background image. It must be noted, as discussed in the above model, that the region selection methods used in the database interface component and the SMI component are consistent with each other (i.e. the same algorithms were used in both components).

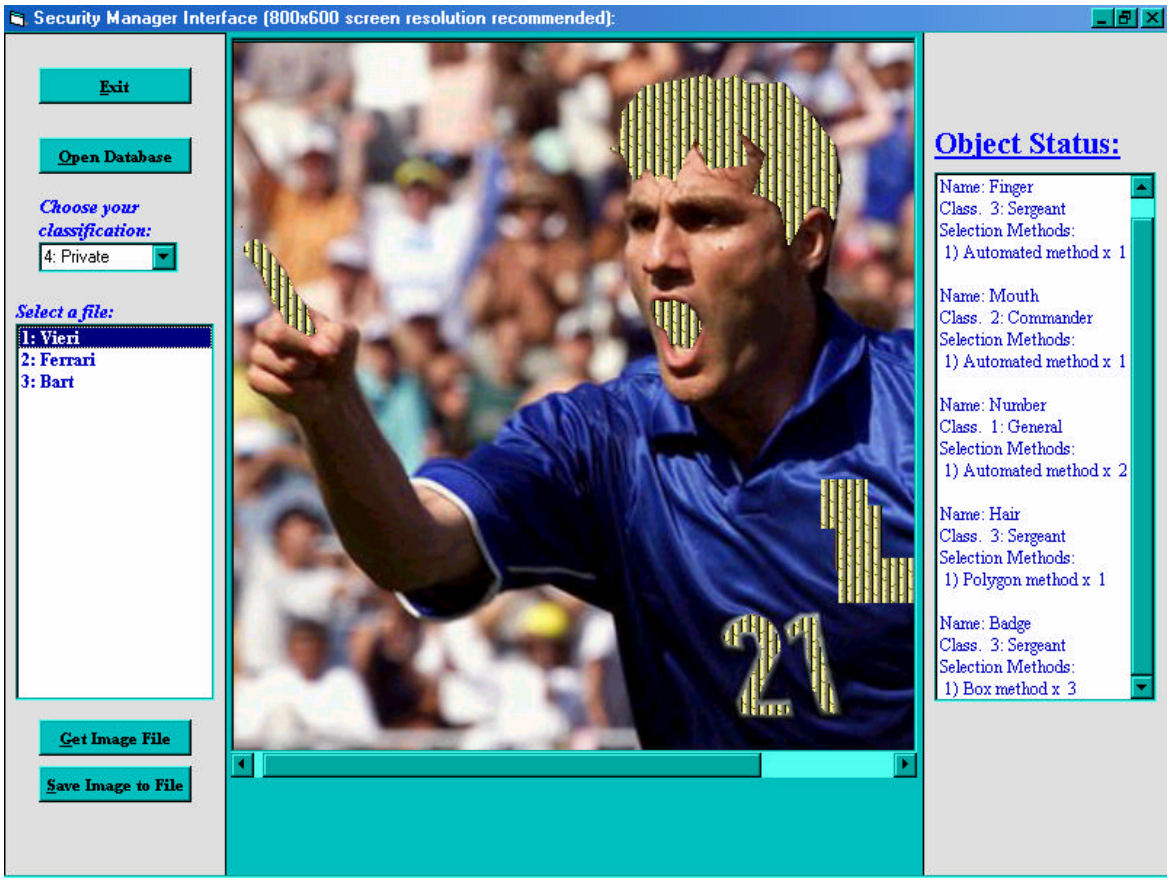

Figure 6.5: A screenshot of the SMI component

Figure 6.5 above shows a screenshot of the SMI component implemented in my prototype. As can be seen, the background image chosen is a repetition of yellow and black stripes. This background image was chosen to make the changes more

visible on paper; a more suitable background image would however normally be used (the next chapter will discuss this further). Five image objects were created altogether. ‘Hair’ was selected using the polygon method, and ‘Badge’ was selected using the box method three times. ‘Finger’, ‘Mouth’, and ‘Number’ were selected using the automated method (the numbers ‘2’ and ‘1’ were selected separately, but both regions were associated with the image object called ‘Number’). The objects ‘Finger’, ‘Hair’, and ‘Badge’ were classified as level 3: Sergeant, ‘Mouth’ was classified as level 2: Commander, and ‘Number’ was classified as level 1: General. In order to produce the above result, all the image objects associated with this image would have to have a higher classification than the requesting entity. The security classification assigned to the requesting entity was therefore 4: Private.

Since the aim of the prototype was not to implement a fully secure application for the secure image database model, several security features were left out of the prototype. The SMI simply lists all the possible classifications without requiring the requesting entity to authenticate itself, for example. A simple numerical scheme was also used to classify for this prototype in order to concentrate on more important issues; naturally, any access control mechanism could be used in a final implementation. A properly implemented application based on the secure image database model must ensure that these security features are incorporated in order for the system to serve its purpose.

6.3.2 Problems and improvements

Important aspects concerning the secure image database model that were discovered during the development of the prototype will now be discussed. Firstly, it does not matter in what order the region selection methods are recorded in the image database or performed in any of the interface components, provided that the region selection operations are all performed on the original image only. In other words, the region selection operations must not at any time be performed on the ‘semi-assembled’ or ‘previewed’ image, since these results will depend on the order in which the operations were performed (because parts of the background image may in this case be included in determining the selected regions). Performing the region selection operations on the ‘semi-assembled’ image would result in a very complex situation where we would have to record the order in which the operations were performed for each specific classification. To avoid unnecessary complexity, the approach taken in the prototype was to perform all operations exclusively on the original image.

The next point concerns the colour models used in the computer systems implementing the secure image database model. The system containing the database interface component (e.g. a server) should have the same colour model as the system containing the SMI component. This is because the region selection methods may come up with inconsistent results when performed on systems with different colour models. A system using the RGB colour model with a 24 bit colour resolution may therefore select different regions (given the same information) as one using the YIQ colour model with another colour resolution.

This unfortunately also means that the colour model should not be changed when making use of the secure image database, since the colour model that was used when classifying an image must be the same as the colour model used when assembling that image (this also applies to the case where the database interface and SMI components are on the same computer). The colour model shouldn't really be a problem in most cases, because the database interface and SMI components should both be in the control of the entity/entities with the highest classification (the requesting entity is not affected by this condition, because only the assembled image is sent to the requesting entity).

A problem concerning region selection still exists, since it is still difficult to accurately select worldly objects for highly detailed images. A large number of automated region selections may need to be performed and associated with an image object before a worldly object can be fully selected. More region selection methods should be added to allow more flexibility when selecting a worldly object. Advanced graphics techniques such as Laplacian edge detection could be combined with the region selection methods to simplify the process of selecting a worldly object. This would obviously require us to record all information relating to these advanced graphic techniques, similar to the way in which we record region specific information. Unfortunately, increasing the amount of graphic manipulation necessary to assemble an image means increasing the power requirements for the computer system(s) containing the database interface and SMI components. Some potential improvements regarding region selection will be further discussed in the next chapter.

A very powerful system is needed to handle multiple requests from requesting entities, with each assembled image requiring many graphic manipulations to be performed. A number of servers may need to be used for multi-processing in order to fulfil the scalability requirement, especially since the amount of graphics processing needed to assemble a requested image varies extremely.

6.4 A variation of the secure image database model

The above secure image database model assembles an image according to the requesting entity's security classification *every time* it is requested. A variation of this model would be to calculate the assembled images *for all possible security classifications* once the image objects for that image have all been finalised (i.e. just before the database interface writes the image objects to the database). All of these images would then be stored in the image repository, allowing the SMI to request the relevant pre-calculated image (depending on the requesting entity's classification) without having to assemble it. Figure 6.6 illustrates the changes that need to be made to phase 1 and 2 of the model. 'Image for C_1' represents the assembled image for security classification level 1, 'Image for C_2' for level 2, and so on up to the lowest classification level C_P.

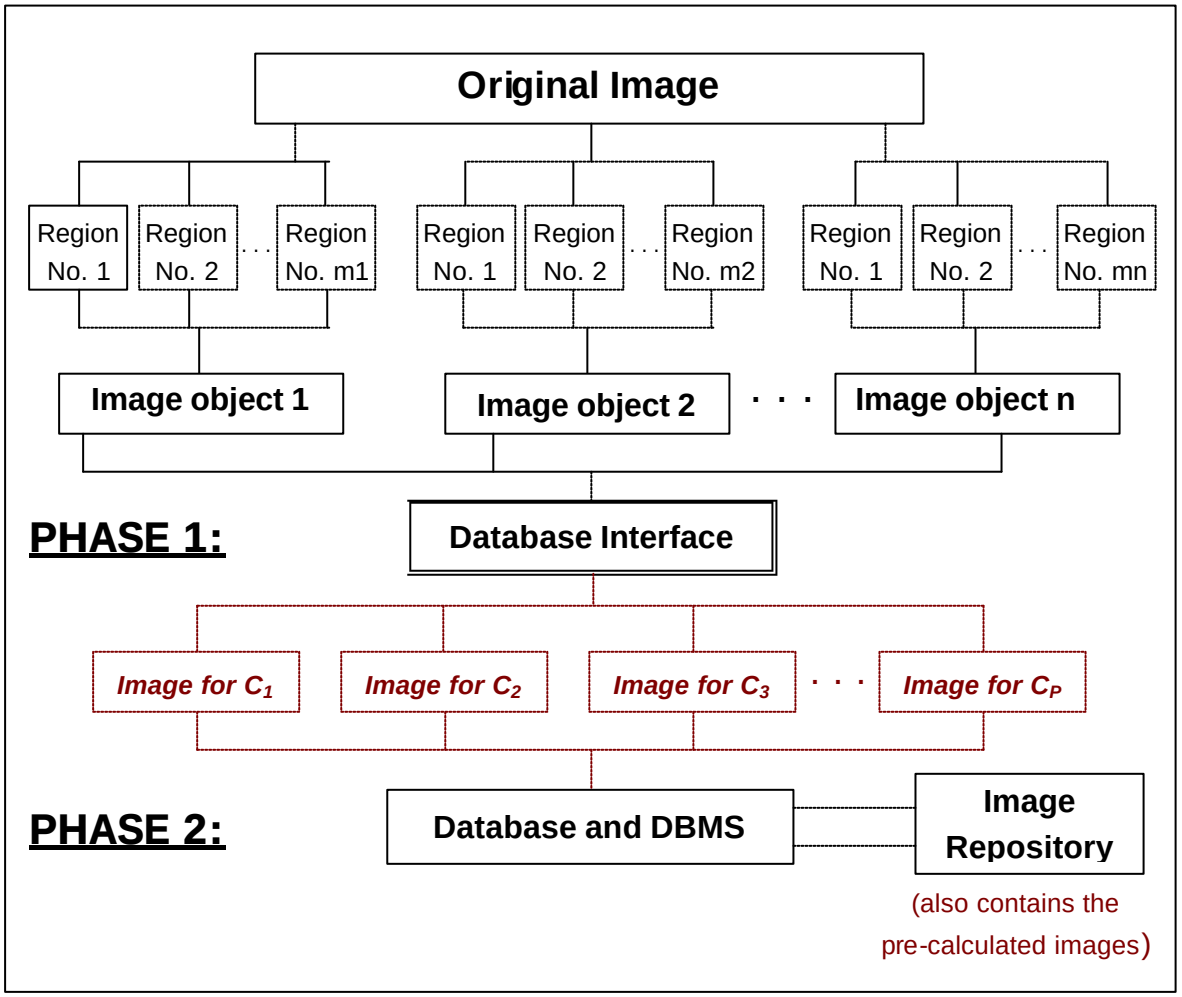

Figure 6.6: Variation of the secure image database model (phase 1 and 2)

An entity will create the image objects and assign regions to them in the same way as before; the entity will not notice any changes made to this model when

creating image objects, selecting regions, defining classifications, etc. The main difference in phase 1 is in the database interface component itself. Once the entity has completed and chooses to write the objects to the secure image database, the database interface assembles the images for each classification. It is therefore the database interface's responsibility to ensure that an image is created for every possible security classification that exists in the security policy (the 'Classification' table in the case of the prototype).

After all the pre-calculated images have been created, they will be written to the image repository. As with the first model, the secure image database will contain the image information needed by the SMI. A unique identifier for each original image will be created. In this model, the paths to each of the pre-calculated images (within the image repository) and their classifications will be associated with the image identifier. The defined image objects and their regions are no longer needed and are not associated with the image identifier; all that is needed is the pre-calculated images and their classifications.

The SMI is still used to make requests for images, although in this variation, the SMI does not need to assemble the image according to the requesting entity's security classification. When an image is needed, the SMI simply requests the RDBMS for the relevant pre-calculated image corresponding to the requesting entity's security classification. The system must therefore still ensure that only the SMI can request the RDBMS for (in this model) pre-calculated images. The SMI assumes that the database interface correctly assembled the image according to that classification. Once the SMI receives the pre-calculated image, it will be transferred to the requesting entity.

This model variation definitely has its advantages. The speed at which the SMI retrieves and transfers the image to the requesting entity is much faster than in the previous model, since the time consuming part of the process is the assembling of the image. The amount of power required by the computer containing the SMI is also greatly decreased, decreasing overall costs required to implement this model. This model variation also has some smaller advantages, e.g. the colour model problem mentioned earlier is solved with this approach, since the database interface is used to select *and* assemble the images.

It should be noted however that any change made to an image object requires the database interface to reassemble all of the pre-calculated images that have an associated classification below that particular image object's classification. This includes any modifications regarding region selection, association, and image

object classification. Image object additions and deletions also require lower classified pre-calculated images to be reassembled. This model isn't therefore suitable for images whose image objects/regions change frequently, especially if requests aren't often made. The previous model is more ideal in this case, since image information may be directly changed, possibly without reselection (because the image objects and region information are in the database).

Both of these models have their advantages and disadvantages. The computer environment will as a result be the deciding factor as to which of these models to use. A combination may also be an option, especially for multimedia applications with different requirements. Generally, the original model may be preferred for cases where image objects change frequently, whereas the model variation may be preferred when efficient retrievals are necessary.

6.5 Conclusion

This chapter introduced a model that can be used to develop a secure multimedia database. The role of each model component was explained together with the relationship between these components. The prototype allowed for a discussion on a more practical level, providing more information needed for a realistic implementation of the model. The model variation offers a similar approach in developing a secure image database aimed at satisfying multimedia applications with a different set of needs.

Certain aspects regarding these models in general such as suspicion, identification and authentication, and confidentiality have not been covered in this chapter. The next chapter will go into more detail on these important issues.