GRAPHICAL PASSWORD AUTHENTICATION SYSTEM

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

By

BIMALENDU GUPTA

ER & DCI NOIDA

IIT Roorkee-ER&DCI, Noida
C-56/1, "Anusandhan Bhawan"
Sector 62, Noida-201 307
FEBRUARY, 2003

Enrolment No. 019014

621.380285
GUP

CANDIDATE'S DECLARATION

This is to certify that the work, which is being presented in this dissertation, entitled "**GRAPHICAL PASSWORD AUTHENTICATION SYSTEM**", in partial fulfillment of the requirements for the award of the degree of **Master of Technology** in **Information Technology** submitted in **IIT, Roorkee – ER&DCI Campus, Noida,** is an authentic record of my own work carried out from August 2002 to February 2003, under the supervision of **Dr. P.R. Gupta**, Reader, Electronics Research and Development Centre of India, Noida.

I have not submitted the matter embodied in this dissertation for the award of any other degree.

Date: 21/02/2003

Place: Noida

[illegible]

(Bimalendu Gupta)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 21.2.2003

Place: Noida

[illegible]

(Dr. P.R. Gupta)

Reader,

ER&DCI, Noida

[illegible]

ACKNOWLEDGEMENT

I would like to thank **Prof. Prem Vrath**, Director, IIT-Roorkee, **Mr. R.K. Verma**, Executive Director, ER&DCI, Noida, **Dr. A.K. Awasthi**, Dean, Post Graduate Studies and Research and **Dr. R.P. Agarwal**, Course Coordinator for giving me this valuable opportunity to pursue this work.

I would like to thank my guide, **Dr. P. R. Gupta**, Reader, ER&DCI, Noida for her patience and encouragement. She has been an invaluable source of support and guidance all through this dissertation.

I am grateful to **Mr. V. N. Shukla,** Course Coordinator for providing the best of facilities to carry out the dissertation. I am also grateful to **Mr. Munish Kumar** for the cooperation extended by him in the successful completion of this report.

Most of all I would like to thank my family. My parents provided me a perfect environment for my studies and supported me throughout. Finally, I would like to extend my gratitude to all those persons who directly or indirectly helped me in the process and contributed towards this work.

[illegible]

(Bimalendu Gupta)

019014

Contents

ABSTRACT

Authentication is necessary for providing access to required information to the authorized user. User authentication has become an important and central component of present day security measures.

In the present thesis, different authentication systems in use are analyzed with reference to their merits and demerits. Thrust is on the knowledge based authentication systems as most of the security systems are based on the concept of knowledge based authentication.

A case of UNIX operating system is included, as this operating system uses knowledge based authentication and is considered one of the most secure systems. Various arrangements and developments in this operating system for authentication, since its inception, are discussed in detail. Further the weaknesses and drawbacks of the knowledge based authentication system are explored.

Image/graphics based password can offer an alternative of text based password because of their inherent properties such as large volume of information. A change in image parameter generates exponentially large sets of possible password combinations, to be guessed.

A solution using graphical password authentication system is proposed based on the analysis. The solution proposed suggests how the information contained in the images, which is unique for different images, could be very well used for generating and retrieving a secure, unique and memorable passwords.

CHAPTER 1

INTRODUCTION

1.1 Background

Authentication is done to verify whether the person who is trying to have a access to the system is right one or not and whether he or she should be allowed to access the system or not.

User authentication is very important and central component of currently deployed security infrastructures. Whether user is trying to log on to their personal systems or they are trying to check their mails, they need to prove their identity to the system for being an authenticated user. The job of verification of the users in any security systems is done by the set of programs specially written for authentication. If authentication mechanisms deployed are categorized, from past till now, three main techniques could be distinguished which are deployed for user authentication: Knowledge-based systems, token-based systems, and systems based on biometrics [2].

Knowledge based systems are those systems which require user name and password combination for the authentication of the user. One comes across such systems, while logging on to personal system or trying to check mailboxes, in all such cases one has to provide user name and password to the system for accessing information or other resources.

Token based systems are the systems which requires token apart from the PINs or password, both token and password/PIN are required for authentication. Token is a physical mean for carrying secret information, it may be a card with magnetic tape containing data. This kind of authentication is used in the ATM machines and credit cards etc.

Biometrics are automated methods of recognizing a person based on physiological or behavioral characteristics.

In today's security systems, knowledge-based schemes are predominantly used for user authentication. Although biometrics can be useful for user identification, one problem with these systems is the difficult tradeoff between impostor pass rate and false alarm rate. In addition, many biometric systems require specialized devices, and some can be unpleasant to use.

Most token-based authentication systems also use knowledge-based authentication to prevent impersonation through theft or loss of the token. Like in ATM authentication, it requires a combination of a token (a bank card) and secret knowledge (a PIN).

For the reasons stated above, knowledge based techniques are currently the most frequently used method for user authentication.

Despite their wide usage, knowledge based authentication systems have number of shortcomings mainly concerning passwords and PINs. It could be easily observed that simple or meaningful passwords are easier to remember, but are vulnerable to attack. Passwords that are complex and arbitrary are more secure, but are difficult to remember. Since users can only remember a limited number of passwords, they tend to write down or use similar or even identical passwords for different purposes. Users generally tend to share their passwords with others to share some information, they take it as a feature rather than a risk. This behavior of the user may lead to the penetration of the system even if that system is using most secure encryption algorithms for securing passwords.

One approach to improve user authentication is to replace the precise recall of a password or PIN with the recognition of a previously seen image, a skill at which humans are remarkably proficient. In general, it is much easier to recognize something than to recall the same information from memory without help. Humans have a vast, almost limitless memory for pictures in particular. In fact, human visual perception is so good that user can remember and recognize hundreds to thousands of pictures in fractions of a second of perception. By replacing precise recall of the password with image recognition, one can minimize the user's cognitive load, help the user to make fewer mistakes and provide a more pleasant experience [2].

1.2 Problem Description

All the knowledge based authentication systems are dependent totally on the secret information, i.e. the password, which in turn again depends on the user. If the user is aware of the requirements of the good passwords and only if the user is satisfying them then only it can be said that it is hard to break the system or else the system could be easily penetrated.

It has been observed that even when the user is aware of the requirements of good passwords user generally tend to choose memorable passwords. As aid to their memory, users generally choose simple dictionary words, may be their first names, their telephone number, house number, family name and many other things, which are very easy to guess. It's a general tendency of the user to prefer convenience over security, which leads to the selection of passwords which are very weak.

If anyone tries to find out the root cause of the weaknesses that are existing in the currently deployed knowledge based authentication systems, one will see that the main factor, which in turn leads to the penetration of the system by a malicious user is limitation of human mind in precise recall of the information. Human mind is not good at precise recall of any information and in all the knowledge based authentication systems precise recall of the password is required. Moreover one very important point is that problem of precise recall directly conflicts with the requirements of a good password.

This thesis is an attempt to design an authentication system using images for generating and retrieving very secure and memorable passwords, which will remove most of the weaknesses that are associated with the current knowledge based authentication systems.

The system designed will meet the following requirements:

- It will heavily reduce the problem of precise recall of the passwords.
- It will generate very secure passwords. Passwords generated will meet all the requirements of a good password.
- It becomes very difficult for the user to write down the passwords anywhere or to share their passwords with anyone.
- Brute force attack, which is very common for breaking up the systems that rely on passwords, becomes computationally infeasible.
- Could be easily implemented and incorporated in the existing systems.

1.3 Scope

Although there is a great development in the area of biometrics still most of the security infrastructures in the current environment are using either knowledge based authentication systems or token based authentication systems. These systems are just software implementations and require very minimal hardware to operate which is quite cheaper. Therefore these systems are very prominent these days. Both knowledge based as well as token based systems rely on the secret information which is the password. If it is weak the whole system will be penetrated.

Graphical password authentication system will be reducing the possibilities of breaking up of the system by removing most of the weaknesses of the currently deployed authentication systems. The system designed could be well incorporated with the operating systems currently used, as it does not require any special or costly hardware it can embed as software. Microsoft researchers are working in the area of graphical password authentication systems and they are thinking about incorporating it in their operating system in the coming future [4].

Moreover this kind of authentication system will be very useful if it is used with the mobile devices which support displaying images and input to the device is given with the help of stylus.

1.4 Organization of the thesis

In chapter 2 basic details have been given for one way functions and few color models. Case study for UNIX is presented to discuss that how knowledge based authentication system works. Later on in this chapter weaknesses of knowledge based authentication system have been discussed. In chapter 3 analysis of the problem is done and a solution is proposed. Solution proposed is analyzed for various kinds of attacks. Chapter 4 deals with the design of the system different phases and architecture is discussed. In chapter 5 output of the system is shown with the help of screen shots. Results of the experiment conducted are also given in this chapter. Chapter 6 includes conclusion, problems associated, few recommendations and future work. In the last references are included for the further reading.

CHAPTER 2
LITERATURE SURVEY

2.1 One way hash functions

A one way hash function, H(*M*), operates on an arbitrary length message *M*. It returns a fix length hash value, *h*.

h=H(M), where h is of length m.

Many functions can take arbitrary length input and return an output of fixed length, but one way hash functions have additional characteristics that make them one way:

Given *M*, it is easy to compute *h*.

Given *h*, it is hard to compute *M* such that $H(M)=h$.

Given *M*, it is hard to find another message, *M'*, such that H(M)=H(M').

The whole point of one way hash function is to provide a fingerprint of *M* which is unique.

In the real world, one way hash functions are built on the idea of a compression function [7]. This one way function outputs a hash value of length n given an input of some larger length m. The inputs to the compression function are a message block and the outputs of the previous blocks of the text. The output is the hash of all blocks up to that point. That is, the hash of block M_i is

$$H_i=f(M_i,h_{i-1})$$

This hash value along with the next message block becomes the next input to the compression function. The hash of the entire message is the hash of the last block.

There are many standard one way hash functions each having few advantages and disadvantages, table 2.5.1 below gives the details of few one way hash algorithms in terms of hash length produced and speed at which they execute. Data has been recorded on a 33Mhz 486 machine [7].

Algorithm	Hash Length	Encryption speed (Kilobytes/Second)
Abreast Davies-Meyer (with IDEA)	128	22
Davies-Meyer (with DES)	64	9
GOST hash	256	11
HAVAL (3 passes)	Variable	168
HAVAL (4 passes)	Variable	118
HAVAL (5 passes)	Variable	95
MD2	128	23
MD4	128	236
MD5	128	174
N-HASH (12 rounds)	128	29
N-HASH (15 rounds)	128	24
RIPE-MD	128	182
SHA	160	75
SNEFRU (4 passes)	128	48
SNEFRU (8 passes)	128	23

Table 2.5.1 shows details of the various one way hash functions in terms of hash value and encryption speed.

Depending on the situation and the environment in which the system is to be used the appropriate algorithm could be chosen.

2.2 Color Models

There are many ways to represent a color numerically. A system for representing colors is called color model. Color models are usually designed to take advantage of a particular type of display device.

On most color monitors there are three phosphors (red, green, and blue), or light emitters for each pixel. Adjusting the intensity of individual phosphors controls the color of the pixel. When all three phosphors are at their minimum intensity phosphor appears black. At their maximum intensity the pixel appears white. If the red phosphor is the only one active, the pixel appears red. When the red and green phosphors are on they combine to produce the shades of yellow, and when all three phosphors are at their full intensity the pixel appears white.

2.2.1 RGB color model

The most common color model used in computer applications is known as RGB (Red-Green-Blue). The RGB model mimics the operation of computer displays. In RGB, colors are composed of three component values that represent the relative intensities of red, green, and blue. Relationship of colors can be represented by RGB color model as shown in figure 2.6.1. The range of colors that can be represented by a color model is known as color space. In the figure the cube represents the RGB color space [14].

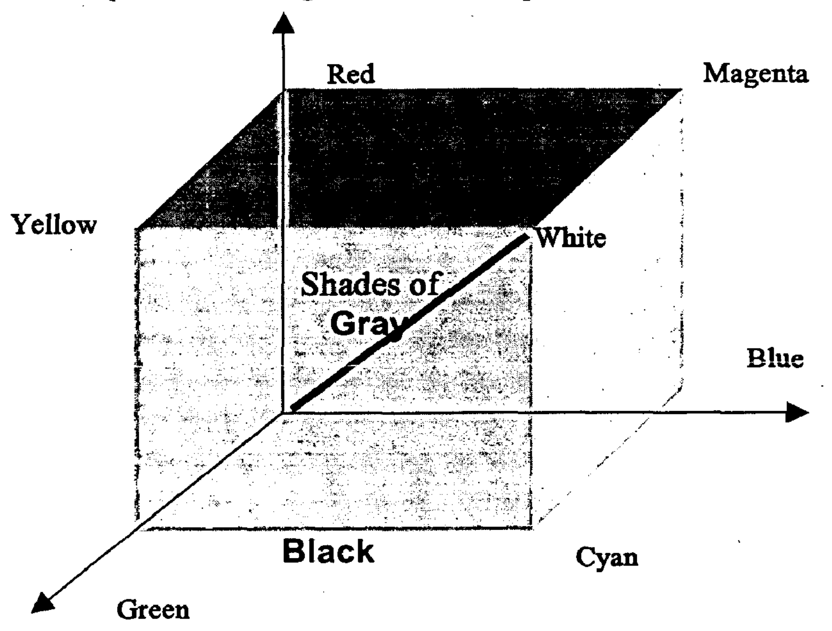


Figure 2.6.1 shows the RGB color space

In programming and image formats, unsigned integer component values are almost always used. The range of values for a color component is determined by the sample precision, which is the number of bits used to represent a component. Integer component values can range from 0 to $2^{\text{sample precision}} - 1$. For sample precision of 8 bits, 256 different shades of primary colors can be represented.

2.2.2 YCbCr color model

RGB is not the only color model in use. At one time the HSB (Hue-Saturation-Brightness) color model was commonly used in computer systems and still is used by some image processing applications. JPEG images are almost always stored using three-component color space known as YcbCr. The Y, or luminance, component represents the intensity of the image. Cb and Cr are the chrominance components. Cb specifies the blueness of the image and Cr gives the redness.

The YcbCr color model is similar to the one used in television sets that allows color images to be compatible with black and white sets. In the YcbCr color model, the Y component on its own is a grayscale representation of the color image.

The relation between the YcbCr and RGB models as used in JPEG is represented by

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = -0.1687R - 0.3313G + 0.5B + 2^{\text{Sample Precision/2}}$$
$$CR = 0.5R - 0.4187G - 0.0813B + 2^{\text{Sample Precision/2}}$$

$$R = Y + 1.402Cr$$
$$G = Y - 0.34414(Cb - 2^{\text{Sample Precision/2}}) - 0.71414(Cr - 2^{\text{Sample Precision/2}})$$
$$B = Y + 1.722(Cb - 2^{\text{Sample Precision/2}})$$

We can easily see from the equation that Y component contributes the most information to the image [14].

2.3 Authentication system in UNIX a case study and related work done for improvement of authentication systems

Since authentication system of UNIX operating system uses most of the features of the good authentication system and is considered to be one of the most secure systems to break in, we would be taking UNIX operating system to learn about various historical development in the area of knowledge based authentication systems. We will also see what improvements were incorporated in the system with time. Finally we would be looking at the weaknesses still associated with the system.

After the study of the weaknesses associated with the authentication system used in UNIX related work done in the area of improving authentication mechanisms will be presented, which mainly includes the work, done for using images for authentication.

2.4 Initial implementation of authentication system in UNIX

The UNIX system was first implemented with a password file that contained the actual passwords of all the users, and for that reason the password file had to be heavily protected against being either read or written [6]. Although historically, this had been the technique used for remote access systems, it was completely unsatisfactory for several reasons. The technique is excessively vulnerable to lapses in security.

Temporary loss of protection can occur when the password file is being edited or otherwise modified. There is no way to prevent the making of copies by privileged users. Experience with several earlier remote access systems showed that such lapses occur with frightening frequency.

For an instance when a system administrator at MIT was editing the password file and another system administrator was editing the daily message that is printed on everyone's terminal on login. Due to a software design error, the temporary editor files of the two users were interchanged and thus, for a time, the password file was printed on every terminal when it was logged in [6].

Once such a lapse in security has been discovered, everyone's password must be changed, usually simultaneously, at a considerable administrative cost. This is not a great matter, but far more serious is the high probability of such lapses going unnoticed by the system administrators. Security against unauthorized disclosure of the passwords is impossible with this system because, for example, if the contents of the file system are put on to magnetic tape for backup, as they must be, then anyone who has physical access to the tape can read anything on it with no restriction. Many programs must get information of various kinds about the users of the system, and these programs in general should have no special permission to read the password file. The information that should have been in the password file actually was distributed (or replicated) into a number of files, all of which had to be updated whenever a user was added to or dropped from the system. Overall this initial approach for authentication was very insecure and hence some improvement over this approach was badly needed.

2.4.1 Improvements over this initial approach

The obvious solution that comes into picture is to arrange that the passwords not appear in the system at all. It is not difficult to decide that this can be done by encrypting each user's password, putting only the encrypted form in the password file, and throwing away his original password (the one that he typed in). When the user later tries to log in to the system, the password that he types is encrypted and compared with the encrypted version in the password file. If the two matches, his login attempt is accepted else that login fails.

It also seemed advisable to devise a system in which neither the password file nor the password programs itself needs to be protected against being read by anyone. All that was needed to implement these ideas was to find a means of encryption that was very difficult to invert, even when the encryption program is available. This assures that even if the malicious person gets the program or even the password file it becomes impossible for him to crack the system.

To implement this approach crypt was adopted for UNIX authentication system. Passwords are encrypted using this algorithm and are kept in publicly readable file /etc/passwd.

Crypt uses the resistance of DES to known plain text attack to make it computationally infeasible to determine the original password that produced a given encrypted password by exhaustive search [1]. The only publicly known technique that may reveal certain passwords is password guessing that is passing large word lists through crypt function to see if any match the encrypted password entries in an /etc/passwd file. In this system password was not used as the text to be encrypted but as the key, and a constant string of zeros was encrypted using this key. The encrypted result was stored in the file.

2.4.2 Attack on the approach taken for initial improvements

Suppose that the hacker has available the text of the password encryption program and the complete password file. Suppose also that he has substantial computing capacity at his disposal. One obvious approach to penetrating the password mechanism is to attempt to find a general method of inverting the encryption algorithm. Very possibly this can be done, but few successful results have come to light, despite substantial efforts extending over a period of more than five years [6]. The results have not proved to be very useful in penetrating systems.

Another approach to penetration is simply to keep trying potential passwords until one succeeds. This is a general cryptanalytic approach called key search. Human beings being what they are, there is a strong tendency for people to choose relatively short and simple passwords that they can remember. Given free choice, most people will choose their passwords from a restricted character set (e.g. all lower-case letters), and will often choose words or names. This human habit makes the key search job a great deal easier.

The critical factor involved in key search is the amount of time needed to encrypt a potential password and to check the result against an entry in the password file. The running time to encrypt one trial password and check the result turned out to be approximately 1.25 milliseconds on a PDP-11/70 when the encryption algorithm was recorded for maximum speed [6]. It is takes essentially no more time to test the encrypted trial password against all the passwords in an entire password file, or for that matter, against any collection of encrypted passwords, perhaps collected from many installations. If somebody want to check all passwords of length n that consist entirely of lower-case letters, the number of such passwords is 26^n. If we suppose that the password consists of printable characters only, then the number of possible passwords is somewhat less than 95^n. (The standard system ''character erase'' and ''line kill'' characters are, for example, not prime candidates.) One can immediately estimate the running time of a program that will test every password of a given length with all of its characters chosen from some set of characters. The table 2.2.2.1 gives estimates of the running time required on a PDP-11/70 to test all possible character strings of length n chosen from various sets of characters namely, all lower-case letters, all lowercase letters plus digits, all alphanumeric characters, all 95 printable ASCII characters, and finally all 128 ASCII characters [6].

N	26Lowercase letters	36Lowercase letters&digits	62 alphanumeric characters	95 printable characters	All 128 ASCII Characters
1	30 msec	40 msec	80 msec	120msec	160msec
2	800 msec	2 sec	5 sec	11 sec	20 sec
3	22 sec	58 sec	5 min	17min	43 min
4	10 min	35 min	5 hrs	28 hrs	93hrs
5	4 hrs	21 hrs	318 hrs		
6	107 hrs				

Table. 2.2.2.1 Time taken for comparing all character combinations.

One has to conclude that it is no great matter for someone with access to a PDP-11 to test all lower-case alphabetic strings up to length five and, given access to the machine for, say, several weekends, to test all such strings up to six characters in length [Table 2.2.2.1]. By using such a program against a collection of actual encrypted passwords, a substantial fraction of all the passwords will be found.

Another profitable approach for the hacker is to use the word list from a dictionary or to use a list of names. For example, a large commercial dictionary contains typically about 250,000 words these words can be checked in about five minutes. Again, a noticeable fraction of any collection of passwords will be found. Improvements and extensions will be (and have been) found by a determined hacker. Some good things to try are:

- The dictionary with the words spelled backwards.
- A list of first names (best obtained from some mailing list). Last names, street names, and city names also work well.
- The above with initial uppercase letters.
- All valid license plate numbers in your city.
- Room numbers, social security numbers, telephone numbers, and the like.

So one can easily see that if the password chosen by the user belongs to any one of the above-mentioned categories there is a good chance that it would be broken.

2.5 Further improvements in the approach

2.5.1 Slower Encryption

Obviously, the first algorithm used was far too fast. The announcement of the DES encryption algorithm by the National Bureau of Standards was timely and fortunate. The DES is, by design, hard to invert, but equally valuable is the fact that it is extremely slow when implemented in software. The DES was implemented and used in the following way:

The first eight characters of the user's password are used as a key for the DES, then the algorithm is used to encrypt a constant. Although this constant is zero at the moment, it is easily accessible and can be made installation-dependent. Then the DES algorithm is iterated 25 times and the resulting 64 bits are repacked to become a string of 11 printable characters. Since the algorithm was very slow in the software implementation therefore it posed a requirement of good computing power for anyone who is trying to break the system using the technique of key search. But if someone is having a good computation power at disposal then he or she would be able to make a try to break the system.

2.5.2 Less Predictable Passwords

The password entry program was modified so as to urge the user to use more obscure passwords. If the user enters an alphabetic password (all upper-case or all lower-case) shorter than six characters, or a password from a larger character set shorter than five characters, then the program asks him to enter a longer password. This further reduces the efficacy of key search [6].

These improvements make it exceedingly difficult to find any individual password. The user is warned of the risks and if he cooperates, he is very safe indeed. On the other hand, he is not prevented from using his spouse's name if he wants to.

2.5.3 Salted Passwords

The key search technique is still likely to turn up a few passwords when it is used on a large collection of passwords, and it seemed wise to make this task as difficult as possible. To this end, when a password is first entered, the password program obtains a 12-bit random number (by reading the real-time clock) and appends this to the password typed in by the user. The concatenated string is encrypted and both the 12-bit random quantity (called the salt) and the 64-bit result of the encryption are entered into the password file.

When the user later logs in to the system, the 12-bit quantity is extracted from the password file and appended to the typed password. The encrypted result is required, as before, to be the same as the remaining 64 bits in the password file. This modification does not increase the task of finding any individual password, starting from scratch, but now the work of testing a given character string against a large collection of encrypted passwords has been multiplied by 4096 (2^{12}). The reason for this is that there are 4096 encrypted versions of each password and one of them has been picked more or less at random by the system.

With this modification, it is likely that the malicious user can spend days of computer time trying to find a password on a system with hundreds of passwords, and find none at all. More important is the fact that it becomes impractical to prepare an encrypted dictionary in advance. Such an encrypted dictionary could be used to crack new passwords in milliseconds when they appear.

There is a (not inadvertent) side effect of this modification. It becomes nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them, unless you already know that [3].

2.5.4 The Threat of the DES Chip

Chips to perform the DES encryption are already commercially available and they are very fast. The use of such a chip speeds up the process of password hunting by three orders of magnitude. To avert this possibility, one of the internal tables of the DES algorithm (in particular, the so-called E-table) is changed in a way that depends on the

12-bit random number. The E-table is inseparably wired into the DES chip, so that the commercial chip cannot be used. Obviously, the hacker could have his own chip designed and built, but the cost would be unthinkable. Since the DES chip cannot be incorporated everywhere it becomes necessary to find a solution, which could be, incorporated everywhere very easily.

2.5.5 Shadow Passwords

One method of restricting availability of the encrypted passwords is the shadow password file. A shadow password file is a file that contains the encrypted passwords and is not publicly accessible. This prevents the average user from accessing the password file. One problem with the approach is that a system administrator has access to the file. This is not the problem while the person remains an administrator, since presumably he has access to everything anyway. The problem is when a system administrator leaves. He may have copied encrypted passwords while he was an administrator and can now use these passwords to infiltrate the system. Also, if an error in access permission is made, the encrypted password file may become available to all the users of a system. Shadow password file cannot hurt, but it seems unwise to count on them alone for system security [3].

We can easily see from the above study that although so many arrangements have been made in the UNIX operating system, like various arrangements for secure password file as well as use of complex encryption algorithms for the proper authentication it is still vulnerable. One careful observation one can find the root cause as the imprecise recall of the passwords by the user and few other factors. Next section describes work done in the area of authentication using images.

2.6 Related work for improving user authentication

To remove all the weaknesses of knowledge based authentication systems many researches are going on. All these are mainly concentrating on replacing text based passwords with the graphical images.

Passlogix Inc. distributes v-go, an application, which remembers user names and passwords and automatically logs the user on to password-protected Web sites and applications [8]. They allow users to create passwords by clicking on objects in a graphical window, such as by entering the time on a clock, drawing cards from a card deck, selecting ingredients to mix a cocktail or to cook a meal, dialing a phone number, hiding objects in a room, trading stocks, and entering a password on a keyboard.

The weaknesses of their system are many folds. First, the space of different passwords is very small. For example, there are only limited places available to select to cook a meal. In the case of hiding objects in a room, the requirement to hide the objects already strongly reduces the state space. It would be better if the user could place objects in arbitrary locations. There are only a few places in the given room where the objects can really be hidden, for example under the mattress or the cabinets are locations which users are likely to select. Furthermore, the system allows users to pick poor passwords. For example, it is likely that many users will choose commonly known combinations, for example by choosing to mix the same drinks.

IDArts distributes Passfaces, an authentication system based on recognizing previously seen images of faces [9]. This idea is also based on the image recognition, and there is strong evidence to support their claim that humans have an innate ability to remember faces. They claim that authentication rates can be significantly improved by training the user during passface creation. A drawback of their system is that users pick faces which they are attracted to, which greatly facilitates impersonation attacks.

Ellison propose a scheme in which a user can protect a secret key using "the personal entropy in his whole life", that is by encrypting the pass phrase using the answers to several personal questions [11]. The scheme is designed so that a user can forget the answers to a subset of the questions and still recover the secret key, while an attacker must learn the answer to a large subset of the questions to learn the key.

Naor and Shamir propose a Visual Cryptography scheme, which splits secret information into two transparencies, such that each part contains no useful information, but the combination reveals the secret [12].

Naor and Pinkas extend this idea as a means for a user to authenticate text and images. In this case, the recipient is equipped with a transparency [11]. When the recipient places the transparency over a message or image that was sent to him, the combination of both images reveals the message. Visual cryptography could be used to devise a user authentication scheme that is token based. Images could be divided by dividing each pixel into number of sub-pixels and each set of sub-pixels will represent a sub-image as shown in the figure 2.4.1

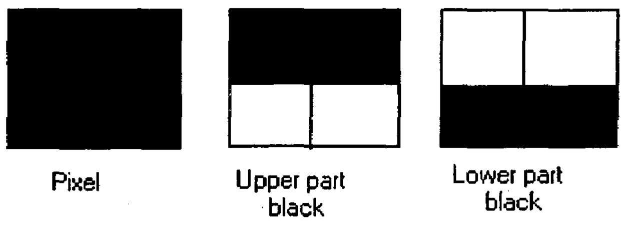


Figure 2.4.1 One pixel divided into two sub-pixels

The two sub-images formed from the pixels would be consisting of upper and lower part of the original pixel.

Rachna Dhamija and Adrian Perrig are developing an authentication system based on images. In their system they present a set of about 200 images to the user out of which the user is prompted to choose few images as a password as shown in figure 2.4.2

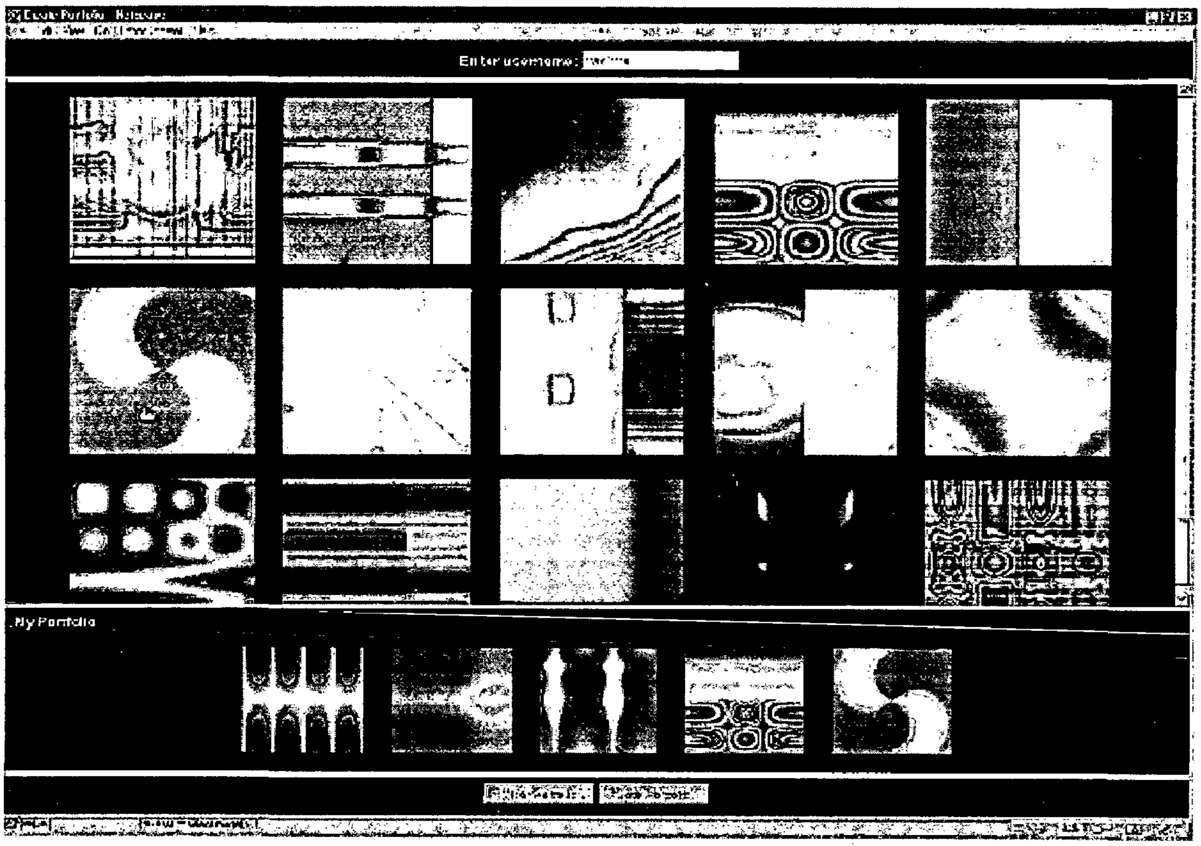


Figure 2.4.2 Screen shot of "Déjà vu" the authentication system based on images

This set of images chosen by the user acts as a portfolio for that particular user. When the user tries to log on to the system next time he or she is presented with the set of images that contains the images used for making portfolio as some decoy images. If the user is able to recognize the correct set of images, which constitute the portfolio of that particular user, he or she is allowed to log on else not [2].

Adams and Sasse propose that educating users in security is a solution for the problem of choosing weak passwords. They claim that if users receive specific security training and understand security models, they will select secure passwords and refrain from engaging in insecure behavior [13]. However, the level of security training did not prevent users from choosing trivial passwords or from storing them insecurely. This is the case because people prefer convenience over security. Therefore, security should be an inherent component of the system by default.

In next chapter summary of all the shortcomings is given which is observed from the study of authentication system in UNIX.

CHAPTER 3

ANALYSIS

In this section the general shortcomings of password based authentication have been summarized, which is observed from the study of authentication system used in UNIX operating system and various other related works done in the area of designing good authentication systems.

3.1 Shortcomings of Password-Based Authentication

Password and PIN-based user authentication have numerous deficiencies. Unfortunately, many security systems are designed such that security relies entirely on a secret password. Many researches have shown that weak passwords are the most common cause for system break-ins.

The main weakness of knowledge based authentication is that it relies on precise recall of the secret information. If the user makes a small error in entering the secret, the authentication fails. Unfortunately, precise recall is not a strong point of human cognition. People are much better at imprecise recall, particularly in recognition of previously experienced stimuli.

The human limitation of precise recall is in direct conflict with the requirements of strong passwords. Since the good password requirements says that the password chosen should be at least 8 characters long also it should must consists of alphanumeric characters, it should not be a dictionary word, it should not be somebody's name and many other things. If user will follow the guidelines for choosing good passwords then it will become difficult for the user to recall it back. So people tend to choose the passwords, which they can recall easily, means they go for common words or names. Many researchers show that people pick easy to guess passwords. For example, study shows that over 15% of users picked passwords shorter or equal to three characters. Furthermore, it is found that 85% of all passwords could be trivially broken through a simple exhaustive search to find short passwords and by using a dictionary to find longer ones. Study conducted on password security shows that 25% of all passwords can be broken with a small dictionary [5].

Other studies conducted to design password crackers, because of these password cracker programs, users need to create unpredictable passwords, which are more difficult to memorize. As a result, users often write their passwords down and hide them close to their workspace. Strict password policies, such as forcing users to change passwords periodically, only increase the number of users who write them down to aid memory [5].

As companies try to increase the security of their IT infrastructure, the number of password-protected areas is growing. Simultaneously, the number of Internet sites, which require a username and password combination, is also increasing. To cope with this, users employ similar or identical passwords for different purposes, which reduce the security of the password to that of the weakest link.

Another problem with passwords is that they are easy to write down and to share with others. Some users have no qualms about revealing their passwords to others they view this as a feature and not as a risk.

The majority of solutions to the problems of weak passwords fall into three main categories.

The first types of solutions are proactive security measures that aim to identify weak passwords before they are broken by constantly running a password cracking programs. In this kind of solution the system administrator continually checks for weak passwords by running password cracking programs, these programs use dictionary of words to find out easy-to-guess passwords, these programs often make the use of the word lists containing the common names of the people, city etc. as soon as the password is broken the user is prompted to change the password by the administrator.

The second types of solutions are also technical in nature, which utilizes techniques to increase the computational overhead of cracking passwords. This solution is the most common solution which most of the security systems are deploying. Very complex encryption algorithms are used which supports a very large key value as well as the algorithm is very hard to invert. If passwords are encrypted using these algorithms it is almost impossible to get the passwords by searching or by using any such technique, as the computational power required to get the result in time will be too large to harness.

The third types of solutions involve user training and education to raise security awareness and establishing security guidelines and rules for users to follow. It is observed that even after proper training people tend to choose the passwords, which seem convenient to them as good passwords are hard to remember and it is the general tendency of most of the people prefer convenience over security.

It was observed that all the three classes of solutions do not remedy the main cause of password insecurity, which is the human limitation of memory for secure passwords. If the user has no problem for memory then the user will be choosing good passwords. In fact, most previously proposed schemes for knowledge based user authentication rely on perfect memorization.

3.2. Solution of the problem

After analyzing the authentication systems in detail it is analyzed where the actual problem lies and it should be taken care that in general the system should not rely totally on precise recall. Instead, it should be based on recognition to:

- Make the authentication task more reliable and easier for the user.
- The system should generate good passwords that are not easy to break.
- The system should make it difficult to write passwords down and to share them with others.

Current secure systems suffer because they neglect the importance of human factors in security. Fundamental weakness of knowledge based authentication schemes is addressed, which is the human limitation to remember secure passwords. Approach for the system to be designed to improve the security of these systems relies on recognition based, rather than totally relying on recall based authentication. After examining the requirements of a recognition based authentication system a solution is proposed, which authenticates a user through his or her ability to recognize previously seen images. It will be more reliable and easier to use than traditional recall based schemes, which require the user to precisely recall passwords or PINs. Furthermore, it has the advantage that it prevents users from choosing weak passwords and makes it difficult to write down or share passwords with others.

The concept is mainly based on the fact that visual perception in human beings is far better than any other perception. Human being can easily recognize a previously seen image within a fraction of seconds. A very general example one can see in their day today life, a person meets so many people around but its worth observing that he can easily recognize the faces he has seen but he may not be in a position to recall their names.

So the solution, which has been proposed, is based on this property of human being. Replacing the textual passwords with the images would heavily reduce the load of the user of recalling tough to remember passwords. In the next chapter design and implementation part is been explained in detail.

3.3 Attacks and Countermeasures

There are various attacks possible on the system, in this section we would be analyzing few most powerful attacks on the system and will show that how the system is showing good resistance to them.

Firstly, analysis of the system against for most powerful attack brute force will be done.

3.3.1 Brute force attack

Mathematically it will be shown that the system designed shows a very good resistance against a brute force attack. Since each of the block consists of 1600 pixels, and each of the pixel is represented by 32 bits we can easily see that one block will represent 51200 bits. If one considers ASCII characters then its known that one ASCII character is represented by 8 bits and therefore one image block will generate about 6400 characters. Moreover it can be easily calculated that even if the user is choosing just 4 blocks then those 4 blocks will generate 25600 characters or 204800 bits.

All these bits are passed through one way hash function that uses it as input for generating secure password, and it is a property of a good one way hash function that even if there is a change in a single bit the output value will change.

Suppose user has selected just a single block for password then it at least 2^{51200} attempts will be required to crack the password, which is computationally infeasible. It can be easily calculated and shown that how powerful will be the generated password, if four or more than four blocks are selected for generating the password.

So brute force attack is ineffective for this system and hence it is very secure against common attacks like password guessing.

Moreover if suppose for an instance that the cracker has somehow received the image which was used for the generation of password even then it could be shown that the system is still hard to break as compared to the traditional authentication systems. Lets consider that the user is choosing 4 character textual password out of 95 printable characters the total number of combinations which the cracker needs to try is 95^4. If the user is choosing 4 blocks out of 150 blocks of image then number of possible combinations to try is 150^4. It could be seen that the probability in the first case for correct password is $1/95^4$ whereas in the second case it will be $1/150^4$. This probability could be lowered much farther if we increase the number of blocks in which the image is divided.

3.3.2 Educated guess attack

This attack is based on the prediction about the taste of the user. Like if many individual images like images of flowers, images of cars etc is given to the user and asked to select the images out of that set, then it's a general tendency, not always true, that women will be going for choosing flowers and other such kind of things but men will be going to choose like horse, cars etc.

Moreover if suppose someone knows the taste of the user then one can predict the kind of images he or she has chosen for generating the password. In this system this kind of attack is very tough as no individual images are provided but same image is divided into the number of blocks out of which user has to choose few blocks as his or her password. So no any such prediction will work, as individual blocks of the image will not reveal any information about the taste of the user.

Although the system is resistant to very powerful attacks like brute force even then it is recommended that after few number of unsuccessful logins the system should halt. This would give one more level of security against any such malicious attempt to break in the system.

succeeds in entering the correct blocks with correct sequence the login succeeds else fails [Figure 4.2.3.1].

This is the basic design of the system proposed and in the next section it will be seen the basic flow diagram which will be followed for the implementation of the system.

4.2 Flow diagram

4.2.1 Password generation procedure

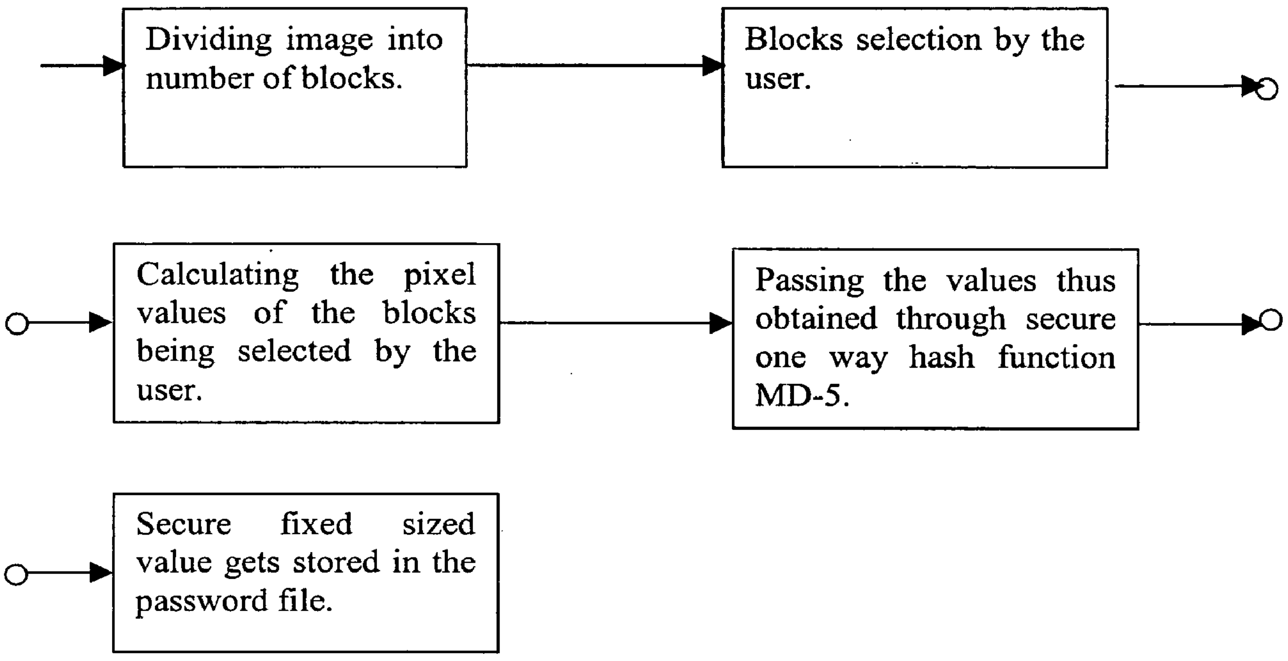

Dividing image into number of blocks.
Blocks selection by the user.
Calculating the pixel values of the blocks being selected by the user.
Passing the values thus obtained through secure one way hash function MD-5.
Secure fixed sized value gets stored in the password file.

Figure 4.2.1.1 Steps in password generation from image.

4.2.2 Login procedure

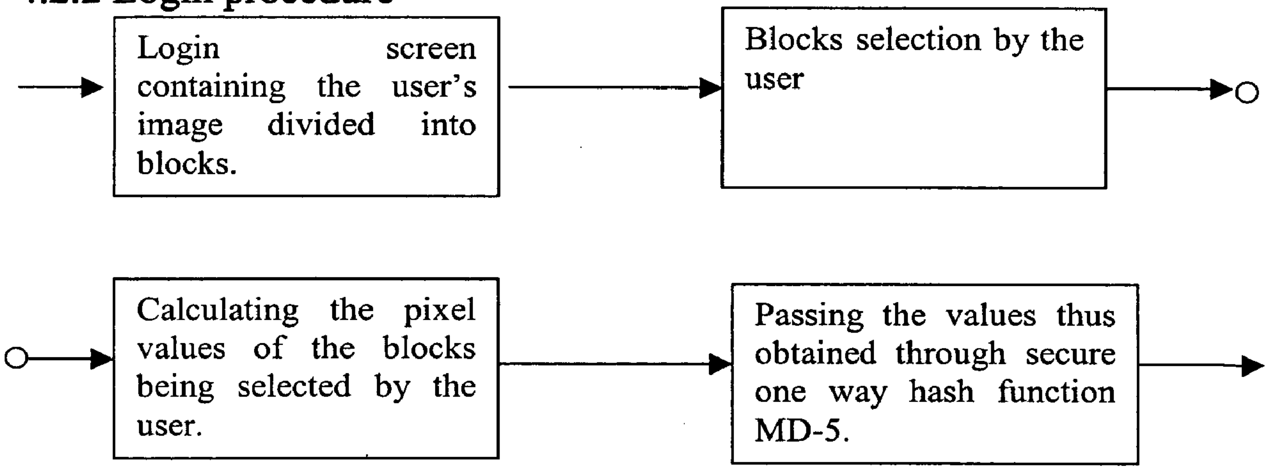

Login screen containing the user's image divided into blocks.
Blocks selection by the user
Calculating the pixel values of the blocks being selected by the user.
Passing the values thus obtained through secure one way hash function MD-5.

Figure 4.2.2.1 Login process using image as password.

4.2.3 Authentication procedure

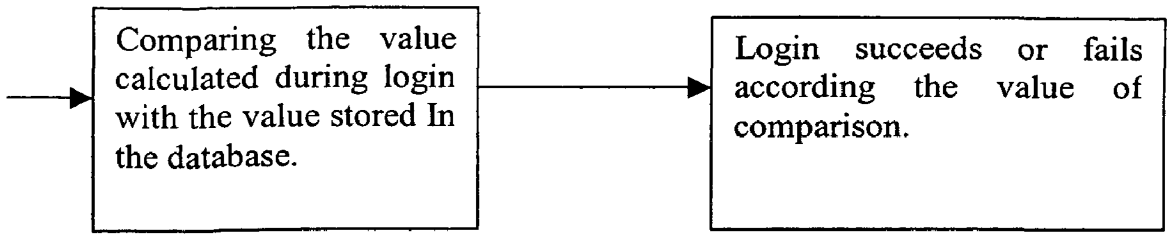

Comparing the value calculated during login with the value stored In the database.
Login succeeds or fails according the value of comparison.

Figure 4.2.3.1 Authentication process.

In this system user would create his or her portfolio from the set of images that are generated by dividing the user's own image or any other image into number of blocks and each of the blocks would represent an independent image.

Once the user has chosen the image blocks, those blocks sequence would become his or her password for entering the system. When the user tries to log on next time, after entering the user name he or she will be presented with the same image that was used for password creation. User will be prompted to select the same block sequence, which was selected for password creation, if the user selects the same sequence then he or she is allowed to log on else not.

4.3 Implementation of the system

System is implemented according to the design discussed before. MD5 is used as one way hash function [7].

4.3.1 Description of MD5

First, the message is padded so that its length is just 64 bits short of being a multiple of 512. This padding is a single 1-bit added to the end of the message, followed by as many zeroes as required. Then, a 64-bit representation of the message's length (before padding bits were added) is appended to the result. These two steps serve to make the message length an exact multiple of 512 bits in length (required for the rest of the algorithm), while ensuring that different messages will not look the same after padding.

Four 32-bits variables are initialized:

A=0x01234567

B=0x89abcdef

C=0xfedcba98

D=0x76543210

These are called chaining variables.

Now, the main loop of the algorithm begins. This loop continues for as many 512-bits blocks are in the message.

The four variables are copied into different variables: a gets A, b gets B, c gets C, d gets D. The main loop has four rounds, all very similar. Each round uses a different operation 16 times. Each operation performs a nonlinear function on three of a, b, c, and d. Then it adds that result to the fourth variable, a sub-block of the text and a constant. Then it rotates that result to the right a variable number of bits and adds the result to one of a, b, c, or d. Finally the result replaces one of a, b, c, or d [Figure 4.3.1.1].

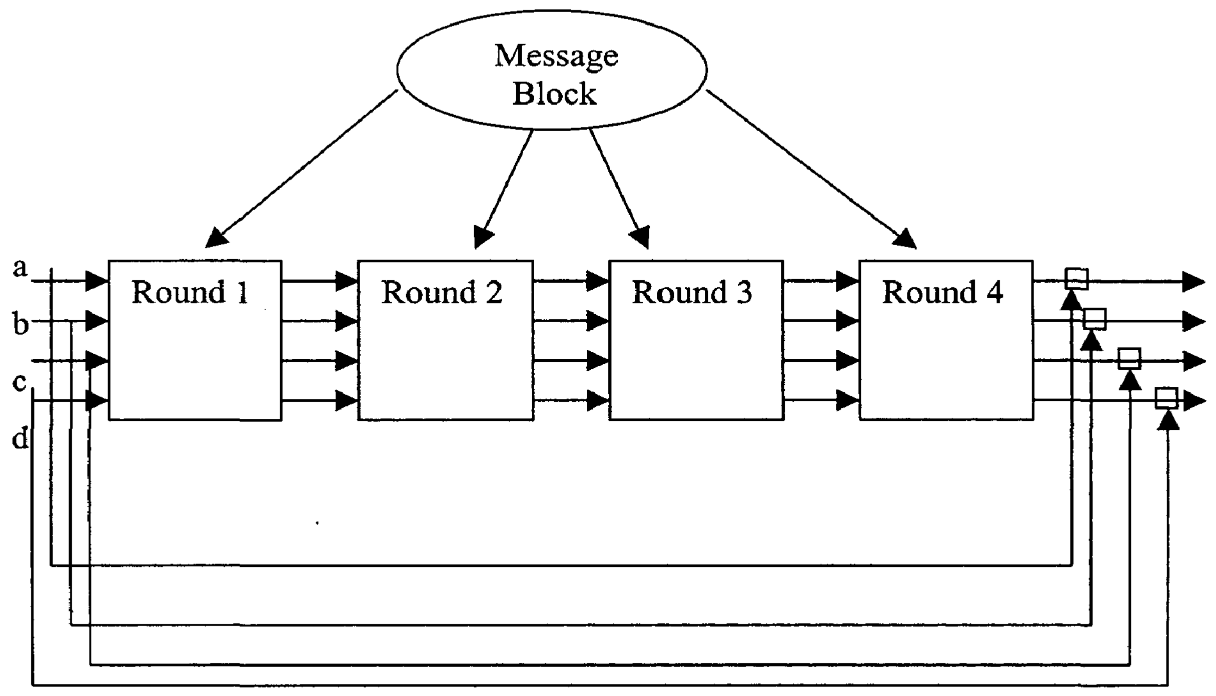


Figure 4.3.1.1 MD5 main loop.

There are four non-linear functions, one used in each operation (a different one for each round).

F(X, Y, Z) = (X^Y) || ((~X)&Z)

G(X, Y, Z) = (X&Z) || (Y&(~Z))

$H(X, Y, Z) = X*Y*Z$

$I(X, Y, Z) = Y*(X \| (\sim Z))$

(* is XOR, || is OR, & is AND and ~ is NOT)

These function are designed so that if the corresponding bits of X, Y, and Z are independent and unbiased, then each bit of the result will also be independent and unbiased.

If M_j represents the jth sub-block of the message (from 0 to 15), and <<<S represents a left circular shift of S bits, the four operations are [Figure 4.3.1.2]:

FF(a, b, c, d, M_j, S, t_i) denotes a = b + ((a + F(b, c, d) + M_j + t_i) <<<S)

GG(a, b, c, d, M_j, S, t_i) denotes a = b + ((a + G(b, c, d) + M_j + t_i) <<<S)

HH(a, b, c, d, M_j, S, t_i) denotes a = b + ((a + H(b, c, d) + M_j + t_i) <<<S)

II(a, b, c, d, M_j, S, t_i) denotes a = b + ((a + I(b, c, d) + M_j + t_i) <<<S)

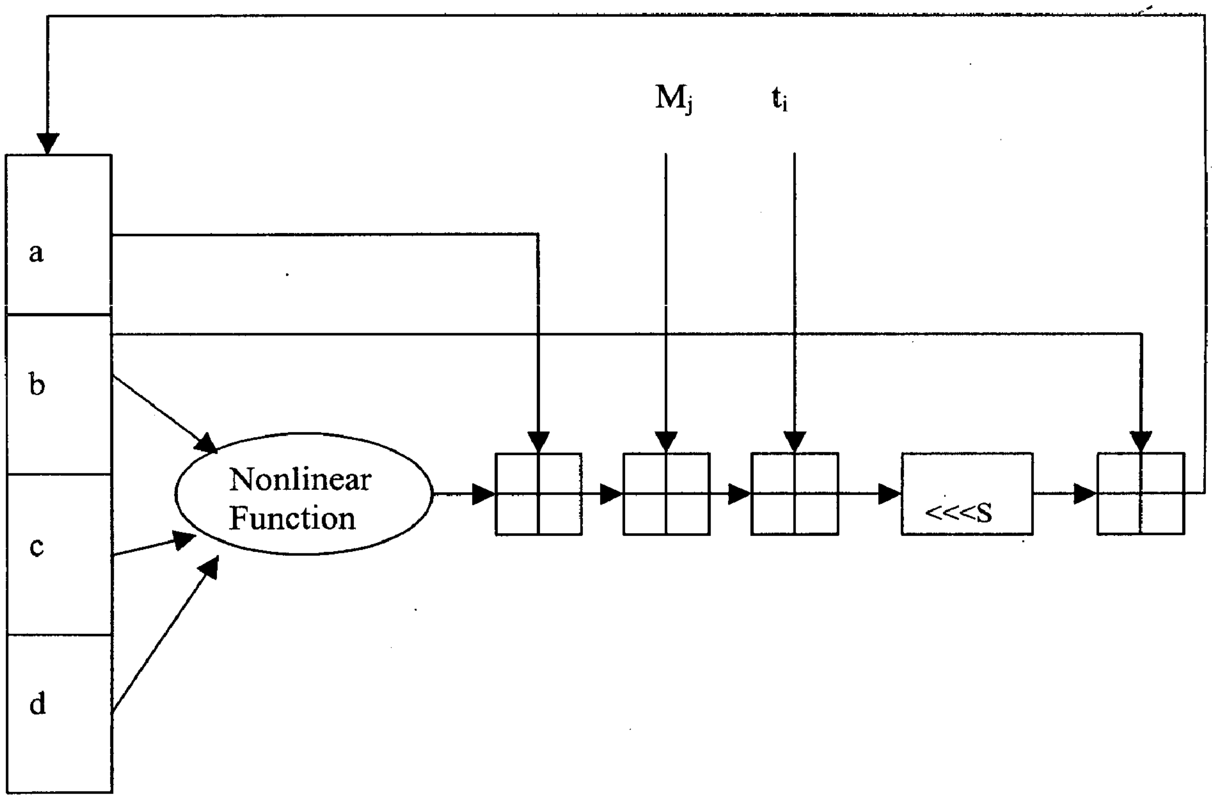


Figure 4.3.1.2 One MD5 operation.

Constants t_i is chosen as follows:

In step I, t_i is the integer part of 2^{32}*abs(sin(i)), where i is in radians.

After all this a, b, c, and d are added to A, B, C, and D respectively, and the algorithm continues with the next block of data. The final output is the concatenation of A, B, C and D.

This algorithm will take the pixel values of the image blocks selected by the user as a input and will produce a 128 bit unique value which will be stored in the password file.

4.3.2 Generation of input values for MD5

The algorithm has following steps by which it generates the input values for MD5 by reading the blocks selected by the user from the image.

(a) Divide the whole image into equal sized square blocks.

(b) As soon as the first click takes place on the image, start recording the co-ordinates that is rows and columns.

(c) According to the position of the co-ordinates block is located and all the pixels of that block is read.

(d) 8 bits each for Red, Green, Blue and one more component, which is luminance, is recorded for each pixel.

(e) Link list is formed in which each node will have data for one block.

(f) Recording continues till the user clicks to finish entering password.

(g) The whole link list is passed to the temporary buffer where padding and appending of length is done before sending it as an input for MD5.

CHAPTER 5
RESULTS AND DISCUSSION

5.1 Output screens

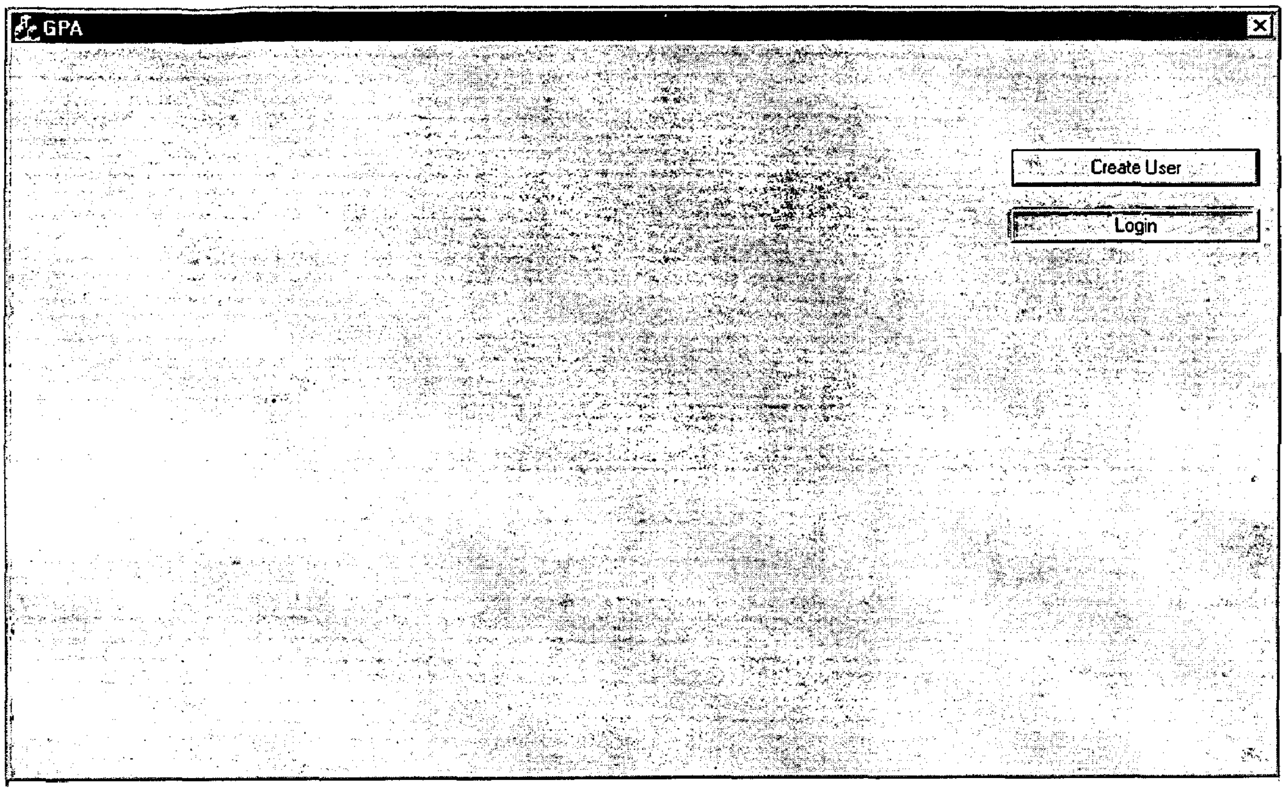
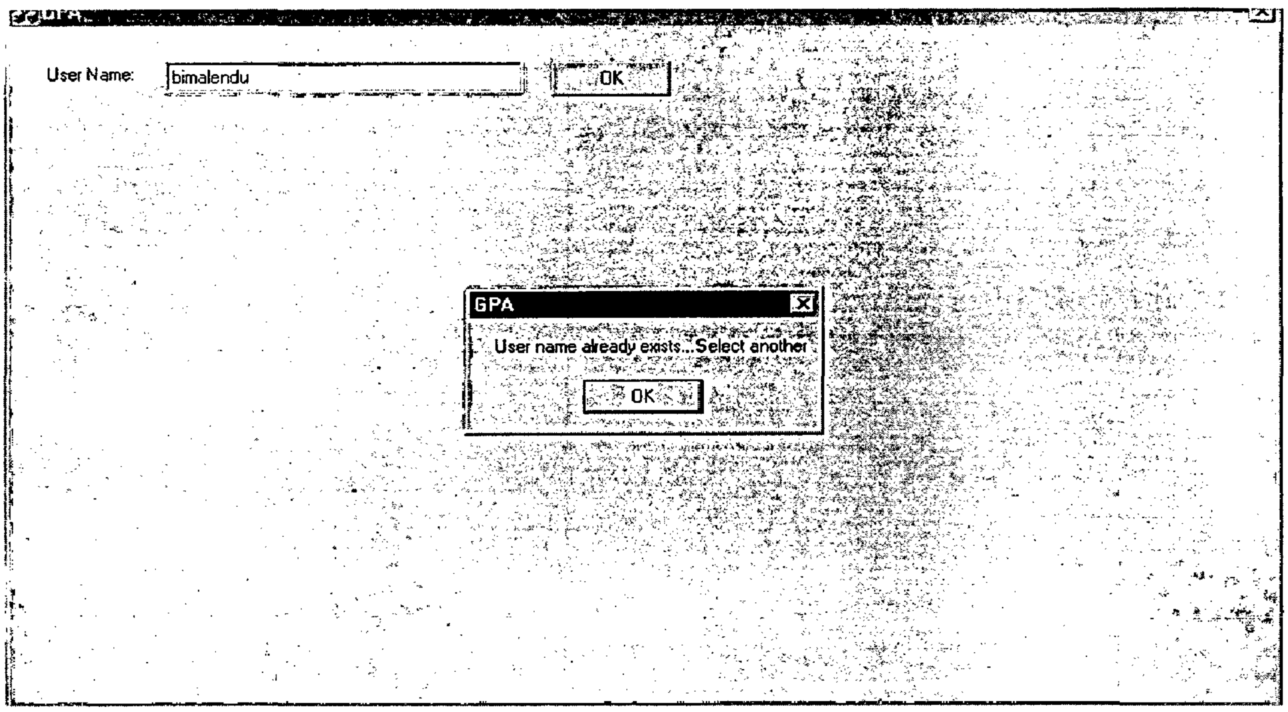


Figure 5.1.1 The initial screen of the authentication system designed.

This is the initial screen of the system developed for generating passwords using images. Two options are given one for login and other for creating new user. On clicking Login button user is prompted for user name after entering the user name the image is displayed which was used for the password creation.


User Name: bimalendu

OK

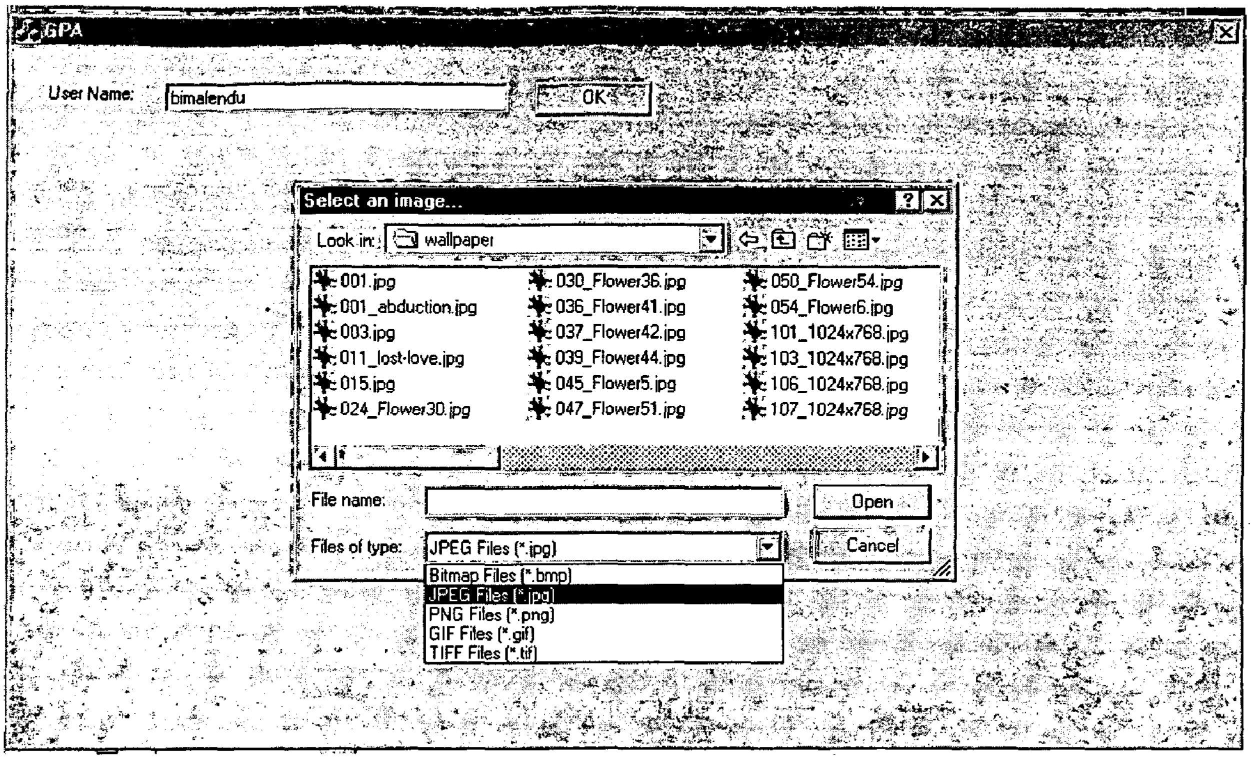
GPA

User name already exists...Select another

OK

Figure 5.1.2 The error message screen on creating user if user already exists.

On pressing Create User button text box is displayed and user is prompted to select a user name. If the user name exists already, message is displayed and user is advised to choose another user name.



User Name: bimalendu

Look in: wallpaper

001.jpg
001_abduction.jpg
003.jpg
011_lost-love.jpg
015.jpg
024_Flower30.jpg
030_Flower36.jpg
036_Flower41.jpg
037_Flower42.jpg
039_Flower44.jpg
045_Flower5.jpg
047_Flower51.jpg
050_Flower54.jpg
054_Flower6.jpg
101_1024x768.jpg
103_1024x768.jpg
106_1024x768.jpg
107_1024x768.jpg

File name:

Files of type: JPEG Files (*.jpg)

Bitmap Files (*.bmp)
JPEG Files (*.jpg)
PNG Files (*.png)
GIF Files (*.gif)
TIFF Files (*.tif)

Figure 5.1.3 Screen displayed for choosing image file.

On successfully choosing the username the user is allowed to choose image. Browser is provided to get access to the image files. Image files could be of bmp, jpeg, png, gif or tif format.

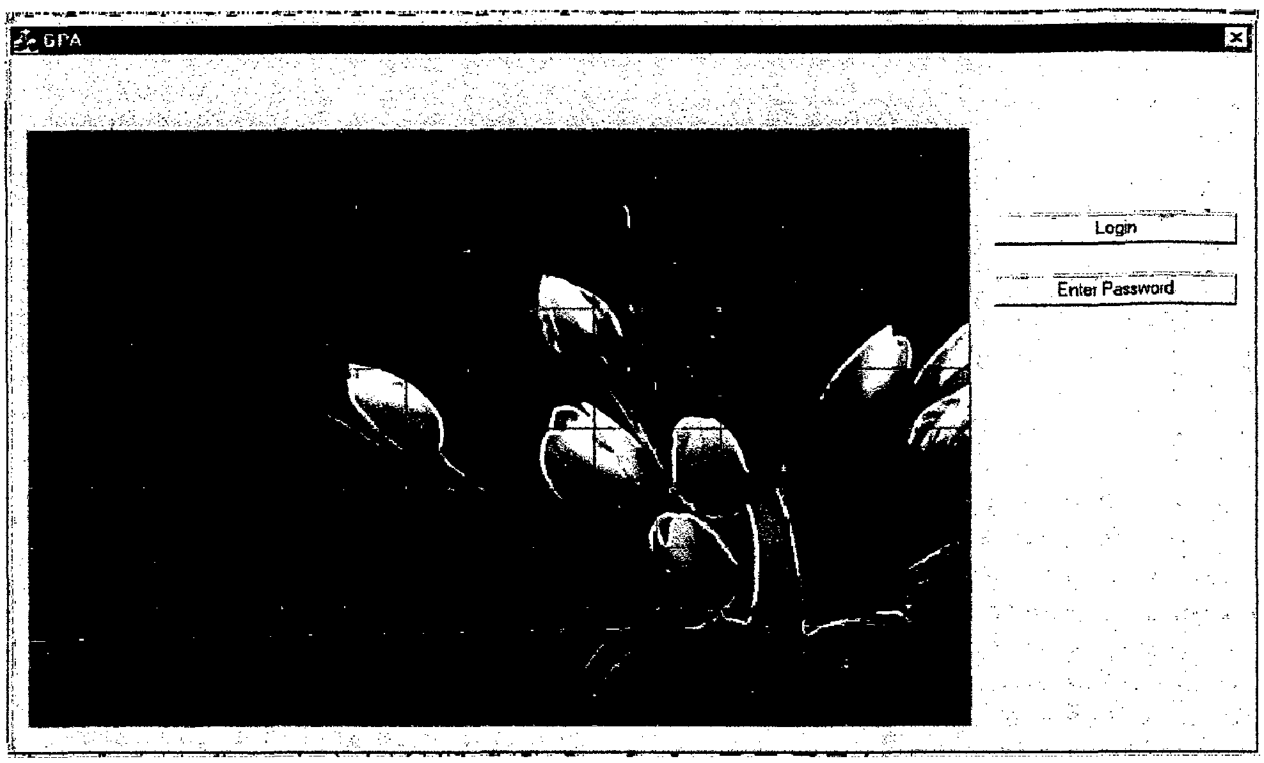


Figure 5.1.4 How image will be divided and displayed on the screen.

After choosing the image it is displayed and is divided into number of square blocks. User can choose from the blocks displayed by clicking on the particular block. The chosen blocks and that particular sequence will decide the password generated.

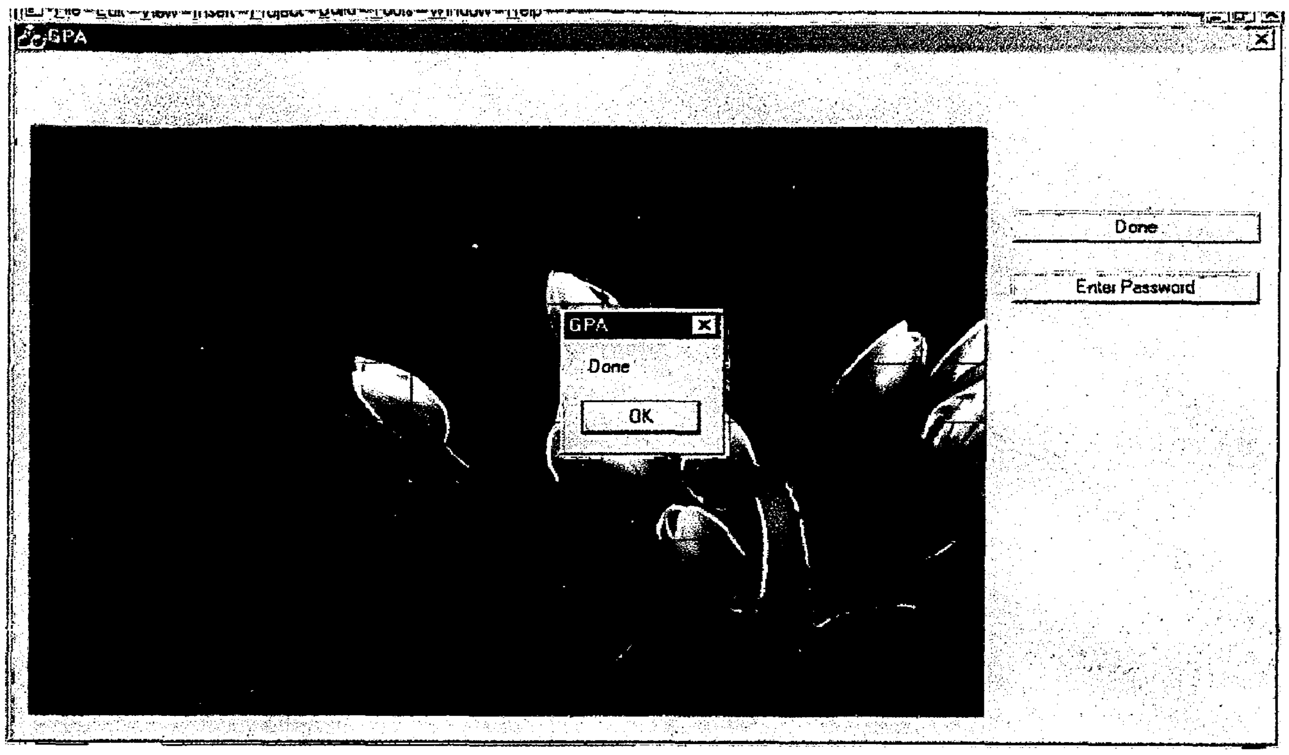
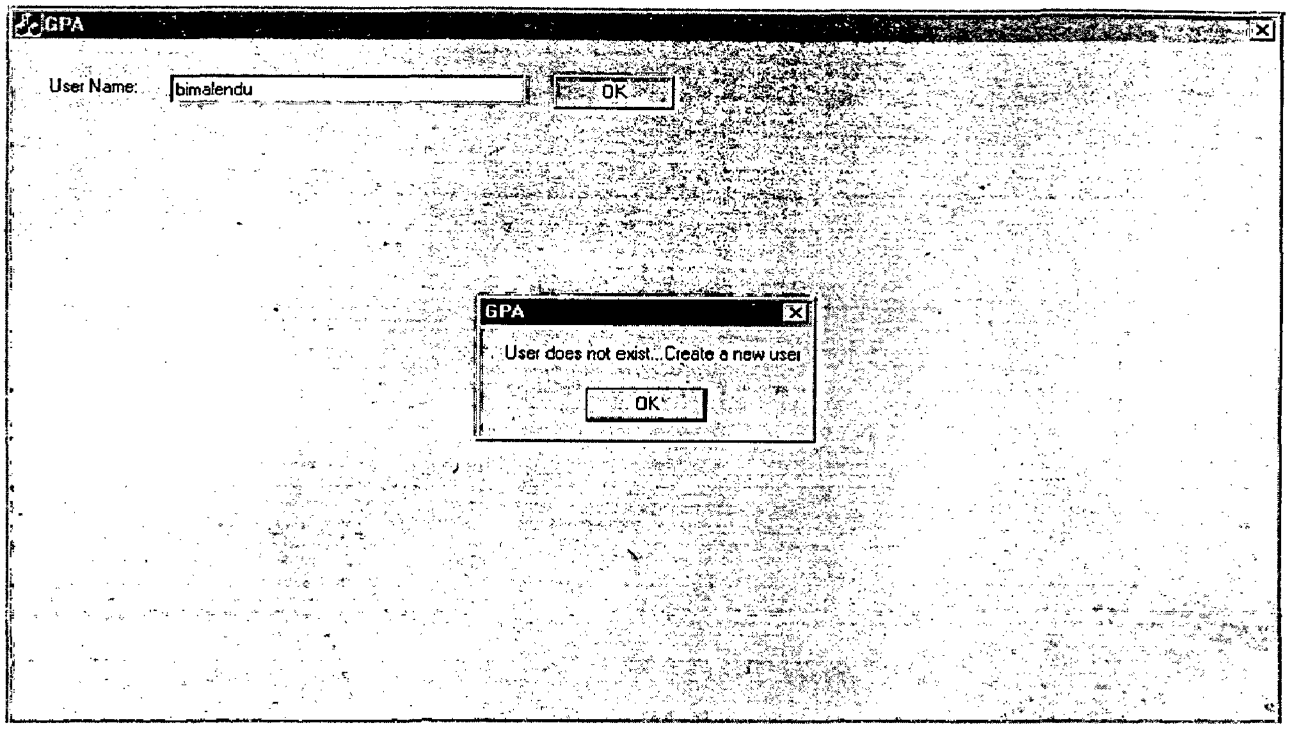


Figure 5.1.5 Screen as soon as the user selects the password.

After choosing the blocks done button is pushed to generate he password. The blocks chosen appears to be red in the screen which user could see and keep in mind. Now those red blocks will be used for further login.



User does not exist...Create a new user

Figure 5.1.6 If invalid user name is selected.

For login the user presses Login button a text box appears in which user has to type the user name allotted. If user does not exist, message is given and user is encouraged to re-enter the user name else the image associated to that particular user will be displayed on the screen.

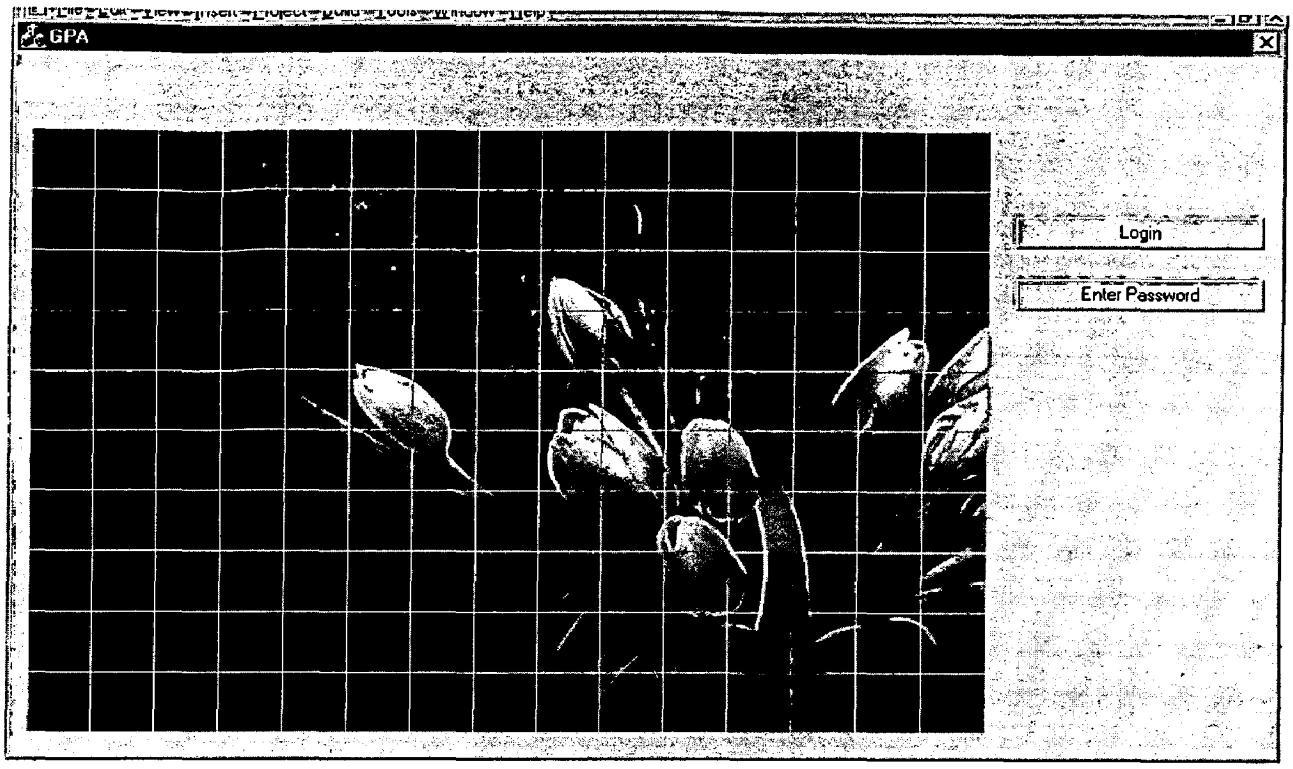


Figure 5.1.7 How image will be displayed after entering user name.

Screen displaying the image associated with particular user on successfully entering the username. The image is divided into the same number of blocks and each block of same size. Now user has to select the same blocks that were used to create his or her during password creation phase.

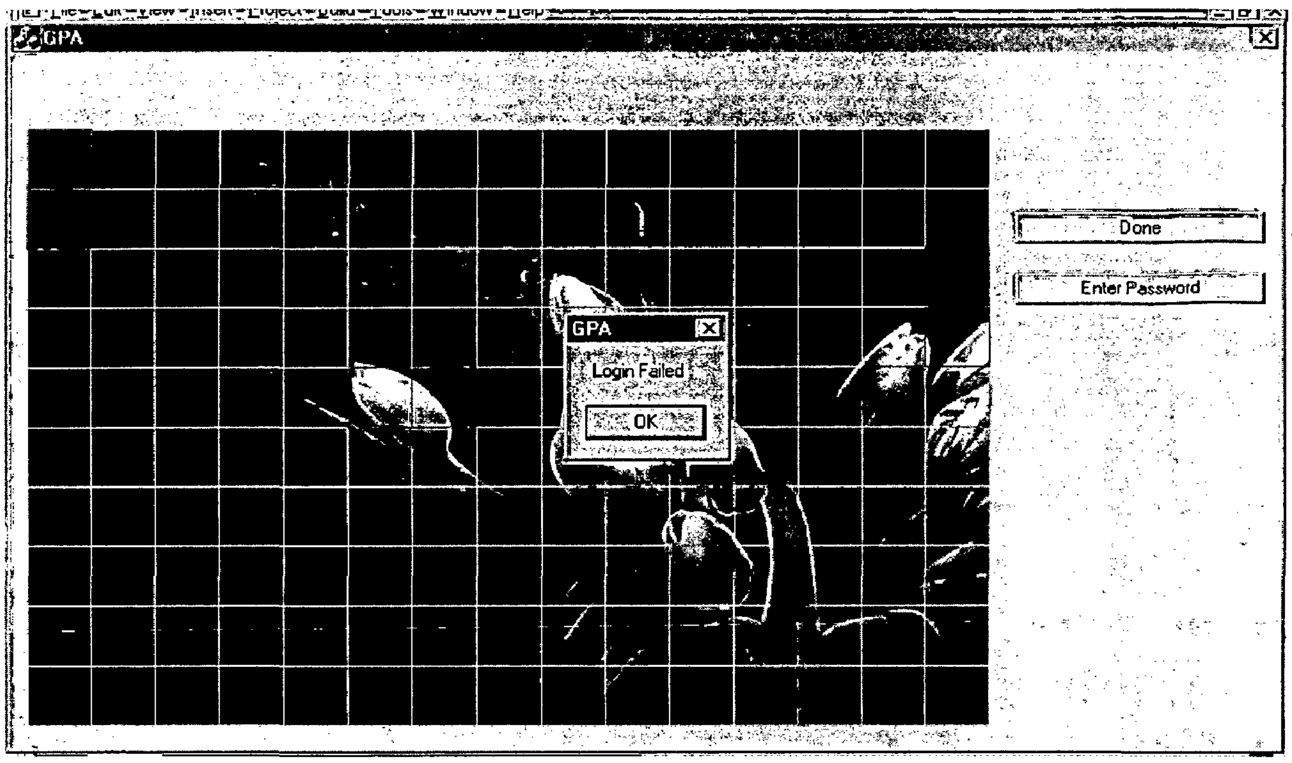
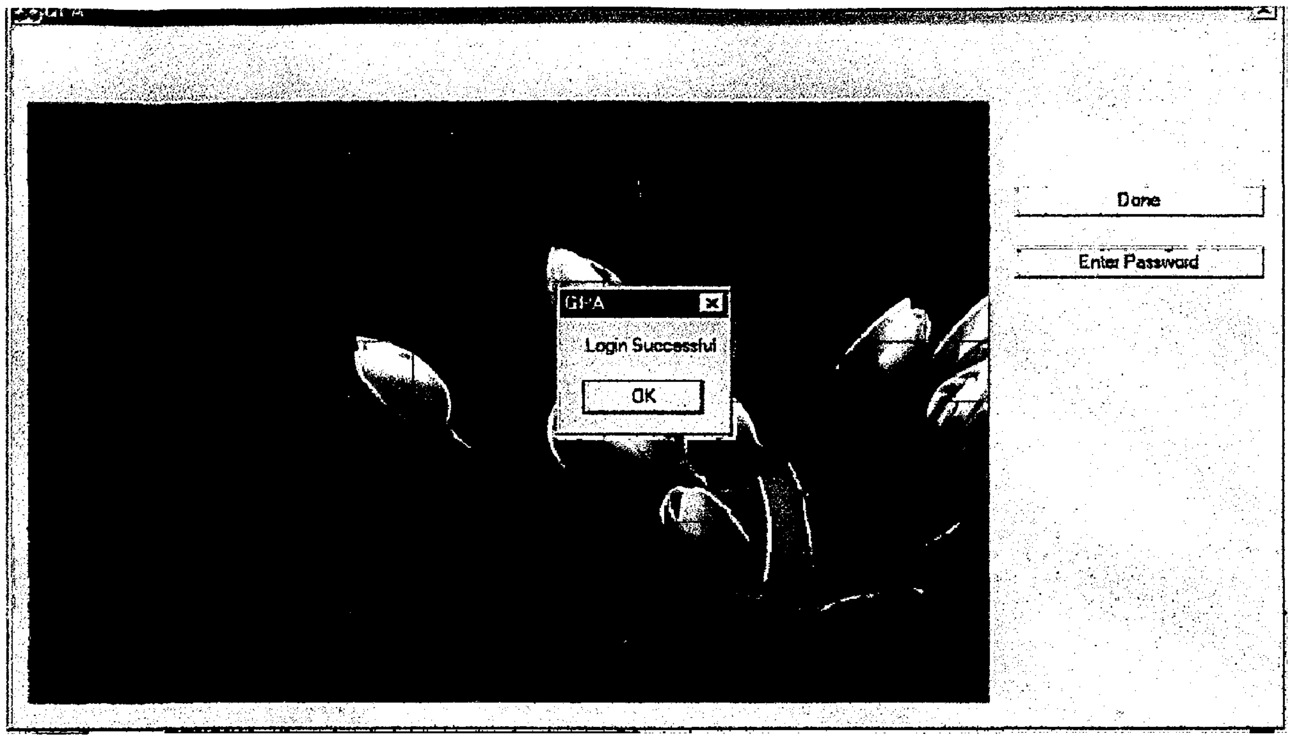


Figure 5.1.8 The Login Fail message if wrong blocks or sequence is chosen.

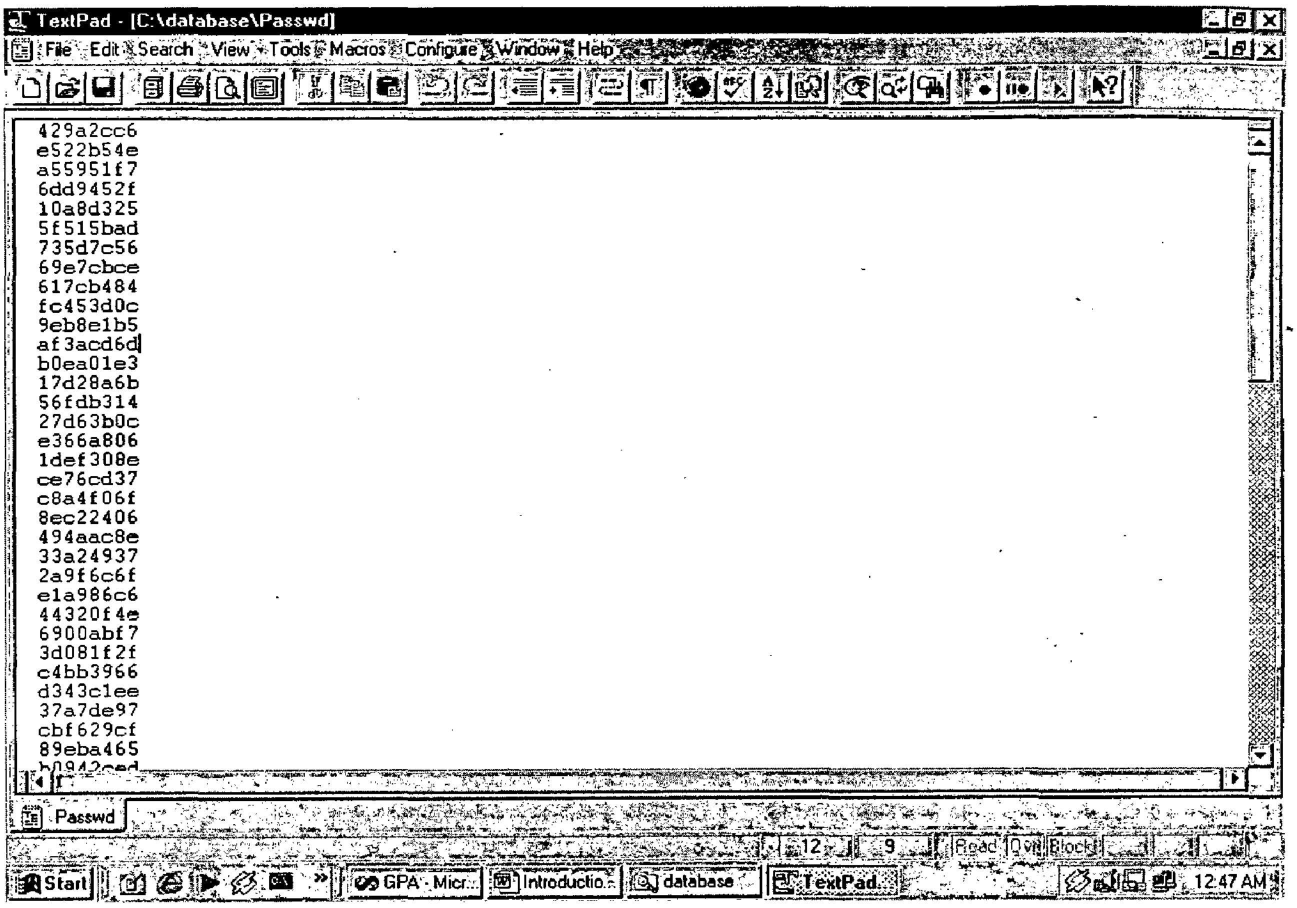
If the user is malicious and enters wrong sequence or wrong block login will fail. The user will be again asked to enter the correct password.



Login Successful

Figure 5.1.9 Screen when correct password is chosen.

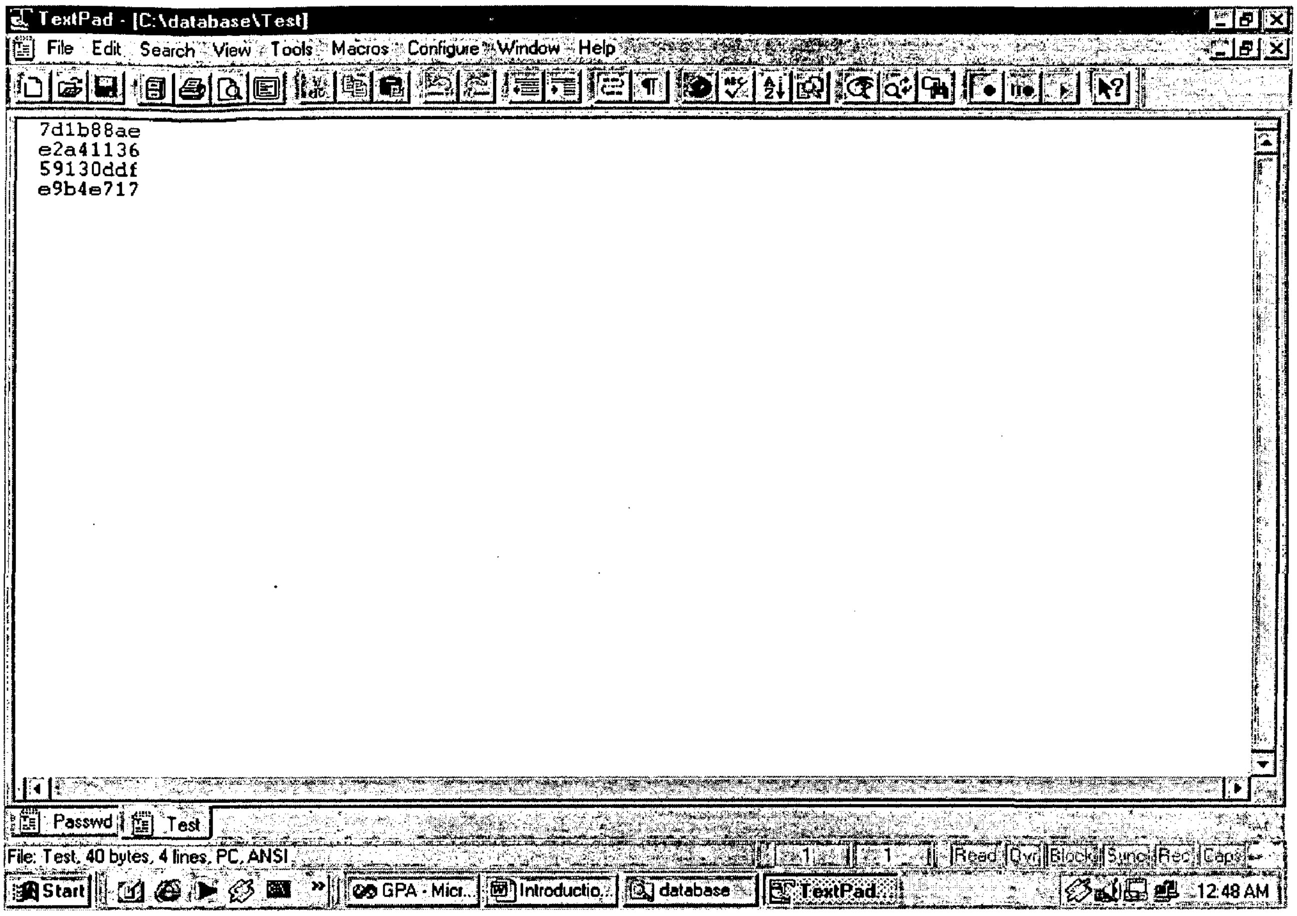
If the user enters the correct blocks with correct sequence the login successful message appears.


TextPad - [C:\database\Passwd]
File Edit Search View Tools Macros Configure Window Help
Passwd
12
9
Read
Ovr
Block
Start
GPA - Micr...
Introductio...
database
TextPad...
12:47 AM

```
429a2cc6
e522b54e
a55951f7
6dd9452f
10a8d325
5f515bad
735d7c56
69e7cbce
617cb484
fc453d0c
9eb8e1b5
af3acd6d
b0ea01e3
17d28a6b
56fdb314
27d63b0c
e366a806
1def308e
ce76cd37
c8a4f06f
8ec22406
494aac8e
33a24937
2a9f6c6f
e1a986c6
44320f4e
6900abf7
3d081f2f
c4bb3966
d343c1ee
37a7de97
cbf629cf
89eba465
```

Figure 5.1.10 How password is stored in the password file.

Password file consists of the sequence of the values generated by MD5 without mentioning the user name.

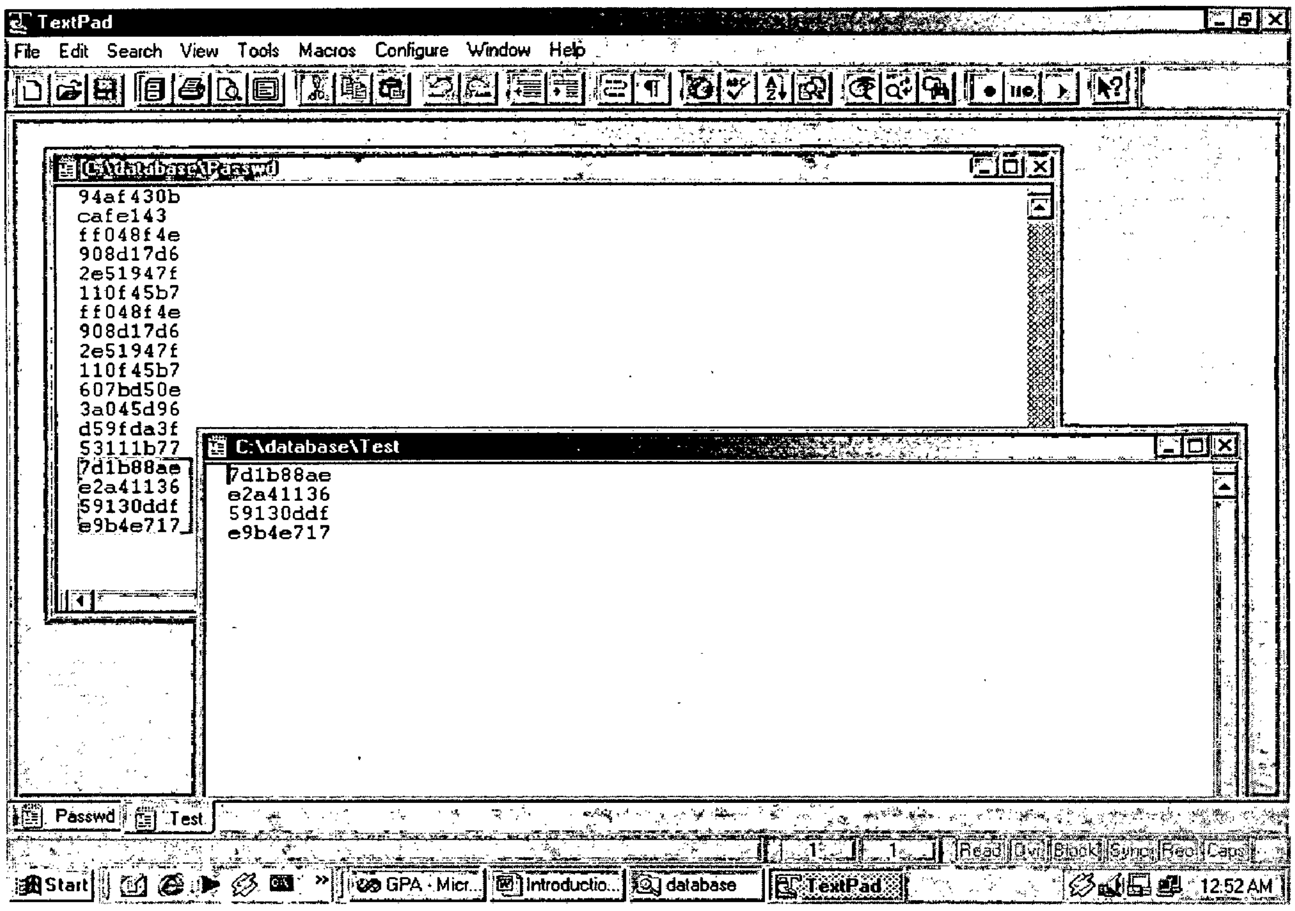

TextPad - [C:\database\Test]
File Edit Search View Tools Macros Configure Window Help
Passwd
Test
Start
GPA - Micr...
Introductio...
database
TextPad
12:48 AM

```
7d1b88ae
e2a41136
59130ddf
e9b4e717
```

```
File: Test, 40 bytes, 4 lines, PC, ANSI
```

Figure 5.1.11 The password generated when image blocks are selected.

View of the password generated by selecting the image blocks. This password will be compared to the list of password stored in the password file and if any matching is there login will be successful else not.



```
94af430b
cafe143
ff048f4e
908d17d6
2e51947f
110f45b7
ff048f4e
908d17d6
2e51947f
110f45b7
607bd50e
3a045d96
d59fda3f
53111b77
7d1b88ae
e2a41136
59130ddf
e9b4e717
```

```
7d1b88ae
e2a41136
59130ddf
e9b4e717
```

Figure 5.1.12 The password generated is found in the password file.

We can see the generated password matched with one of the member from list of passwords stored in the password file. It is shown highlighted.

5.2 Experiment conducted to study user behavior

Experiment was conducted to see user behavior about choosing passwords. About 30 people were interviewed, which included those people who belonged to the group who are trained about choosing good passwords and those who are not at all trained.

Although the number of people interviewed is quite small but we can assume them to represent the whole community. The general observation made by the interview was that people always tend to choose easy to remember passwords and since good passwords are hard to remember even those people who choose good passwords keep the same password for their multiple account. Most of the people try to choose their date of birth, their family names or dictionary words.

In general it is observed that people prefer convenience over security. Therefore the security should be inherent in any authentication system.

5.3 Experiment for visual perception

One experiment was conducted to check the visual perception of the user. In this experiment, 10 users were selected and they were given different textual passwords which satisfy the requirements of a good password and they were given few image blocks to recognize which will be used for their password.

One week later they were asked to write down the textual passwords. Only 5 users could recall the correct password, whereas in case of images 8 users successfully recognized the blocks with sequence. This experiment showed that visual perception is far better than any other perception.

CHAPTER 6

CONCLUSION

From the study of knowledge-based authentication systems the weaknesses associated with them were explored. Most of the authentication systems suffer with these weaknesses because they ignore the human factor.

It was shown that if we use images instead of textual passwords most of the weaknesses associated with the knowledge based authentication system could be removed up-to great extent. The solution proposed could be well incorporated in the existing infrastructure.

Moreover this kind of authentication system will be very useful in the coming mobile devices which support images and which use stylus as a input device.

6.1 Problems and future work

One of the major problem associated with the system is that while the user is trying to log onto the system image will be displayed on the screen and the tap regions could be observed by someone who, by several observations, may know the image blocks used for logging on.

One solution suggested could be to use the concept of visual cryptography. The image used for a particular user could be divided into two parts. One part will remain with the system and other with the user on some storage medium. Now when the user will try to log on to the system, system will prompt for the half part of the image that is with the user. User will provide that half part to the system, system will generate the original image which will be temporary. This image will be used by the user to get the image blocks required for password. After successful login this temporary image will be destroyed by the system.

Advantage of this solution is that, even if somebody observes the login procedure he or she will need that half part of the image for log on which is with the authentic user. This will provide one more level of security by the authentication system.

It is also recommended to use well featured images as it will give more variation in the pixel values, moreover well featured images are quite easier to recognize.

6.2 OTHER WORKS DONE BY AUTHOR

A research paper based on this thesis has been published in the proceedings of the National Conference on Information Security held on January 8, 9 at New Delhi [15].

REFERENCES

[1] David C. Feldmeier and Philip r. karn, "UNIX password security- Ten years later", Advances in cryptology CRYPTO 89 proceedings, volume 435, 1989.

[2] Rachna Dhamija and Adrian Perrig, "Déjà vu: A user study using images for authentication", SIMS/CS, University of California Berkeley, June 15 2000.

[3] Udi Manber, "A simple scheme to make passwords based on one way functions much harder to crack", Computers and security, November 1994.

[4] Linda Dailey Paulson, "Taking a graphical approach to the password", IEEE computer magazine, volume 35, No. 07, page 19, july 2002.

[5] Ian Jermyn, Alain mayer, Febian Monrose, Michael K. Reiter, Avi Rubin, "The design and analysis of graphical passwords", In the proceedings of the 8th USENIX security symposium, Washington D.C., August 1999.

[6] Robert Morris and Ken Thompson, "Password Security: A case history", Communications of the ACM, November 1979.

[7] B. Schneier, "Applied cryptography", Second edition, John Wiley & Sons, ISBN:0471117099, 1996.

[8] Passlogix. V-go. WWW at http://www.passlogix.com/, 2000.

[9] ID Arts. http://www.id-arts.com/technology/papers, 1999.

[10] Carl Ellison, Chris hall, Randy Milbert, and Bruce Schneier, "Protecting secret keys with personal entropy to appear in future generation computers", 1999.

[11] M. Naor and B. Pinkas, "Visual authentication and identification", Advances in cryptology-CRYPTO 89 proceedings, 1989.

[12] M. Naor and A. Shamir, "Lecture notes in computer science", Visual cryptography.

[13] Anne Adams aand Martina Angela Sasse, "Users are not the enemy: Why users compromise computer security mechanisms and how to take remedial measures", Communications of the ACM, 42(12):40-46, December 1999.

[14] John Miano, "Compressed Image File Formats", Addison Wesley, ISBN: 0-201-60443-4.

[15] Bimalendu Gupta, "Graphical password authentication system", In the proceedings of the National conference on Information security, January 7-8, 2003.

APPENDIX-A

GDIPLUS CLASSES AND METHODS USED IN THE SOFTWARE

A.1 GDI Plus:

Microsoft Windows GDI+ is a class-based application-programming interface (API) for C/C++ programmers. It enables applications to use graphics and formatted text on both the video display and the printer.

GDI+ is the subsystem of the Windows XP operating system or Windows.NET Server family. This API is exposed through a set of C++ classes. As its name suggests, GDI+ is the successor to Windows Graphics Device Interface (GDI), the graphics device interface included with earlier versions of Windows.

A graphics device interface, such as GDI+, allows application programmers to display information on a screen or printer without having to be concerned about the details of a particular display device. The application programmer makes calls to methods provided by GDI+ classes and those methods in turn make the appropriate calls to specific device drivers. GDI+ insulates the application from the graphics hardware, and it is this insulation that allows developers to create device-independent applications.

The C++ interface to Microsoft Windows GDI+ contains about 40 classes, 50 enumerations, and 6 structures. There are also a few functions that are not members of any class. Two such functions are GdiplusStartup and GdiplusShutdown. GdiplusStartup must be called before making any other GDI+ calls, and GdiplusShutdown must be called when the use of GDI+ is over.

A.2 Working with Images:

A.2.1 Gdiplus Classes used in the software

A.2.1.1 The Graphics Class:

The Graphics class provides methods for drawing lines, curves, figures, images, and text. A Graphics object stores attributes of the display device and attributes of the items to be drawn.

A.2.1.2 The Image Class:

GDI+ provides the Image class for working with raster images (bitmaps) and vector images (metafiles). The Image class provides methods for loading and saving raster images (bitmaps) and vector images (metafiles). An Image object encapsulates a bitmap or a metafile and stores attributes that can be retrieved by calling various Get methods. Image objects can be constructed from a variety of file types including BMP, ICON, GIF, JPEG, Exif, PNG, TIFF, WMF, and EMF.

For displaying a raster image (bitmap) on the screen, an Image object and a Graphics object are required. The name of a file (or a pointer to a stream) is passed to the constuctor of the Image class to create the Image object. After the Image object has been created, the address of that Image object is passed to the DrawImage method of an object of the Graphics class.

The following example creates an Image object from a JPEG file and then draws the image with its upper-left corner at (60, 10):

```
Image image(L"Grapes.jpg");
graphics.DrawImage(&image, 60, 10);
```

Figure A.2.1.2.1 shows the image drawn at the specified location.

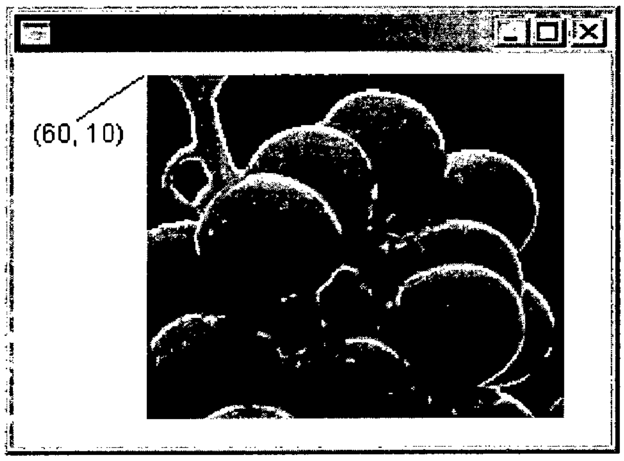


Figure A.2.1.2.1: An image displayed using the DrawImage method

A.2.1.3 The Bitmap Class:

The **Bitmap** class inherits from the **Image** class and expands on the capabilities of the Image class by providing additional methods for loading, saving, and manipulating raster images.

A.2.1.4 BitmapData Class

The **BitmapData** class is used by the LockBits and UnlockBits methods of the Bitmap class. A **BitmapData** object stores attributes of a bitmap. The following table lists the members exposed by the **BitmapData** object.

The following table lists the members exposed by the **BitmapData** object. Click a tab on the left to choose the type of member you want to view.

Data member	**Type**	**Description**
Width	UINT	Number of pixels in one scan line of the bitmap.
Height	UINT	Number of scan lines in the bitmap.
Stride	INT	Offset, in bytes, between consecutive scan lines of the bitmap. If the stride is positive, the bitmap is top-down. If the stride is negative, the bitmap is bottom-up.
PixelFormat	PixelFormat	Integer that specifies the pixel format of the bitmap.
Scan0	void *	Pointer to the first (index 0) scan line of the bitmap.
Reserved	UINT_PTR	Reserved for future use.

Table A.2.1.4.1: lists the members exposed by the BitmapData object.

A.2.2 Constructors and methods of the above classes used in the software

A.2.2.1 Image::Image Constructor

Creates an **Image** object based on a file.

Syntax

```
Image(

   const WCHAR* filename,
   BOOL useEmbeddedColorManagement
);
```

Parameters

[in] Pointer to a wide-character string that specifies the name of the file.

useEmbeddedColorManagement

[in] Optional. Boolean value that specifies whether the new **Image** object applies color correction according to color management information that is embedded in the image file. Embedded information can include International Color Consortium (ICC) profiles, gamma values, and chromaticity information.

FALSE

Default. Specifies that color correction is enabled

TRUE

Specifies that color correction is not enabled

Return Value

No return value.

Remarks

You can construct **Image** objects based on files of a variety of types including BMP, Graphics Interchange Format (GIF), JPEG, PNG, TIFF, and EMF.

A.2.2.2 Graphics::DrawImage Method

Overload 1: The DrawImage method draws an image at a specified location.

Syntax

```
Status DrawImage(

    Image* image,
    INT x,
    INT y
);
```

Parameters

image

[in] Pointer to an Image object that specifies the image to be drawn.

x

[in] Integer that specifies the x-coordinate of the upper-left corner of the rendered image.

y

[in] Integer that specifies the y-coordinate of the upper-left corner of the rendered image.

Return Value

If the method succeeds, it returns **Ok**, which is an element of the Status enumeration.

A.2.1.4 BitmapData Class

The **BitmapData** class is used by the LockBits and UnlockBits methods of the Bitmap class. A **BitmapData** object stores attributes of a bitmap. The following table lists the members exposed by the **BitmapData** object.

The following table lists the members exposed by the **BitmapData** object. Click a tab on the left to choose the type of member you want to view.

Data member	Type	Description
Width	UINT	Number of pixels in one scan line of the bitmap.
Height	UINT	Number of scan lines in the bitmap.
Stride	INT	Offset, in bytes, between consecutive scan lines of the bitmap. If the stride is positive, the bitmap is top-down. If the stride is negative, the bitmap is bottom-up.
PixelFormat	PixelFormat	Integer that specifies the pixel format of the bitmap.
Scan0	void *	Pointer to the first (index 0) scan line of the bitmap.
Reserved	UINT_PTR	Reserved for future use.

Table A.2.1.4.1: lists the members exposed by the BitmapData object.

A.2.2 Constructors and methods of the above classes used in the software

A.2.2.1 Image::Image Constructor

Creates an **Image** object based on a file.

Syntax

```
Image(

   const WCHAR* filename,
   BOOL useEmbeddedColorManagement
);
```

Parameters

[in] Pointer to a wide-character string that specifies the name of the file.

useEmbeddedColorManagement

[in] Optional. Boolean value that specifies whether the new **Image** object applies color correction according to color management information that is embedded in the image file. Embedded information can include International Color Consortium (ICC) profiles, gamma values, and chromaticity information.

FALSE

Default. Specifies that color correction is enabled

TRUE

Specifies that color correction is not enabled

Return Value

No return value.

Remarks

You can construct **Image** objects based on files of a variety of types including BMP, Graphics Interchange Format (GIF), JPEG, PNG, TIFF, and EMF.

A.2.2.2 Graphics::DrawImage Method

Overload 1: The DrawImage method draws an image at a specified location.

Syntax

```
Status DrawImage(

     Image* image,
     INT x,
     INT y
);
```

Parameters

image

[in] Pointer to an Image object that specifies the image to be drawn.

x

[in] Integer that specifies the x-coordinate of the upper-left corner of the rendered image.

y

[in] Integer that specifies the y-coordinate of the upper-left corner of the rendered image.

Return Value

If the method succeeds, it returns **Ok**, which is an element of the Status enumeration.

If the method succeeds, it returns **Ok**, which is an element of the Status enumeration.

If the method fails, it returns one of the other elements of the **Status** enumeration.

Overload 2: The DrawImage method draws an image.

Syntax

```
Status
 DrawImage(
   Image* image,
   const Rect &destRect,
   INT srcx,
   INT srcy,
   INT srcwidth,
   INT srcheight,
   Unit srcUnit,
   ImageAttributes* imageAttributes,
   DrawImageAbort callback,
   VOID* callbackData
);
```

Parameters

image

[in] Pointer to an Image object that specifies the source image.

destRect

[in] Reference to a rectangle that bounds the drawing area for the image.

srcx

[in] Integer that specifies the x-coordinate of the upper-left corner of the portion of the source image to be drawn.

srcy

[in] Integer that specifies the y-coordinate of the upper-left corner of the portion of the source image to be drawn.

srcwidth

[in] Integer that specifies the width of the portion of the source image to be drawn.

[in] Integer that specifies the height of the portion of the source image to be drawn.
srcUnit
[in] Element of the Unit enumeration that specifies the unit of measure for the image. The default value is **UnitPixel**.
imageAttributes
[in] Pointer to an ImageAttributes object that specifies the color and size attributes of the image to be drawn. The default value is NULL.
callback
[in] Callback method used to cancel the drawing in progress. The default value is NULL.
callbackData
[in] Pointer to additional data used by the method specified by the *callback* parameter. The default value is NULL.
Return Value
If the method succeeds, it returns **Ok**, which is an element of the Status enumeration.
If the method fails, it returns one of the other elements of the **Status** enumeration.
Remarks
The portion of the source image to be drawn is scaled to fit the rectangle.

A.2.2.3 Image::GetHeight Method
The GetHeight method gets the image height, in pixels, of this image.
Syntax
UINT GetHeight(VOID);
Return Value
This method returns the height, in pixels, of this image.

A.2.2.4 Image::GetWidth Method
The GetWidth method gets the width, in pixels, of this image.
Syntax
UINT GetWidth(VOID);
Return Value
This method returns the width, in pixels, of this image.

This method returns the width, in pixels, of this image.

A.2.2.5 Image::Save Method

The **Save** method saves this image to a file.

Syntax

```
Status Save{
        const WCHAR* filename,
        const CLSID* clsidEncoder,
        const EncoderParameters* encoderParams
    );
```

Parameters

filename

[in] Pointer to a null-terminated string that specifies the path name for the saved image.

clsidEncoder

[in] Pointer to a **CLSID** that specifies the encoder to use to save the image.

encoderParams

[in] Optional. Pointer to an EncoderParameters object that holds parameters used by the encoder. The default value is NULL.

Return Value

If the method succeeds, it returns Ok, which is an element of the Status enumeration.

If the method fails, it returns one of the other elements of the **Status** enumeration.

A.2.2.6 Bitmap::Bitmap Constructor

Overload 1: Creates a **Bitmap** object of a specified size and pixel format. The pixel data must be provided after the **Bitmap** object is constructed.

Syntax

INT *height*,

PixelFormat *format*

);

Parameters

width

[in] Integer that specifies the width, in pixels, of the bitmap.

height

[in] Integer that specifies the height, in pixels, of the bitmap.

format

[in] Optional. Integer that specifies the pixel format of the bitmap. The **PixelFormat** data type and constants that represent various pixel formats are defined in Gdipluspixelformats.h. For more information about pixel format constants, see Image Pixel Format Constants. The default value is PixelFormat32bppARGB.

Return Value

No return value.

Overload 2: Creates a **Bitmap** object based on an image file.

Syntax

Bitmap(

const WCHAR* *filename*,

BOOL *useIcm*

);

Parameters

filename

[in] Pointer to a null-terminated string that specifies the path name of the image file. The graphics file formats supported by GDI+ are BMP, GIF, JPEG, PNG, TIFF, Exif, WMF, and EMF.

useIcm

[in] Optional. Boolean value that specifies whether the new **Bitmap** object applies color correction according to color management information that is embedded in the image file. Embedded information can include International

/

[in] Optional. Boolean value that specifies whether the new **Bitmap** object applies color correction according to color management information that is embedded in the image file. Embedded information can include International Color Consortium (ICC) profiles, gamma values, and chromaticity information. TRUE specifies that color correction is enabled, and FALSE specifies that color correction is not enabled. The default value is FALSE.

Return Value

No return value.

A.2.2.7 Bitmap::LockBits Method

The **LockBits** method locks a rectangular portion of this bitmap and provides a temporary buffer that you can use to read or write pixel data in a specified format. Any pixel data that you write to the buffer is copied to the Bitmap object when you call UnlockBits.

Syntax

```
Status LockBits(
   const Rect* rect,
        UINT flags,
        PixelFormat format,
        BitmapData* lockedBitmapData
   );
```

Parameters

rect

[in] Pointer to a rectangle that specifies the portion of the bitmap to be locked.

flags

[in] Set of flags that specify whether the locked portion of the bitmap is available for reading or for writing and whether the caller has already allocated a buffer. Individual flags are defined in the ImageLockMode enumeration.

format

[in] Integer that specifies the format of the pixel data in the temporary buffer. The pixel format of the temporary buffer does not have to be the same as the

Constants. Microsoft® Windows® GDI+ version 1.0 does not support processing of 16-bits-per-channel images, so you should not set this parameter equal to PixelFormat48bppRGB, PixelFormat64bppARGB, or PixelFormat64bppPARGB.

lockedBitmapData

[in, out] Pointer to a BitmapData object. If the ImageLockModeUserInputBuf flag of the *flags* parameter is cleared, then *lockedBitmapData* serves only as an output parameter. In that case, the **Scan0** data member of the **BitmapData** object receives a pointer to a temporary buffer, which is filled with the values of the requested pixels. The other data members of the **BitmapData** object receive attributes (width, height, format, and stride) of the pixel data in the temporary buffer. If the pixel data is stored bottom-up, the **Stride** data member is negative. If the pixel data is stored top-down, the **Stride** data member is positive. If the ImageLockModeUserInputBuf flag of the *flags* parameter is set, then *lockedBitmapData* serves as an input parameter (and possibly as an output parameter). In that case, the caller must allocate a buffer for the pixel data that will be read or written. The caller also must create a **BitmapData** object, set the **Scan0** data member of that **BitmapData** object to the address of the buffer, and set the other data members of the **BitmapData** object to specify the attributes (width, height, format, and stride) of the buffer.

Return Value

If the method succeeds, it returns Ok, which is an element of the Status enumeration.

If the method fails, it returns one of the other elements of the **Status** enumeration.