The Multi-Dimensional Encryption of Data

New Mexico Adventures In
Supercomputing Challenge

Final Report
April 5, 2006

Team 005
Albuquerque Academy

Project Member:

Wesley Smalls

Sponsoring Teacher/Project Mentor:

Jim Mims

Table of Contents

Executive Summary

Cryptography is an indispensable tool used around the world to protect people's important information. It is for this reason that countries world-wide spend billions of dollars every year researching new encryption methods. This project attempts to display the security power of a new Multi-Dimensional encryption routine through its ability to encrypt simple text messages into a completely unreadable format.

In order for the program to be useful under any circumstances, the user must be able to easily understand the interface. Also, the program required many specific built-in functions in order to operate properly. Because these reasons, I chose Visual Basic 6.0 as my programming language. Visual Basic 6.0's spectacular user interfaces and natural syntax made the aesthetics simple, and left me more time to concentrate on creating the code for the complex encryption algorithms.

The method that the program uses to encrypt the data is based on three main concepts: the use of the "Keyword" version of the Caesar-Shift encryption method, the conversion of the data into numbers, and the Multi-Dimensional encryption routine.

The "Keyword" version of the Caesar-Shift encryption method is one based on the original Caesar Shift. The use of the "Keyword" version is the basis behind most large-scale encryption routines used today due to the fact that there are an almost infinite number of possible passwords. I chose to use the "Keyword" version because of this fact.

The method used to convert the encrypted text into a string of numbers is one of the project's most important routines. By converting the text into numbers, the program is able to manipulate the data in ways that would be impossible if it were using the ASCII characters. I chose to use this method because of its versatility and because it could be used under any circumstances.

The Multi-Dimensional encryption routine is one that utilizes a multi-dimensional array consisting of six dimensions (Appendix A, Figure 3). The data is added to and then removed from the array in unrelated orders. I chose to use the Multi-Dimensional encryption routine as the backbone for the project because of the fact that it brings many powerful aspects into the overall algorithm. These aspects include: an almost untraceable method of scrambling the data, the ability to make many messages of differing sizes the same length, and the ability to add random data into the end result so as to further mislead people trying to gain illegal access to the encrypted data.

The results from the program's testing showed that the algorithm was completely effective. It successfully converts all of the message's data to numbers, shifts those numbers using the Caesar-Shift method, and scrambles the numbers using the Multi-Dimensional encryption routine. It was a major success for the Project.

Introduction

In selecting a project idea, I wanted to choose something that would present me with a challenge. I eventually chose to research cryptography because of its uses in home and business security and, in the end, I chose to create a program that could encrypt a message into a completely unreadable format. Hopefully, this program will be the forerunner to various, more advanced encryption routines that will help to both ensure security on the highest level, and increase our understanding of new encryption methods.

By using this program, notes, messages, and entire papers can be successfully encrypted with relative ease. However, because of the resource-intensive nature of the encryption routine the program uses, extremely long messages can often take some time to encrypt. Because of this reason, a supercomputer's resources and speed would be required to fully demonstrate the program's usefulness and ability, making the project very appropriate for the AiS Challenge.

There are many different types of encryption schemes known today, the most common being "Shifts" and "Scrambles". Shifts are both easy to use and simple to compute. Scramble methods on the other hand, are more difficult to compute because of the fact that the program must be able to move freely among the string being encrypted. However, Scramble methods are much more difficult to crack, and so therefore desirable in encryption routines. Because both of these were fairly easy to compute, I decided to implement both of them into the encryption routine.

Shift ciphers are methods that relate characters such as letters to arbitrary numbers, and then "shift" the characters up by a certain number, producing a new character. The most common type of Shift cipher is the Caesar Shift method. In the Caesar Shift, letters of the alphabet are assigned to the numbers 1 through 26 so that: a = 1, b = 2, c = 3, etc. If the letters were shifted by 3, then: a = 4, b = 5, c = 6, and so on. All the encryption routine would have to do then is substitute in the letter corresponding to the shifted number. In this case, the conversion would go as follows: a = 1, 1 + 3 = 4, and 4 = d.

Scrambling methods tend to be more complex than simple Shift methods. For instance, in a “Rail Fence” scramble, a new message is created by taking, in turn, every other letter in the original message and adding it to the end of the new message. An example of this would be the message “In the closet.” The scrambled version would be “I h lstntecoe.” As you can see, it is quite confusing, and therefore very useful in encryption methods and routines.

Though “Shifts” and “Scrambles” seem almost laughably easy for a computer to decrypt when they are used on their own, with the right combination of the two types of methods, an almost unbreakable code can be achieved. In an attempt to create one of these “perfect” codes, the encryption routine developed in this project was made to use both the “Keyword” version of the Caesar Shift method and a Multi-Dimensional encryption Routine. The routine was then augmented by the addition of a routine that manipulates and changes the data by converting it to numbers and systematically changing the numbers’ values.

Description and Methods

This program strives to make full use of three different methods in an attempt to create an unbreakable encryption. These methods are: the "Keyword" version of the Caesar Shift method, a Multi-Dimensional encryption routine, and the manipulation of data through the process of converting that data into numbers and changing the numbers' values. In my originally simplified program, each method was used to create a separate type of encrypted file. I used this program to perfect each of the methods to the point where they could be successfully combined to achieve my goals.

During my research, I found that the order in which I implemented each algorithm in conjunction with the others was very important. After consulting my mentor, Jim Mims, I decided that the best method to use to create the correct algorithm was to try all of the different possible combinations until I came up with the best fit choice.

Before starting on the application, I had to select a programming language. After reviewing the different languages and what it was I wanted to accomplish, I chose Visual Basic 6.0. As the name suggests, it is very visually and interface oriented which was very important to the outcome of the program. Also, the natural-language syntax made it considerably easier to write the complex code than it would have been in other languages. These properties allowed me to easily create an interface that is both user-friendly and aesthetically pleasing, while being able to spend most of my time working on the many complex algorithms involved. For a simpler, more concise view of my algorithm and general approach, see the flowchart in Appendix A, Figure 1.

In order to have the program be able to effectively manipulate all of the encrypted messages, I had to create an interface and program structure that was heavily object oriented. Object Oriented Programming, or OOP, is a method of programming that utilizes "objects" of a "class" and then manipulates these objects for the user's purpose. The reason that OOP is so useful in this project is because the program needs to be able to manipulate each encrypted message separately. To do this, all of the encrypted messages must be uniform and possess all of the same properties. Also, the program can be easily expanded upon by adding additional properties or methods to the class. In these ways, OOP helps to increase the efficiency and functionality of the program.

In my original prototype program, I focused on the development of each method separately. In the prototype, each method was used to create separate encryptions that would be compared to expected results obtained from research. At the end of this "testing" phase, all three separate algorithms would me melded into one large algorithm.

The first method to be perfected was the "Keyword" version of the Caesar Shift method. The original Caesar Shift Method, or CSM, is a simple monalphabetic shift method that uses the arbitrary value of a single letter (or simply a number) to shift the values of all the letters in any given message up or down and uses the letters corresponding to the new values to create the ciphertext (encrypted text). Although the CSM has been used for centuries and is still used today, it is a very simple method of encryption and so easily decrypted. However, the "Keyword" version of the CSM is a much more complex method. It is implemented in much the same way as the original CSM, but instead of using a single letter to encrypt a message, it sequentially uses the individual values of the letters in the Keyword (password) to shift corresponding letters in the plaintext (un-encrypted text) so as to create the ciphertext. In this way, depending on the lengths of the plaintext and the password, there is essentially no limit to the number of possible combinations.

The next method to be adapted for use in my program was the manipulation of data by converting it to numbers and systematically changing the numbers' values. There were many different ways that I could have gone about implementing this method. However after reviewing the pros and cons of many different types, I chose to use a method that treated the numbers as strings so that the program could manipulate them both as numbers and as strings. The method creates a fixed length string (4 characters long) and assigns the product of the value of a letter and an arbitrary number to the string. After completing this, zeros are added to the front of the string (so the value doesn't change) until the total character count of the string equals 4, and the string is reversed. By using this method, the original value of the letter can be hidden very effectively.

The third and final method that I used was the Multi-Dimensional Encryption Routine, or MDER. This routine is definitely the most important part of the program. It creates an array consisting of six dimensions and uses the array to scramble the plaintext. Each character in the plaintext is sequentially added to the elements of the array. This is accomplished by adding a character of the plaintext to one set of coordinates in the array, changing one of the coordinates, and then adding the next character to the new coordinates. The algorithm continues in this way until the entire plaintext is contained within the array. Because of the fact that there are usually empty spaces left in the array, these empty spaces are then filled with random data. When the array is completely full, the data is removed in much the same way as it was entered into the array. The only difference is that the starting coordinates and the order in which the coordinates are changed are different from the way in which the data was entered into the array.

Because I designed the program for use with multiple people, I based the interface and program structure around maximum security. However, because of the increasing complexity of the program, it became clear to me that it would need a certain, pre-runtime-created folder and file structure in order to be able to operate correctly. It was because of this that I decided to create a simple installer; a support program that would create this folder and file structure in a directory decided upon by the user. Because I decided to build the program around a maximum security model, I created three different classes: CUsers, CProjects, and CEncryptions. Each class has a set of properties that are used in identification and security. I also decided to create two BASIC modules: ProjectLoader and EncryptionBuilder. ProjectLoader contains Procedures used to extract information about Users, Projects, and Encryptions from their corresponding files. EncryptionBuilder contains two Procedures that are used to encrypt and decrypt the properties of Projects and Encryptions.

CUsers is the class that deals with different user profiles. Before a person can use the program, they must register a user-name and password with the program so that it can create their user-profile. This is done so that only registered users can use the program. CUsers contains several properties: Name (the property that holds the user's name), Password (used to hold the user's password), Encryption (holds the contents of the encrypted file that stores the user's profile), EntryAttempts (holds the number of times the user has entered the wrong password; Used to decide if someone is attempting to illegally use the user's account), and LockedOut (used to determine if the user can use the program). These properties are used by the program's driving routines in order to allow the user to interface with the program correctly and securely.

The two Procedures (Sub Procedures) in CUser are Encrypt and Decrypt. Encrypt is used to create a ciphertext containing the user's password. This ciphertext is created with the user's name as the "password" for the encryption. The Full Encryption Routine, or FER, which is described in the CEncryption Class, is the routine used. After Encrypt has created the ciphertext, it stores it in a file created for the user. Decrypt is the routine that decrypts the ciphertext stored in the user's file and assigns the decrypted plaintext (the user's password) to the Password property of the user's user-profile. Decrypt uses the Full Encryption Routine Decrypter, or FERD, to decrypt the plaintext. The FERD will also be discussed in the CEncryption Class.

CProject is the class that deals with all of the users' projects. Objects of CProject, called Projects, can be thought of as the folders that hold all of a user's important documents (Encryptions). A project must be created by a registered user and has three modes: Public, Limited, and Private. In Public mode, any user with the project's password can access it. In Limited mode, only certain, pre-registered users who know the project's password can access it. In Private mode, only the user who created the project can use it. Projects have several properties: Name (holds the project's name), Password1 (holds the project's first password), Password2 (holds the project's second optional password), EType (holds the project's current mode), CreationDate (when the project was created), EDate (when the project was last edited), ENumber (used in the number manipulation of the FER), Owner (holds the name of the user that created the project), Status (states whether the project is open or closed), and Encryption (holds the ciphertext of the file that stores the project). These properties are used by the program to assist with the user-interface as well as with the program's overall security level.

CProject's Procedures are: Encrypt, SmallEncrypter, and Decrypt. Encrypt is the Procedure that is called when the project is being created and/or edited. It sets the project's EDate property to the current date and time (separated by a space) and creates the proper folder and file structure for the project in the program's installation directory. After it has done all of this, it calls SmallEncrypter.

SmallEncrypter is what actually encrypts the project's data. SmallEncrypter creates two fixed length strings of 4 characters each: L and Temp. It then sets the project's Encryption property to empty, and sets the ENumber property to a randomly generated integer from 1 to 39. After that, it sets the values of two variables in the BASIC module (a code module that is public and can be accessed from any form or class) EncryptionBuilder:

ENumber and Pass, to the values of the project's ENumber property and Owner property respectively. After performing these preliminary steps, SmallEncrypter then begins to encrypt each of the project's properties. It does this by setting the fixed length string variable L to the length of the property and passing L to the public function LengthCheck. LengthCheck takes the string passed to it, and adds zeros to the beginning of the string until it is exactly 4 characters in length. Upon L's return from the LengthCheck function, SmallEncrypter sets the value of a variable in EncryptionBuilder, EncypteeLen, to the value of L. L is then passed to the built-in function StrReverse. StrReverse reverses the order of the characters in the string passed to it. For example, if you passed the string "Cat" to StrReverse, it would return the string "taC". When L is passed back from StrReverse, it is added to the project's Encryption property. SmallEncrypter then sets the value of EncryptionBuilder's variable, Encryptee, to the value of the property being encrypted and calls EncryptionBuilder's Encrypt Procedure. EncryptionBuilder's Encrypt Procedure is like the FER except for the fact that it doesn't implement the MDER in its algorithm. The MDER was removed from EncryptionBuilder's Encrypt Procedure because the Encrypt Procedure is also used by the CEncryption class; if it were included, the properties would be indistinguishable from the encrypted ciphertext and the Decrypt routine would be unable to successfully decrypt it. EncryptionBuilder's Encrypt Procedure assigns the ciphertext to the public variable Encryptee. After SmallEncrypter calls EncryptionBuilder's Encrypt Procedure, it adds EncryptionBuilder's Encryptee variable to the end of the project's Encryption property. SmallEncrypter applies this method of encryption to the following properties in their respective order: CreationDate, EDate, EType, Password1, Password2, and Status. It then passes the project's ENumber property to the LengthCheck function, and sets the value of the fixed length string variable Temp to the returned value. SmallEncrypter's last step is to add Temp to the end of the Encryption property and write the Encryption property to the project's file for storage.

The last Procedure in the CProject Class is the Decrypt Procedure. The Decrypt Procedure is essentially the reverse of the Encrypt procedure and is called only when the project's Encryption property has the project's ciphertext in it. Decrypt uses no local variables at all, but instead relies on many interwoven functions to assign values from the ciphertext to the project's properties. Decrypt's first step is to set the project's ENumber property to the integer value of the right 4 numbers of the Encryption property and then remove these four numbers from the right of the Encryption property. Decrypt then sets EncryptionBuilder's variables: ENumber and Pass to the values of the projects ENumber and Owner properties respectively. After performing these preliminary steps, Decrypt begins to decrypt the ciphertext (the Encryption property) into the project's properties.

Decrypt's first step in decrypting the actual ciphertext into the individual properties is to pass the left 4 numbers in the ciphertext to the StrReverse function, and assigning the integer value of what is passed back to EncryptionBuilder's EncrypteeLen variable. Decrypt then removes those four numbers from the beginning of the ciphertext and sets EncryptionBuilder's Encryptee variable to the remainder of the ciphertext. The four numbers that were removed represent the length of the property that is next in line in the ciphertext. Because of this, Decrypt can then remove a number of characters from the ciphertext equal to the product of the value of those four numbers and the number 4. In so doing, Decrypt removes that portion of the ciphertext and is then able to move on to the rest of the ciphertext. Decrypt then calls EncryptionBuilder's Decrypt Procedure, which will decrypt the EncryptionBuilder's Encryptee variable using the opposite algorithm as its Encrypt Procedure in order to produce the project's property. Decrypt then assigns the project's corresponding property to EncryptionBuilder's Encryptee variable (now the decrypted plaintext). Decrypt uses this method to decrypt all of the following properties in their respective orders: CreationDate, EDate, EType, Password1, Password2, and Status.

The final class used in this project is CEncryption. CEncryption is the class that deals with all of the individual encryptions. Objects of CEncryption, Encryptions, can be thought of as the important documents that go into users' folders (Projects). An encryption can only be created from within a project, and so only by a registered user. Encryptions have several properties: Name (holds the encryption's name), ProjectName (holds the name of the encryption's containing project), ProjectOwner (holds the name of the encryption's containing project's owner), Password (holds the encryption's Password), BackupPassword (holds the encryption's randomly generated backup password), CreationDate (when the encryption was created), EDate (when the encryption was last edited), ENumber (used in the number manipulation of the FER), Encryption (holds the ciphertext of the file that stores the Encryption), EScript (holds the plaintext; used in the FER), and Status (states whether the Encryption is locked or unlocked). These properties are used by the program to assist with the user-interface as well as with the program's overall security level.

CEncryption's procedures are: EncryptScript EncryptData, PassCreator, Scramble, Writer, Recorder, SmallRecorder, DecryptScript, DecryptData, EncryptionUnlock, and Placer. EncryptScript, EncryptData, PassCreator, Scramble, Writer, and Recorder make up the FER while DecryptScript, DecryptData, EncryptionUnlock, and Placer make up the FERD. EncryptScript is the Procedure that is called when the Encryption is being created or when the user makes a direct change to the encrypted message and is responsible for calling many of the other Procedures in CEncryption (the Procedures that make up the FER). It sets the Encryption's ENumber property to a randomly generated integer value between 0 and 39. It then creates a randomly generated backup password of a randomly generated length of up to 30 characters. After doing this, EncryptScript calls the Procedures: EncryptData, PassCreator, Scramble, Writer, and Recorder. Between calls to CEncryption's other Procedures EncryptScript updates progress bars and certain labels on the Encrypter and Decrypter forms in order to keep the user informed of the Procedure's current process.

The Procedure EncryptData can be best described as a clone of CProject's SmallEncrypter Procedure as it only deals with the encryption of the Encryption's properties. However, there are some differences. For instance, unlike CProject's SmallEncrypter, EncryptData does not use the fixed length string variable Temp. EncryptData's first step is to set the Encryption's EDate property to the current date and the current time (separated by a space). It then assigns the Encryption's ENumber property's value to EncryptionBuilder's variables ENumber and Pass and sets them to the Encryptions's ENumber property and ProjectName and ProjectOwner (separated by a space) properties respectively. From there EncryptData encrypts all of the Encryption's properties in exactly the same way as CProject's SmallEncrypter.

PassCreator is a fairly minor Procedure. It is used to create the Multi-Dimensional array by calculating the smallest number that when raised to the sixth power is greater than the length of the plaintext. It then ReDims a dynamic array of type String called EncryptionArray with six dimensions, each with a number of elements equal to the "smallest number" referred to earlier in this paragraph. EncryptionArray is used to hold the plaintext/ciphertext during the encryption process. PassCreator's final step is to assign the public variable EncryptionNum the value of the "smallest number".

Unlike PassCreator, Scramble is one of the more important (not to mention large) Procedures in CEncryption. Scramble is responsible for assigning the plaintext to the elements of EncryptionArray and then implementing the CSM and Number Manipulation Routine, or NMR. Scramble uses several Long type variables: i, e, f, g, x, y, z, and n. Of these, six (i, e, f, g, x, and y) are used to represent the coordinates within EncryptionArray, while the other two (n and z) are used as counter variables. Scramble also uses a string variable called Pass which represents the password being used for the encryption and a fixed length string of length 4 characters called Letter that is directly used in the encryption. There are also two variables of type Double called Num1 and Num2 which are used to keep track of the Encryption's progress and update the progress bars on the Encrypter and Decrypter forms. Scramble's first step is to set Num1 equal to the quotient of 50 (representing 50% of the Encryption's progress, the other 50% is covered in the second For Loop of Writer) over the value of EncryptionNum squared. This is done in order to create a highly exact interval by which to increment the progress bars. Scramble's next step is to decide upon the password that it will use in the encryption process. If the Encryption has a valid user-made password (it is optional), then Pass is assigned the Encryption's Password property, otherwise Pass is assigned the value of the randomly generated BackupPassword property. The remaining code in Scramble is made up of six nested (within each other) For Loops (looping structures based whose terminating condition is the value of a numerical variable). Each For Loop's counting variable represents one of the coordinate slots for Encryption Array. All of the Loops' counting variables are set to 0 and all of their terminating points are set to one less than Encryption Num (because of the offset in numbers created when Dimensioning an Array). Each of the Loops will count in increments of 1 from 0 to their terminating points. Each time a Loop runs through its internal code (the code contained within the Loop structure), the counting variable for the Loop within it is set to equal 0 so that the coordinates can be reset to an empty element of EncryptionArray after each pass-over.

The second nested Loop contains code that adds the value of Num1 to the Value of Num2, and then assigns the Num2's value to the Value property of the progress bars on the Encrypter and Decrypter forms. This segment of code is done to keep the user informed of the Encryption's progress.

The sixth nested Loop is the Loop containing the CSM and NMR code. It selects a character from the plaintext and assigns the value of the product of that character's ASCII number and the Encryption's ENumber property to Letter. The ASCII value of a character from Pass is then added to the value of Letter and the total is then passed to the LengthCheck function and the returned string is then assigned to Letter. Letter is then assigned to the element of EncryptionArray that the coordinates currently point to. After each cycle of the Loop, the next characters of the plaintext and Pass are selected.

CEncryption's next Procedure is Writer. Like the relationship between EncryptData and CProject's SmallEncrypter, Writer is essentially a clone of scramble. However, Writer's nested loops are arranged in a different order than Scramble's (the reverse order to be precise). The changed order is done so that the data can be removed from the array in a different order from that which it was entered. However, despite what seems rather obvious from simply reversing the order, this does not produce a reversed plaintext, but instead shuffles the original message beyond recognition. In all reality, the only difference between Scramble and Writer is the fact that Writer does not deal with the CSM, but focuses more on the NMR. In Writer's sixth For Loop all of EncryptionArray's empty elements are filled with random numbers that the computer will disregard upon decryption. These random numbers range from the product (the highest number attainable for the ciphertext) of the Encryption's ENumber property and 255 (the total number of characters in the ASCII table), through 9999. All of the elements are then reversed using the StrReverse function and added to the end of the Encryption's Encryption property. Finally, ENumber is passed to the LengthCheck function and the returned value is added to the end of the ciphertext (Encryption property).

Recorder is the next Procedure in CEncryption. Recorder is used to create the Encryption's file and write the Encryption property (the ciphertext) to it. Other than that, Recorder adds the Encryption's name to the its containing Project's Encryption Registry file so that it will be loaded the next time the program is run.

SmallRecorder is used by the program to save changes to the Encryption's properties. It was created as a shortcut around the time and resource-consuming EncryptScript Procedure. SmallRecorder is only used after the encryption is made and so only has to add the pre-encrypted EScript property to the end of the pre-compiled Encryption property (the program will always call the EncryptData Procedure before calling SmallRecorder) and then call Recorder.

The four remaining Procedures in CEncryption are: DecryptScript, DecryptData, EncryptionUnlock, and Placer. All four of these procedures make up the FERD

DecryptScript can be thought of as the reverse of EncryptScript. It is very simple, containing only two Procedure Calls, the rest of its code being oriented toward the user interface. The two calls are to EncryptUnlock and Placer respectively.

DecryptData is very closely related to CProject's Decrypt Procedure. It sets the ENumber property equal to the value of the last four numbers of the Encryption property, and then removes the four numbers from the ciphertext (Encryption property). It then sets EncryptionBuilder's ENumber and Pass variables to the encryptions ENumber Property and the ProjectName and ProjectOwner (separated by a space) properties. From there on DecryptData decrypts most of the Encryption property into the following properties in their respective orders: BackupPassword, CreationDate, EDate, Name, Password, and Status. The EScript property is then set to equal the remainder of the Encryption property.

EncryptionUnlock is closely related to Writer in that it also utilizes six For Loops in order to access EncryptionArray. However, it is also related to Scramble in that it assigns data to the elements of EncryptionArray. EncryptionUnlock essentially has the reverse of PassCreator built into its first lines of code. It assigns EncryptionNum the value of the smallest number who when raised to the sixth power is equal to the length of the ciphertext divided by 4 (the ciphertext, EScript, is made up of numbers and is 4 times as long as the decrypted plaintext). It then ReDimensions EncryptionArray to fit the plaintext. From there on the six For Loops are used to assign the ciphertext to EncryptionArray.

Like the second For Loops in Scramble and Writer, the second For Loop in EncryptionUnlock uses the Num1 and Num2 variables to constantly update the progress bar on the Decrypter form in order to keep the user informed of the Encryption's progress. The second Loop in EncryptionUnlock deals with the first 50%, and the second Loop in Placer deals with the last 50%.

The sixth Loop in EncryptionUnlock is very simple. It reverses all of EncryptionArray's element's values, and if they are deemed to be random data, the element is emptied. The element's value is deemed random by whether or not it is greater than the product (the highest number attainable for the ciphertext) of the Encryption's ENumber property and 255. If it is found to be higher, it is deemed random.

CEncryption's final Procedure is called Placer. Placer is closely related to both Writer as it deals with the removal of data from EncryptionArray's elements, and to Scramble, in that they deal with the CSM and the order of their For Loops are the same. Like in Scramble, Placer decides what its Pass variable will be depending on the Encryption's Password property. After Pass is decided upon, Placer's remaining code is composed of the six For Loops.

The second Loop, just as in Scramble, Writer, and EncryptionUnlock, is used mainly to update the progress bar on the Decrypter form in order to keep the user informed of the Encryption's progress. This particular Loop deals with the last 50% of the bar.

Placer's sixth Loop deals with the last steps of the FERD. It subtracts the ASCII value of a character in Pass from the current element of EncryptionArray, and then divides the remaining value by the Encryption's ENumber property. The value of the element of EncryptionArray is then equal to some character in the ASCII table. The FERD's final step is to add that character to a textbox on the Decrypter form called Display.

The BASIC modules ProjectLoader and EncryptionBuilder play an important role in both how the program processes the various objects of CUser, CProject, and CEncryption; and in encryption and decryption of the objects' properties.

EncryptionBuilder contains four public variables and two sub procedures. The variables are called Encryptee and Pass (of type String), and EncrypteeLen and ENumber (of type Long). The Procedures are called Encrypt and Decrypt, and are essentially smaller versions of the FER and the FERD. They are smaller because of the fact that they do not utilize the Multi-Dimensional array in their routines. It was removed because of the fact that the routines are used to encrypt and decrypt individual properties, and if it were included, the resulting ciphertext would be unable to be decrypted once the program was shut down and the objects' plaintext properties were lost.

ProjectLoader can be thought of as one of the driving forces behind the program's structure. It is what imports the Users, Projects, and Encryptions from their files and creates their corresponding objects. It then adds the objects to Collections, which are essentially like dynamic arrays that hold objects in their elements. ProjectLoader also contains a very important string variable called Insat. When the program is run, the MainMenu form's Form_Load procedure opens a file on the C drive that was created by the Installer. It then retrieves the single piece of information the file contains and assigns it to Insat. The piece of information is the directory in which the program's important folders and files are located. Without Insat, The program would not be able to run because it would not be able to find any of the user-profiles, projects, or encryptions.

The Installer (support program) that I created is very simple and is composed of nothing more than a few procedures that create a folder and file structure that the main program can recognize in a directory decided upon by the user. This folder and file structure, though absolutely crucial to the main program, is relatively simple and small, consisting of a main folder called "Saati Encryption Center" which contains two folders called "Projects" and "Users". Within the "Users" folder is a file called "Registered Users" that contains the User's encrypted file (the ciphertext of the user's password). Within the "Projects" folder is a file called "Projects". This file contains the names and owners of all of the projects created by the main program. A more detailed picture of the folder and file structure can be found in Appendix A, Figure 2.

Results

The results from normal use of the program are represented in three ways. The primary output is through a textbox called Display on the Decrypter form. When the program is run successfully, the fully decrypted plaintext is displayed in the textbox. The plaintext is displayed because of the successful retrieval and decryption of the ciphertexts corresponding to the User, the User's Project, and the Project's contained Encryption.

The secondary form of output for the program is in the form of the interface. After all of the appropriate ciphertexts have been decrypted into their corresponding plaintexts, the interface uses many of the properties of all three classes to enrich the interface between the program and the user. For example, whether a project is open or closed is blatantly obvious in the form of words and a colored (red or green depending on whether it is open or closed) shape on the ProjectBrowser form. Also, properties such as the Users' EntryAttempts property are used to inform the user how many times they have entered the wrong password and how many tries they have left before their user-profile becomes locked out of the program.

The program's third form of output is through the encrypted files that it creates to store information about all of the Users, Projects, and Encryptions. In earlier forms of the program, the output files would be riddled with empty spaces or obviously too short for the message that was encrypted (an early error that was solved with the introduction of the NMR). However, in the finished program, the outputted files are all uniformly random in appearance, and appear to be completely encrypted because of the fact that they are essentially one long string of numbers with no obvious reference points.

From my research in cryptography methods, I learned that the best encryption would have absolutely no reference points from which to begin using the brute force method to decrypt it.

By adapting the Procedures in EncryptionBuilder to mimic the output of the FER, I was able to seamlessly combine the two routines’ output into a single string that had absolutely no reference points. Because of the proper meshing of the two Routines, that nice, even transition between their outputs was achieved. (Appendix A, Figure 4)

Analysis

From the Display textbox on the Decrypter form, one can easily view the contents of an Encryption. The results from many, many tests of the program show that there is absolutely NO difference between the original plaintext, and the current plaintext. Also, the outputted files all show consistency and uniformity with no gaps or prolonged occurrences of a single digit or patter. This shows that the program is completely accurate in all of its routines and procedures, which is a very important point to be extracted from the results.

In addition, the program's interface shows no imperfections or errors of any kind in regards to the individual parts that are tied directly to the properties and procedures of the classes, the BASIC module ProjectLoader, or the program's driving procedures and routines. This also verifies that the project is working perfectly.

These positive results indicate that my program is ready to become useful to anyone wishing to safely encrypt a message. Although I have only created a very basic version of the FER, the outputs that I have observed from the program indicate that the code is sound and, rather than being changed, only needs to be improved upon.

Overall, it would appear that the outputs from the program coincide with those of cryptologists' theoretical knowledge. These results, though not often truly difficult to achieve in the field of cryptography, offer hope that this encryption routine might one day be powerful enough to use in businesses and militaries around the world. However, while my program would be more than sufficient enough to safely encrypt anyone's private messages, however, there is always room for improvement.

Conclusions

There are many conclusions that I am able to draw from the results I achieved, especially because most of the results indicate that the routines are all working properly. One of the major conclusions that I arrived at after reviewing all of the results is that a Multi-Dimensional encryption Routine can be successfully used in an encryption algorithm in order to completely scramble the plaintext into an unintelligible ciphertext. I also concluded that the “Keyword” version of the Caesar Shift method is a very versatile encryption method that, though based on an incredibly simple principle, is still very powerful. Another conclusion which I arrived at is that the encrypting power of an encryption algorithm can be significantly augmented by the addition of a routine that converts the characters of the plaintext into numbers and then arbitrarily manipulates their values so as to create completely unrelated values. However, the greatest conclusion that I was able to extract from the results was that the combination of a Multi-Dimensional encryption routine, the “Keyword” version of the Caesar Shift method, and a routine for manipulating the numbers behind the data in the plaintext can create an incredibly powerful encryption tool that could stump even the most adept hackers.

Project Achievements

The most rewarding achievement of my project in the field of research that I chose was the working Encryption algorithm that I was able to create. Even though the algorithm is still in its infancy developmentally, it is suitable to use in my fairly basic program. It was important to realize that I was trying to create an algorithm that would be the basis for future algorithms to come. Because of this, I tried to keep the algorithm as simple as possible while still using all three of the principle encryption methods that I chose.

When speaking in terms of Computer Science, the project's most important achievements would have to be the methods by which I managed to create the FER algorithm and that allow for easy expansion of the project. While developing the code for the prototype program, I kept it very modularized, so as to make it easy to not only find and trap errors, but to add to and edit the program. After I eventually created the three classes, I continued to keep the code modularized and so continued to keep the code easy to read and expand upon. When someone decides that they want to build upon the original algorithm that I have created, they will be able to do so with the utmost ease and the smallest chance of causing errors in the rest of the program. In a case such as this, where the complexity and power of the algorithm depend only on how much time the programmer has, this approach becomes extremely important.

A second achievement of the project in Computer Science was the Multi-Dimensional encryption routine. This was a routine that I came up with after being inspired by the many different types of 2-Dimensional scrambling methods that I researched. It forms the backbone of the program and offers unprecedented scrambling capabilities. It is also very important because of its versatility. It can be molded to an infinites number of dimensions, and there is essentially no limit to the number of elements in the array itself, so there would be no limit to what it could encrypt. This routine was very important, not to mention effective.

Recommendations

Because of any encryption algorithm's inherent volatility and tendency to become more advanced over time, the program could have an innumerable amount of different methods and routines added to it. Not to mention the number of properties that could be added to the classes and used by the program to both enrich the interface and provide even deeper layers of security. However, the more properties and routines that are implemented, the more system resources would be taken up by the encryption routine. Eventually, if the routine became too complex, it could cause a slower computer to crash. So, if I, or anyone else for that matter, decide to expand upon the program, I should insert code that would perhaps store part of the array to file so as to save on the amount of resources that the computer would be using (Appendix A, Figure 5).

The results that I extracted from the program's various outputs indicate that the code is sound, but the interface is still fairly basic. If I were allowed more time, I would likely leave the encryption routines as they are and focus on the task of improving the user interface. Perhaps after adding a form from which the user could change things about their user profile, I would add the EntryAttempts and LockedOut properties to CProject and CEncryption. The point of these properties is to prevent hackers from using the brute force method to break into someone's account. Currently, a user is allowed three wrong password entries before the program locks the user out. When a user is locked out, they simply have to restart the program and try again. This would certainly deter hackers who were attempting to use the brute force method, but it is not a terribly strong deterrent, actually it's more of just a nuisance because they simply have to restart the program and can try again. So, if I had more time, I would also expand upon those two properties so that they were included in the encrypted ciphertext for each Project and Encryption, that way, I could use them in conjunction with the EDate property to lock a user out for a length of time, such as an hour or even a whole day. Another approach would be to have a single user (perhaps the first to use the program) register as the program's administrator so that when a User became locked out, the Administrator would be required to enter their password in order to "unlock" the user.

Yet another thing that I could work on would be to convert the plaintext textboxes on the Encrypter and Decrypter forms into rich textboxes so as to allow the user a way to encrypt a specially formatted document, such as a résumé. This would simply entail doing some work on the user interface and the addition of some of the rich textbox's properties into the CEncryption Class. After that, all I would have to do is add those new properties to CEncryption's EncryptData and DecryptData Procedures.

Eventually I would hope that all of this and more will be added to the program so that the user will find it as comfortable to use as some of Microsoft's other applications such as Microsoft Word. However, if all of these things were to happen, then the time and resource demands of the program would mean that everyone using it would have to be running it on supercomputers (or at least computers more powerful than the home PCs that I have been testing it on).

Acknowledgements and Citations

I would like to thank Jim Mims, my teacher and project mentor, for all of his time and effort. Before the completion of this project, I might not have been able to handle such a large program and project. Now, because of his help and guidance, I am able to do many of things with much greater proficiency and speed. He has taught me many new ways to approach problems and challenges that have been helpful with this project, as well as outside of it.

I would also like to extend thanks to Kelvin Smalls, my father. His invaluable time and patience has helped me pull the project out of the gutter countless times. I would also like to thank Mr. Yasuaki Nagatomo, my Shotokan Karate Sensei, whose outlook on problem solving and on life in general was a constant inspiration for me.

Finally, I would like to thank all of my many friends here at Albuquerque Academy and elsewhere that supported and encouraged me throughout the entire process. I would not have made it without them.

Citations

Books

Kippenhahn, Rudolf, *CODE BREAKING   A HISTORY AND EXPLORATION*, Peter Mayer Publishers Inc., The Overlook Press, March 1999, 326 pages

Menezes, Alfred J., van Oorschot, Paul C., and Vanstone, Scott A., *The Handbook of Applied Cryptography,* CRC Press, October 1996, 816 pages

Singh, Simon, *THE CODE BOOK*, Random House Inc., Doubleday, New York, August 2000, 432 pages

Websites

Why Cryptography Is Harder Than It Looks
<http://www.schneier.com/essay-037.html>

The Cryptography Introduction and Guide
<http://www.cryptographyworld.com/ >

Computer Generated Random Numbers
< http://world.std.com/~franl/crypto/random-numbers.html>

Center For TECHNOLOGY & DEMOCRACY
< http://www.cdt.org/crypto/>

Cryptography World
< http://www.cryptographyworld.com/>

International Association for Cryptologic Research (IACR)
< http://www.iacr.org/>

RSA Laboratories
< http://www.rsasecurity.com/rsalabs/node.asp?id=2152>

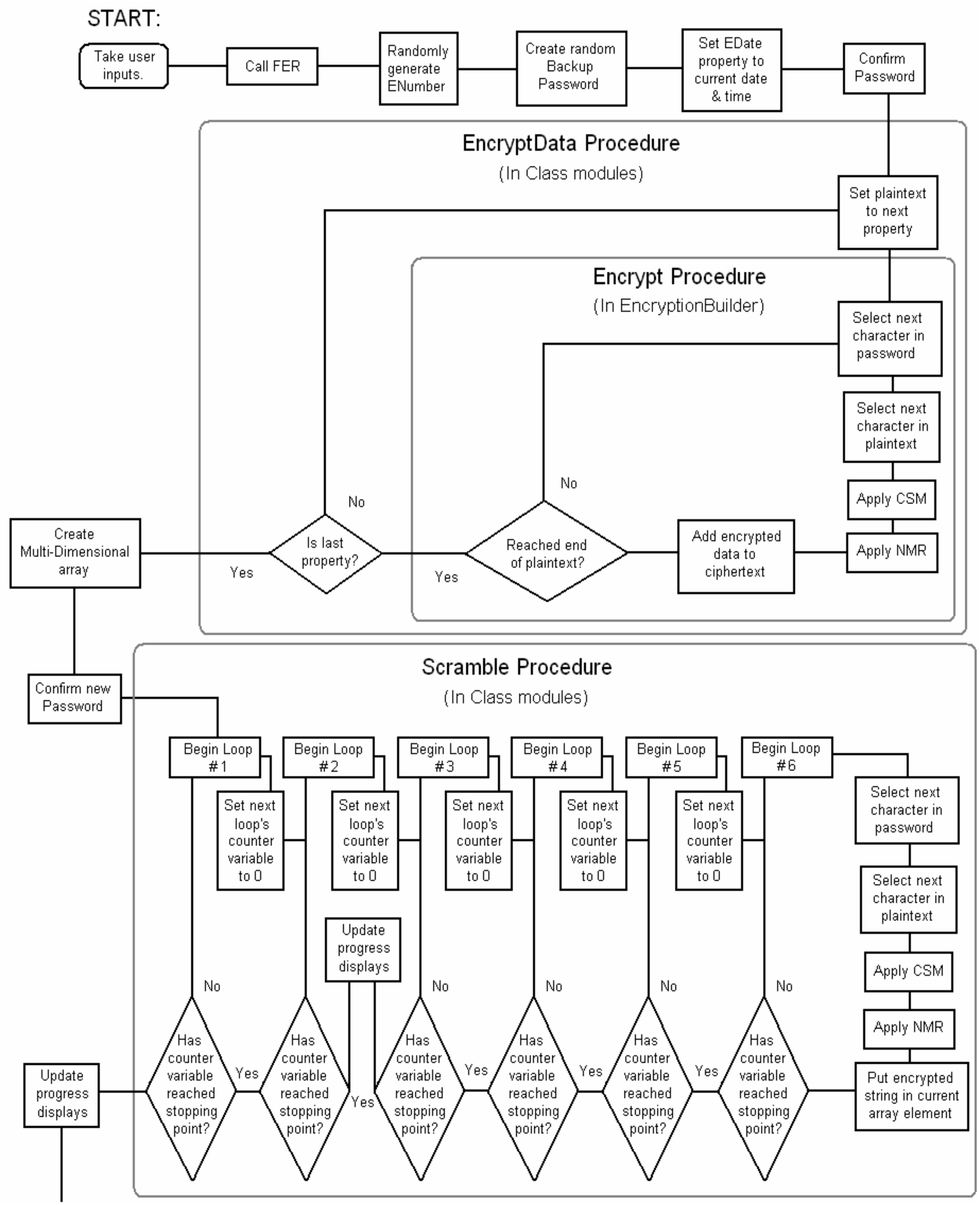
Cryptography and Security
< http://theory.lcs.mit.edu/~rivest/crypto-security.html>

Quadraylay's Cryptography Archive
< http://www.austinlinks.com/Crypto/>

Microsoft Research, Cryptography
< http://research.microsoft.com/crypto/>

Information on Cryptography
< http://www.cs.berkeley.edu/~daw/crypto.html>

Appendix A: Report Figures


START:
Take user inputs.
Call FER
Randomly generate ENumber
Create random Backup Password
Set EDate property to current date & time
Confirm Password
EncryptData Procedure
(In Class modules)
Set plaintext to next property
Encrypt Procedure
(In EncryptionBuilder)
Select next character in password
Select next character in plaintext
Apply CSM
Apply NMR
Add encrypted data to ciphertext
Reached end of plaintext?
No
Yes
Is last property?
No
Yes
Create Multi-Dimensional array
Confirm new Password
Scramble Procedure
(In Class modules)
Begin Loop #1
Set next loop's counter variable to 0
Begin Loop #2
Set next loop's counter variable to 0
Begin Loop #3
Set next loop's counter variable to 0
Begin Loop #4
Set next loop's counter variable to 0
Begin Loop #5
Set next loop's counter variable to 0
Begin Loop #6
Select next character in password
Select next character in plaintext
Apply CSM
Apply NMR
Put encrypted string in current array element
Update progress displays
Has counter variable reached stopping point?
No
Yes
Update progress displays

Continued on next page

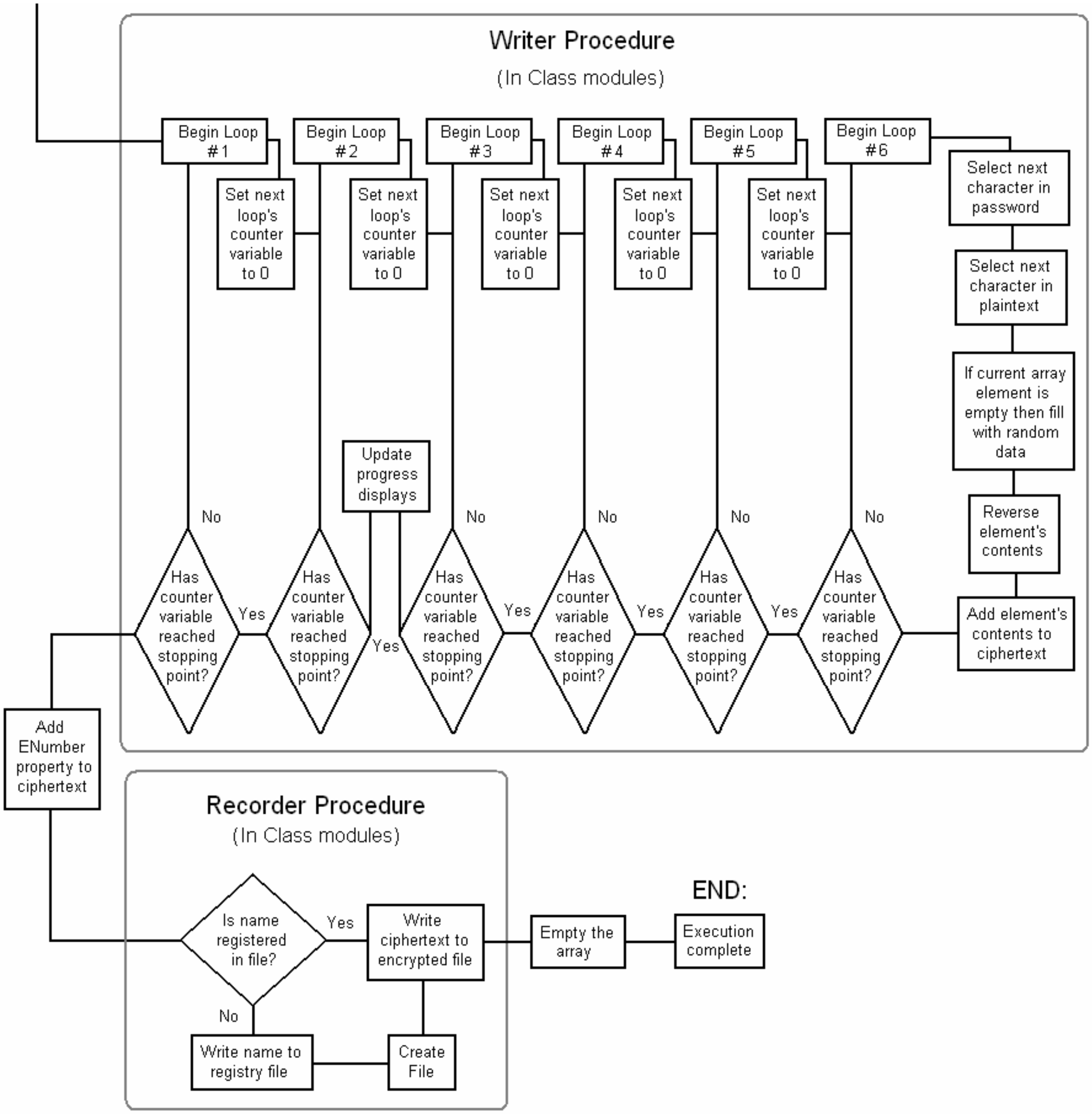

Writer Procedure
(In Class modules)
Begin Loop #1
Set next loop's counter variable to 0
Begin Loop #2
Set next loop's counter variable to 0
Begin Loop #3
Set next loop's counter variable to 0
Begin Loop #4
Set next loop's counter variable to 0
Begin Loop #5
Set next loop's counter variable to 0
Begin Loop #6
Select next character in password
Select next character in plaintext
If current array element is empty then fill with random data
Reverse element's contents
Add element's contents to ciphertext
Update progress displays
No
Has counter variable reached stopping point?
Yes
Has counter variable reached stopping point?
Yes
No
Has counter variable reached stopping point?
Yes
No
Has counter variable reached stopping point?
Yes
No
Has counter variable reached stopping point?
Yes
No
Has counter variable reached stopping point?
Add ENumber property to ciphertext
Recorder Procedure
(In Class modules)
Is name registered in file?
Yes
No
Write ciphertext to encrypted file
Write name to registry file
Create File
Empty the array
END:
Execution complete

Figure 1: A simplified flowchart of the Full Encryption Routine (FER)

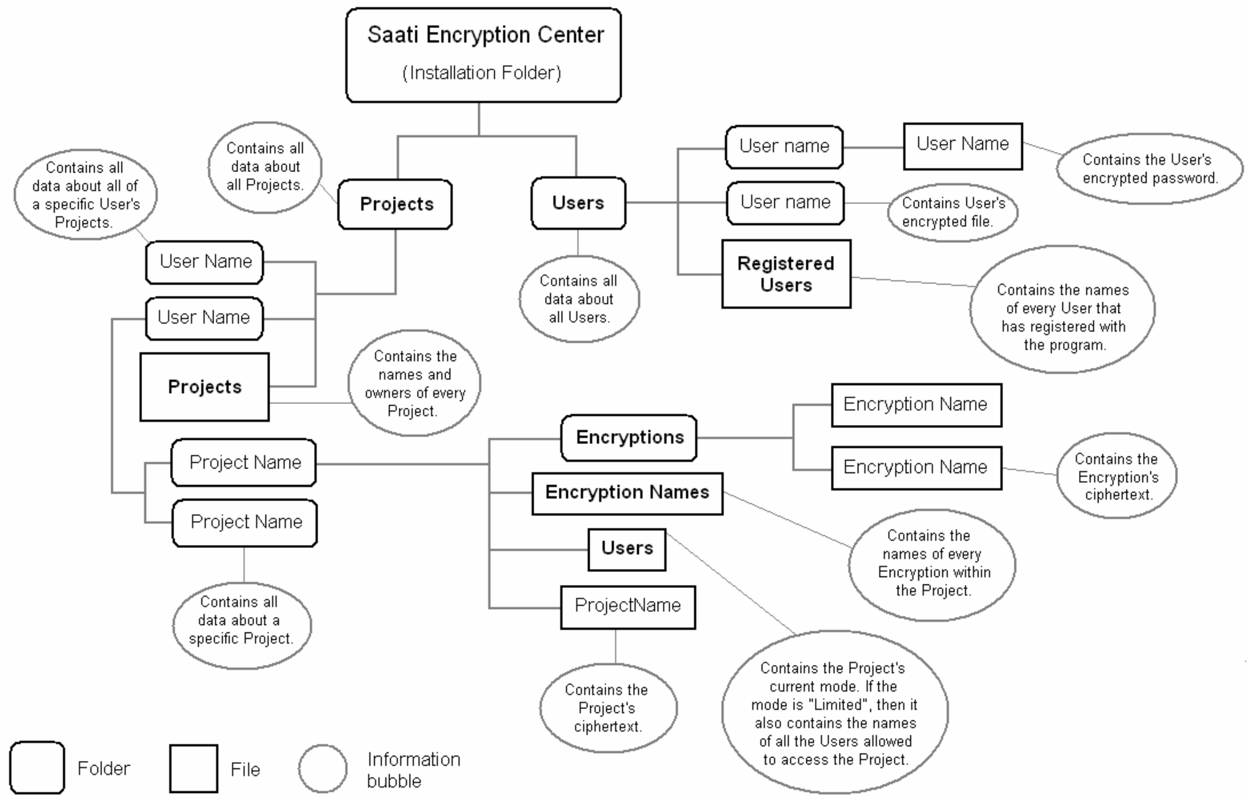

Saati Encryption Center
(Installation Folder)
Contains all data about all Projects.
Projects
Users
User name
User name
User Name
Contains the User's encrypted password.
Contains User's encrypted file.
Registered Users
Contains the names of every User that has registered with the program.
Contains all data about all Users.
Contains all data about all of a specific User's Projects.
User Name
User Name
Projects
Contains the names and owners of every Project.
Project Name
Project Name
Contains all data about a specific Project.
Encryptions
Encryption Names
Users
ProjectName
Encryption Name
Encryption Name
Contains the Encryption's ciphertext.
Contains the names of every Encryption within the Project.
Contains the Project's ciphertext.
Contains the Project's current mode. If the mode is "Limited", then it also contains the names of all the Users allowed to access the Project.
Folder
File
Information bubble

Figure 2: A visual representation of the program's File and Folder system. Please Note: bold type represents a file/folder's actual name; normal type means that the name is user generated.

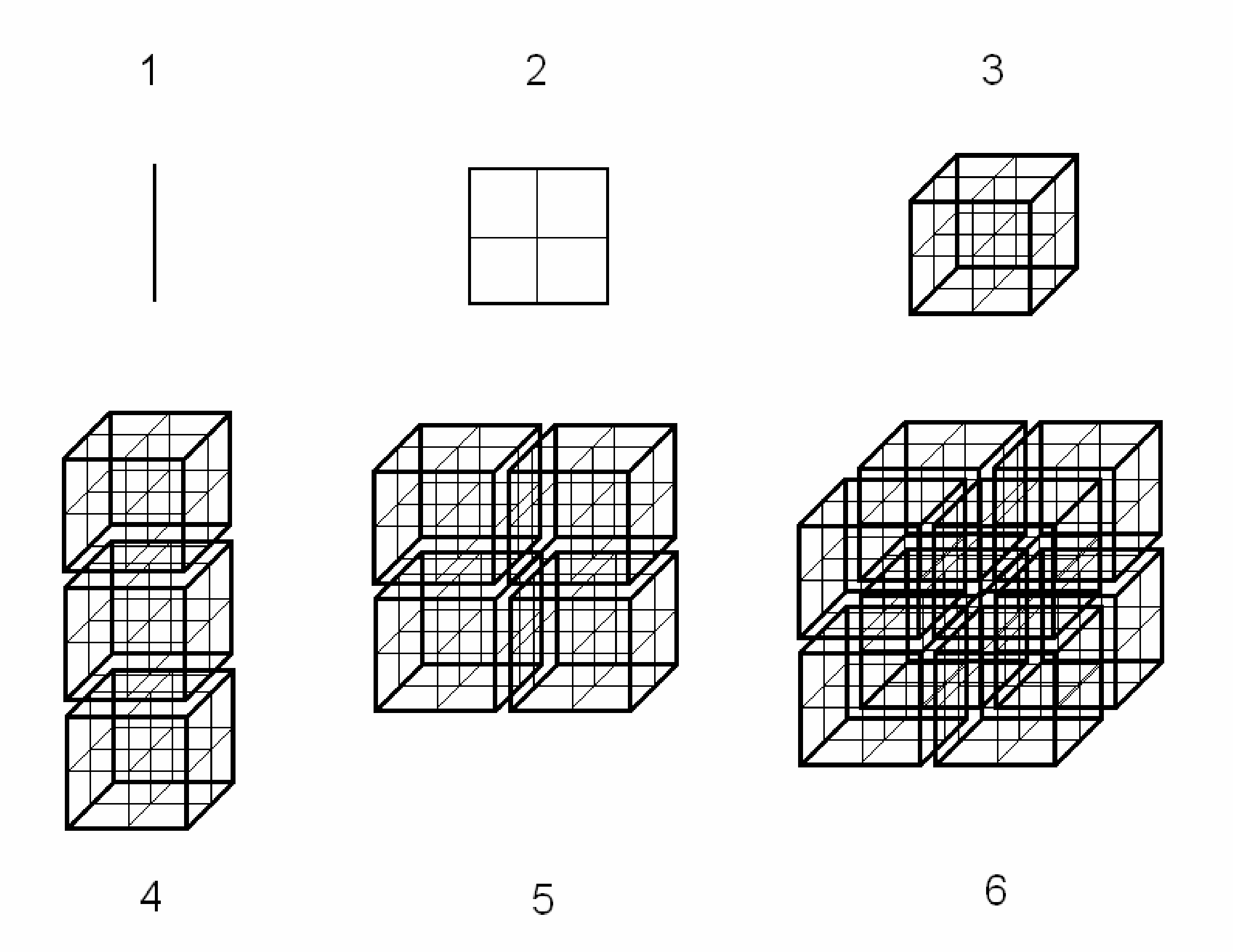


Figure 3: A visual representation of six dimensions and their relative “cube” structures. The sixth dimensional “cube” represents the Multi-Dimensional array used in the program.

Sample Program Input: Gettysburg Address excerpt

Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Program's Output:

```
6300311150012583775300235892722458112052739326515910922235528370318125523390716158112070128222733372777356224572551281149051915071616
5523424677379710200159032018190319041903390159095018460319068901311320195016111139022013660725137410200159032018190319041903390159095
0184603190689013111401789061111390698036607251374171005631509106127312012215813912931262915260202178817881931209819112201260003102149
1430272228591338180007161760261027402418131025091788176313389299679604038936798811258198876815584483759129036631525373686068859127116
9655911217846227263993271618939112899076988135269238551227688128358132881836939102885355414945678547191219687719386072777825688631995
8355291258866877960123956564938634624879112479863585912508588464168117828297802159529577960894572182107757745563712847637075912036565
7976810797092838601329304979600989683651899588281991129056534677125178586526497106645678023828235930916729862815023995460993910128558
9988184375558615502164098501264381159768143959726844653790057992296464697386037358485183567840129979126760987902275689646629593970786
7960927792457802220646680915797503379881201994571012416912757091803615193712718576057091016922867286108522775912286734691102022872250
2373879867619120188633513223916222655305169117638604447242893915318200940383259954911027826128835810836794946894298587595124728583858
8167491676040949563549759114888638386085684659843684185397780246865379371246282698890617365075101289495856780286057725551212694385169
1357913851822756651690889893835865791532702283212599462478499396805993860456599181012105510943122172955471912226977787091019693680959
0199048638609035327519120767289920978559200938029999904779916948536566940537150931226729381510123945522942659916287719124528406796026
0594847033740869448386025863646399107682436591294194877101253174594358111863289929644796796780263283495399116453676721830891815309101
9543765291405687871390152552389512233910889481864636496038700678973802271686355912561821376457136928575291708991491741051835062387593
5479419121295565797914749959928574655328735811428572897606587247941388137770910124617339899808747189897607857935838607137426997608546
6565341587680227579173754939501259496756886868098966529111863797129119375219782078786846768137942837709122260397612537695307780219490
9257402677638595016311949253712179522365291925571552527822822695012823667089112085758375526913952271291529920855291229589176685229574
9651221146218734815777273572912365733759028168878510902066065800394357282758604236175759028838896859865518091634812167017898029238165
5348164956305189191452385789103066358175553657317978185942508189197787648840649261565586075288185770269652447043921157266159199493735
5860135683391257785919451090268937895191615784384947669993987612514522487012113683385191421776845122793859487702513925077777003930263
7603649441779128418687923999549197658600968693978915638851958601265297776125665110579123179983666494246745979122626058737602428613964
0867652256348196591387999127958597778980891908586056690267519127291417620687384487376081050317558138667639172776962929161257760875376
0895901287291733632193391670514569091697691275058745927559991914941677702196694269756345970561222147607569091064685753539732916491612
2626373637600148754674279618415551918598592935029855400987379629097558602239449494123386125718666946511794120168012916126858681968590
26612361522341678283760010705665907850665970018
```

Figure 4: Sample input and output from the program. Please note that due to the size of the input most of the output in this example is random data created by the computer to fill the empty elements within the array.

Largest value that a long variable can hold = 2,147,483,647

2,147,483,647 ^ 6 = 9.80797143413853 X 10 ^ 55 = maximum number of elements in the six-dimensional array.

3.45811922179568 X 10 ^ 44 Terabytes = maximum output file size from the program.

Figure 5: Miscellaneous figures about the six-dimensional array and program.

Appendix B: CEncryption Code

‘Code written in Microsoft Visual Basic 6.0, line comment character is ‘

‘Property and Global variable Dimensioning

```
Public Name As String 'Encryption's name
Public ProjectName As String 'Name of the Encryption's project
Public ProjectOwner As String 'Name of the owner of the Encryption's project
Public Password As String 'User created password
Public BackupPassword As String 'Randomly generated backup password
Public CreationDate As String 'Date that the Encryption was created
Public EDate As String 'Date that the Encryption was last changed or edited
Public ENumber As Integer 'Used in the FER
Public Encryption As String 'stores the ciphertext while the encryption is not being used
Public EScript As String 'Stores the plaintext while the encryption is not being used
Public Status As String 'States whether the Encryption is "locked" or "Unlocked"
Dim EncryptionNum As Long 'Used in the FER
Dim EncryptionArray() As String ' The multi-dimensional array


Public Sub EncryptScript()
   Randomize
   ENumber = Int(Rnd * 38) + 1 'Creates random encryption number
   e = Int(Rnd * 30) + 1 'Sets length for Backup Password
   For i = 1 To e 'Creates random backup password
      BackupPassword = BackupPassword & Chr(Int(Rnd * 244) + 1)
   Next i
   Call EncryptData
   Call PassCreator
   If Encrypter.Visible = True Then Encrypter.Progress.Caption = "Loading Data" 'Updates progress displays
   If Decrypter.Visible = True Then Decrypter.Progress.Caption = "Loading Data" 'Ditto
   Call Scramble
   If Encrypter.Visible = True Then Encrypter.Progress.Caption = "Recording Data" 'Ditto
   If Decrypter.Visible = True Then Decrypter.Progress.Caption = "Recording Data" 'Ditto
   Call Writer
   Call Recorder
   If Encrypter.Visible = True Then Encrypter.Progress.Caption = "Complete!" 'Ditto
   If Decrypter.Visible = True Then Decrypter.Progress.Caption = "Complete!" 'Ditto
   If Encrypter.Visible = True Then Encrypter.Percent.Caption = "100%" 'Ditto
   If Decrypter.Visible = True Then Decrypter.Percent.Caption = "100%" 'Ditto
   ReDim EncryptionArray(0) As String
End Sub


Public Sub EncryptData()
   Dim L As String * 4 'Creates fixed length string 4 characters long
   Encryption = ""
   EDate = Format(Now, "mm/dd/yy") & " " & Format(Now, "hh:mm:ss AM/PM") 'Sets Edate to current date and time separated by a
space
   EncryptionBuilder.ENumber = ENumber
   EncryptionBuilder.Pass = ProjectName & " " & ProjectOwner 'Creates password for property encryption

   EncryptionBuilder.EncrypteeLen = Len(BackupPassword)
   L = StrReverse(LengthCheck(Len(BackupPassword))) 'Hides length of BackupPassword
   Encryption = L
   EncryptionBuilder.Encryptee = BackupPassword
   Call EncryptionBuilder.Encrypt
   Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted BackupPassword to ciphertext

   EncryptionBuilder.EncrypteeLen = Len(CreationDate)
   L = StrReverse(LengthCheck(Len(CreationDate))) 'Hides length of CreationDate
   Encryption = Encryption & L
   EncryptionBuilder.Encryptee = CreationDate
   Call EncryptionBuilder.Encrypt
   Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted CreationDate to ciphertext

   EncryptionBuilder.EncrypteeLen = Len(EDate)
```

```
    L = StrReverse(LengthCheck(Len(EDate))) 'Hides length of EDate
    Encryption = Encryption & L
    EncryptionBuilder.Encryptee = EDate
    Call EncryptionBuilder.Encrypt
    Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted EDate to ciphertext

    EncryptionBuilder.EncrypteeLen = Len(Name)
    L = StrReverse(LengthCheck(Len(Name))) 'Hides length of Name
    Encryption = Encryption & L
    EncryptionBuilder.Encryptee = Name
    Call EncryptionBuilder.Encrypt
    Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted Name to ciphertext

    EncryptionBuilder.EncrypteeLen = Len(Password)
    L = StrReverse(LengthCheck(Len(Password))) 'Hides length of Password
    Encryption = Encryption & L
    EncryptionBuilder.Encryptee = Password
    Call EncryptionBuilder.Encrypt
    Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted Password to ciphertext

    EncryptionBuilder.EncrypteeLen = Len(Status)
    L = StrReverse(LengthCheck(Len(Status))) 'Hides length of Status
    Encryption = Encryption & L
    EncryptionBuilder.Encryptee = Status
    Call EncryptionBuilder.Encrypt
    Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted Status to ciphertext
End Sub


Private Sub PassCreator() 'Creates best-fit array
    Dim i As Long
    i = 1
    Do
        i = i + 1
    Loop Until i ^ 6 >= Len(EScript)
    If i = 1 Then i = 2
    ReDim EncryptionArray(i, i, i, i, i, i) As String 'Creates array
    EncryptionNum = i
End Sub


Private Sub Scramble()
    Dim i As Long, e As Long, f As Long, g As Long, X As Long, Y As Long, z As Long, n As Long
    Dim Pass As String
    Dim Letter As String * 4
    Dim Num1 As Double, Num2 As Double
    Num1 = 50 / (EncryptionNum ^ 2)
    If Password = "" Then 'Decides whether to use Password or BackupPassword
        Pass = BackupPassword
    Else:
        Pass = Password
    End If
    z = 1
    For Y = 0 To EncryptionNum - 1
        X = 0
        For X = 0 To EncryptionNum - 1
            g = 0
            For g = 0 To EncryptionNum - 1
                f = 0
                For f = 0 To EncryptionNum - 1
                    i = 0
                    For i = 0 To EncryptionNum - 1
                        e = 0
                        For e = 0 To EncryptionNum - 1
                            If n = Len(Password) Then 'Selects next character in password
                                n = 1
                            Else:
                                n = n + 1
                            End If
                            Letter = (Asc(Mid(EScript, z, 1)) * ENumber) + Asc(Mid(Password, n, 1)) 'Encrypts data via the NMR
                            EncryptionArray(Y, X, g, f, i, e) = LengthCheck(Letter) 'Assigns encrypted data to current array element
                            z = z + 1
                            If z > Len(EScript) Then Exit Sub
                        Next e
                    Next i
                Next f
            Next g
            Num2 = Num2 + Num1 'Updates progress displays
            If Encrypter.Visible = True Then
                Encrypter.PB1.Value = Num2
                Encrypter.Percent.Caption = Round(Num2, 2) & "%"
            End If
            If Decrypter.Visible = True Then
                Decrypter.PB1.Value = Num2
                Decrypter.Percent.Caption = Round(Num2, 2) & "%"
            End If
        Next X
    Next Y
End Sub


Private Sub Writer()
    Dim i As Long, e As Long, f As Long, g As Long, X As Long, Y As Long
    Dim Num1 As Double, Num2 As Double
    Num1 = 50 / (EncryptionNum ^ 2)
    Num2 = 50
    Randomize
    For e = 0 To EncryptionNum - 1
        i = 0
        For i = 0 To EncryptionNum - 1
            f = 0
            For f = 0 To EncryptionNum - 1
                g = 0
                For g = 0 To EncryptionNum - 1
                    X = 0
                    For X = 0 To EncryptionNum - 1
                        Y = 0
                        For Y = 0 To EncryptionNum - 1
                            If EncryptionArray(Y, X, g, f, i, e) = "" Then EncryptionArray(Y, X, g, f, i, e) = (Int(Rnd * (9999 - ((ENumber + 1) * 255))) + 1) + ((ENumber + 1) * 255) 'If element is empty, assigns random data to it
                            EncryptionArray(Y, X, g, f, i, e) = StrReverse(EncryptionArray(Y, X, g, f, i, e)) 'Reverses current element
                            Encryption = Encryption & EncryptionArray(Y, X, g, f, i, e)
                        Next Y
                    Next X
                Next g
            Next f
            Num2 = Num2 + Num1 'Updates progress displays
            If Encrypter.Visible = True Then
                Encrypter.PB1.Value = Num2
                Encrypter.Percent.Caption = Round(Num2, 2) & "%"
            End If
            If Decrypter.Visible = True Then
                Decrypter.PB1.Value = Num2
                Decrypter.Percent.Caption = Round(Num2, 2) & "%"
            End If
        Next i
    Next e

    Encryption = Encryption & LengthCheck(ENumber) 'Adds ENumber to ciphertext
End Sub


Private Sub Recorder()
    Dim Good As Boolean
    Dim Temp As String
    Open (Insat & "Projects\" & ProjectOwner & "\" & ProjectName & "\Encryption Names") For Input As #1
        Good = True
        Do While Not EOF(1) 'Checks to see if Encryption already exists
            Input #1, Temp
            If Temp = Name Then Good = False
        Loop
    Close #1
```

```
   If Good = True Then 'If Encryption is new, registers name
      Open (Insat & "Projects\" & ProjectOwner & "\" & ProjectName & "\Encryption Names") For Append As #2
         Write #2, Name
      Close #2
   End If
   Open (Insat & "Projects\" & ProjectOwner & "\" & ProjectName & "\Encryptions\" & Name) For Output As #1
      Write #1, Encryption 'Writes ciphertext to file
   Close #1
End Sub


Public Sub SmallRecorder()
   Encryption = Encryption & EScript 'adds encrypted message to encrypted properties to create ciphertext
   Encryption = Encryption & LengthCheck(ENumber) 'Adds ENumber to cphertext
   Call Recorder
End Sub







Public Sub DecryptScript()
   Decrypter.Progress.Caption = "Reading Data" 'Updates progress displays
   Call EncryptionUnlock
   Decrypter.Progress.Caption = "Loading Data" 'Ditto
   Call Placer
   Decrypter.Progress.Caption = "Complete!" 'Ditto
   Decrypter.Percent.Caption = "100%" 'Ditto
   Decrypter.PB1.Value = 100 'Ditto
End Sub

Public Sub DecryptData()
   Dim Temp As String
   ENumber = Val(Right(Encryption, 4)) 'Extracts ENumber from ciphertext
   Encryption = Left(Encryption, Len(Encryption) - 4) 'Removes ENumber info. from ciphertext
   EncryptionBuilder.ENumber = ENumber
   EncryptionBuilder.Pass = ProjectName & " " & ProjectOwner 'Creates password for property decryption

   EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of BackupPassword
   EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
   Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
   Call EncryptionBuilder.Decrypt
   BackupPassword = EncryptionBuilder.Encryptee

   EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of CreationDate
   EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
   Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
   Call EncryptionBuilder.Decrypt
   CreationDate = EncryptionBuilder.Encryptee

   EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of EDate
   EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
   Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
   Call EncryptionBuilder.Decrypt
   EDate = EncryptionBuilder.Encryptee

   EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of Name
   EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
   Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
   Call EncryptionBuilder.Decrypt
   Name = EncryptionBuilder.Encryptee

   EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of Password
   EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
```

```
    Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
    Call EncryptionBuilder.Decrypt
    Password = EncryptionBuilder.Encryptee

    EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of Status
    EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
    Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
    Call EncryptionBuilder.Decrypt
    Status = EncryptionBuilder.Encryptee

    EScript = Encryption 'assigns remaining ciphertext to EScript property
End Sub


Private Sub EncryptionUnlock()
    Dim i As Long, e As Long, f As Long, g As Long, X As Long, Y As Long, z As Long, n As Long
    Dim Num1 As Double, Num2 As Double
    Dim Letter As String * 4
    z = 1
    Do 'Creates array dimensioning variable
        n = n + 1
    Loop Until (n ^ 6) >= Len(EScript) / 4
    EncryptionNum = n
    ReDim EncryptionArray(n, n, n, n, n, n) As String 'Creates best fit array
    Num1 = 50 / (n ^ 2)
    For e = 0 To n - 1
        i = 0
        For i = 0 To n - 1
            f = 0
            For f = 0 To n - 1
                g = 0
                For g = 0 To n - 1
                    X = 0
                    For X = 0 To n - 1
                        Y = 0
                        For Y = 0 To n - 1
                            EncryptionArray(Y, X, g, f, i, e) = StrReverse(Mid(EScript, z, 4)) 'Assigns next piece of ciphertext to current array element
                            If Val(EncryptionArray(Y, X, g, f, i, e)) > (ENumber + 1) * 255 Then EncryptionArray(Y, X, g, f, i, e) = "" 'If element is
random data, empties it
                            z = z + 4
                        Next Y
                    Next X
                Next g
            Next f
            Num2 = Num2 + Num1 'Updates progress displays
            Decrypter.PB1.Value = Num2
            Decrypter.Percent.Caption = Round(Num2, 2) & "%"
        Next i
    Next e
End Sub


Private Sub Placer()
    Dim i As Long, e As Long, f As Long, g As Long, X As Long, Y As Long
    Dim t As Integer
    Dim Num1 As Double, Num2 As Double
    Dim Pass As String
    If Password <> "" Then 'Decides whether to use Password or BackupPassword
        Pass = Password
    Else:
        Pass = BackupPassword
    End If
    Num1 = 50 / (EncryptionNum ^ 2)
    Num2 = 50
    For Y = 0 To EncryptionNum - 1
        X = 0
        For X = 0 To EncryptionNum - 1
            g = 0
            For g = 0 To EncryptionNum - 1
```

```
                f = 0
                For f = 0 To EncryptionNum - 1
                    i = 0
                    For i = 0 To EncryptionNum - 1
                        e = 0
                        For e = 0 To EncryptionNum - 1
                            If EncryptionArray(Y, X, g, f, i, e) <> "" Then
                                If t = Len(Pass) Then 'Selects next character in password
                                    t = 1
                                Else:
                                    t = t + 1
                                End If
                                EncryptionArray(Y, X, g, f, i, e) = Chr((Val(EncryptionArray(Y, X, g, f, i, e)) - Asc(Mid(Pass, t, 1))) / ENumber) 'Decrypts
data via the NMR
                                Decrypter.TextDisplay.Text = Decrypter.TextDisplay.Text & EncryptionArray(Y, X, g, f, i, e) 'Adds element's contents to
plaintext
                            End If
                        Next e
                    Next i
                Next f
            Next g
            Num2 = Num2 + Num1 'Updates progress displays
            Decrypter.PB1.Value = Num2
            Decrypter.Percent.Caption = Round(Num2, 2) & "%"
        Next X
    Next Y
End Sub
```

Appendix C: CProject Code

```
'Property and Global variable dimensioning

Public Name As String 'Project's name
Public Password1 As String '1st password
Public Password2 As String '2nd optional password
Public EType As String 'Project's mode (Public, Limited, or Private)
Public CreationDate As String 'Date the Project was created
Public EDate As String 'Date that the Project was last changed or edited
Public ENumber As Integer 'Random number used in encryption and decryption
Public Owner As String 'Name of the user that created the project
Public Status As String 'States whether the project is "Open" or "Closed"
Public Encryption As String 'Stores the ciphertext while the Project is not being used


Public Sub Encrypt()
   On Error Resume Next 'Error handler
   Dim fso As New FileSystemObject
   Dim i As Integer
   EDate = Format(Now, "mm/dd/yy") & " " & Format(Now, "hh:mm:ss AM/PM") 'Sets EDate to current date and time separated by a
space
   fso.CreateFolder (Insat & "Projects\" & Owner & "\" & Name) 'Creates necessary folders and file
   fso.CreateFolder (Insat & "Projects\" & Owner & "\" & Name & "\Encryptions")
   Open (Insat & "Projects\" & Owner & "\" & Name & "\Encryption Names") For Append As #1
   Close #1
   Call SmallEncrypter
End Sub


Private Sub SmallEncrypter()
   Dim L As String * 4
   Randomize
   Encryption = ""
   ENumber = Int(Rnd * 38) + 1 'Creates random encryption number
   EncryptionBuilder.ENumber = ENumber
   EncryptionBuilder.Pass = Owner

   EncryptionBuilder.EncrypteeLen = Len(CreationDate)
   L = StrReverse(LengthCheck(Len(CreationDate))) 'Hides length of CreationDate
   Encryption = Encryption & L
   EncryptionBuilder.Encryptee = CreationDate
   Call EncryptionBuilder.Encrypt
   Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted CreationDate to ciphertext

   EncryptionBuilder.EncrypteeLen = Len(EDate)
   L = StrReverse(LengthCheck(Len(EDate))) 'Hides length of EDate
   Encryption = Encryption & L
   EncryptionBuilder.Encryptee = EDate
   Call EncryptionBuilder.Encrypt
   Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted EDate to ciphertext

   EncryptionBuilder.EncrypteeLen = Len(EType)
   L = StrReverse(LengthCheck(Len(EType))) 'Hides length of EType
   Encryption = Encryption & L
   EncryptionBuilder.Encryptee = EType
   Call EncryptionBuilder.Encrypt
   Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted EType to ciphertext

   EncryptionBuilder.EncrypteeLen = Len(Password1)
   L = StrReverse(LengthCheck(Len(Password1))) 'Hides length of Password1
   Encryption = Encryption & L
   EncryptionBuilder.Encryptee = Password1
   Call EncryptionBuilder.Encrypt
   Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted Password1 to ciphertext

   EncryptionBuilder.EncrypteeLen = Len(Password2)
   L = StrReverse(LengthCheck(Len(Password2))) 'Hides length of Password2
```

```
    Encryption = Encryption & L
    EncryptionBuilder.Encryptee = Password2
    Call EncryptionBuilder.Encrypt
    Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted Password2 to ciphertext

    EncryptionBuilder.EncrypteeLen = Len(Status)
    L = StrReverse(LengthCheck(Len(Status))) 'Hides length of Status
    Encryption = Encryption & L
    EncryptionBuilder.Encryptee = Status
    Call EncryptionBuilder.Encrypt
    Encryption = Encryption & EncryptionBuilder.Encryptee 'Adds encrypted Status to ciphertext

    Open (Insat & "Projects\" & Owner & "\" & Name & "\" & Name) For Output As #1
        Write #1, Encryption & LengthCheck(ENumber)
    Close #1
End Sub


Public Sub Decrypt()
    ENumber = Val(Right(Encryption, 4)) 'Extracts ENumber from ciphertext
    Encryption = Left(Encryption, Len(Encryption) - 4) 'Removes ENumber info. from ciphertext
    EncryptionBuilder.ENumber = ENumber
    EncryptionBuilder.Pass = Owner 'Creates password for property decryption

    EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of CreationDate
    EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
    Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
    Call EncryptionBuilder.Decrypt
    CreationDate = EncryptionBuilder.Encryptee

    EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of EDate
    EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
    Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
    Call EncryptionBuilder.Decrypt
    EDate = EncryptionBuilder.Encryptee

    EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of EType
    EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
    Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
    Call EncryptionBuilder.Decrypt
    EType = EncryptionBuilder.Encryptee

    EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of Password1
    EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
    Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
    Call EncryptionBuilder.Decrypt
    Password1 = EncryptionBuilder.Encryptee

    EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of Password2
    EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
    Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
    Call EncryptionBuilder.Decrypt
    Password2 = EncryptionBuilder.Encryptee

    EncryptionBuilder.EncrypteeLen = Val(StrReverse(Left(Encryption, 4))) 'Extracts length of Status
    EncryptionBuilder.Encryptee = Right(Encryption, Len(Encryption) - 4)
    Encryption = Right(Encryption, Len(Encryption) - (4 * (EncryptionBuilder.EncrypteeLen + 1))) 'Removes used property info. from
ciphertext
    Call EncryptionBuilder.Decrypt
    Status = EncryptionBuilder.Encryptee
End Sub
```

Appendix D: CUser Code

```
'Property and public variable dimensioning

Public Name As String 'User's name
Public Password As String 'User's password
Public Encryption As String 'Stores the ciphertext when user is being encrypted/decrypted
Public ENumber As Integer 'Random number involved in encryption and decryption
Public EntryAttempts As Integer 'Number of times user has entered the wrong password
Public LockedDown As Boolean 'Determines whether the user can sign in or not
Dim EncryptionNum As Long 'Used in the FER
Dim EncryptionArray() As String ' The multi-dimensional array


Public Sub Encrypt()
  Randomize
  ENumber = Int(Rnd * 38) + 1 'Creates random encryption number
  Call PassCreator
  Call Scramble
  Call Writer
  Call Recorder
  ReDim EncryptionArray(0) As String
End Sub

Private Sub PassCreator() 'Creates best-fit array
  Dim i As Long
  i = 1
  Do
    i = i + 1
  Loop Until i ^ 6 >= Len(Password)
  If i = 1 Then i = 2
  ReDim EncryptionArray(i, i, i, i, i, i) As String 'Creates array
  EncryptionNum = i
End Sub

Private Sub Scramble()
  Dim i As Long, e As Long, f As Long, g As Long, X As Long, Y As Long, z As Long, n As Long
  Dim Letter As String * 4
  z = 1
  For Y = 0 To EncryptionNum - 1
    X = 0
    For X = 0 To EncryptionNum - 1
      g = 0
      For g = 0 To EncryptionNum - 1
        f = 0
        For f = 0 To EncryptionNum - 1
          i = 0
          For i = 0 To EncryptionNum - 1
            e = 0
            For e = 0 To EncryptionNum - 1
              If n = Len(Name) Then 'Selects next character in password
                n = 1
              Else:
                n = n + 1
              End If
              EncryptionArray(Y, X, g, f, i, e) = LengthCheck((Asc(Mid(Password, z, 1)) * ENumber) + Asc(Mid(Name, n, 1))) 'Encrypts
data via the NMR and assigns to current array element
              z = z + 1
              If z > Len(Password) Then Exit Sub
            Next e
          Next i
        Next f
      Next g
    Next X
  Next Y
End Sub
```

```
Private Sub Writer()
   Dim i As Long, e As Long, f As Long, g As Long, X As Long, Y As Long
   Randomize
   For e = 0 To EncryptionNum - 1
      i = 0
      For i = 0 To EncryptionNum - 1
         f = 0
         For f = 0 To EncryptionNum - 1
            g = 0
            For g = 0 To EncryptionNum - 1
               X = 0
               For X = 0 To EncryptionNum - 1
                  Y = 0
                  For Y = 0 To EncryptionNum - 1
                     If EncryptionArray(Y, X, g, f, i, e) = "" Then EncryptionArray(Y, X, g, f, i, e) = (Int(Rnd * (9999 - ((ENumber + 1) * 255))) + 1) + ((ENumber + 1) * 255) 'If element is empty, assigns random data to it
                     Encryption = Encryption & StrReverse(EncryptionArray(Y, X, g, f, i, e)) 'Reverses current element and adds it to ciphertext
                  Next Y
               Next X
            Next g
         Next f
      Next i
   Next e
   Encryption = Encryption & LengthCheck(ENumber) 'Adds ENumber to ciphertext
End Sub


Private Sub Recorder()
   Open (Insat & "Users\" & Name & "\" & Name) For Output As #1
      Write #1, Encryption 'Writes ciphertext to file
   Close #1
End Sub


Public Sub Decrypt()
   Call EncryptionUnlock
   Call Placer
End Sub


Private Sub EncryptionUnlock()
   Dim i As Long, e As Long, f As Long, g As Long, X As Long, Y As Long, z As Long, n As Long
   Dim Letter As String * 4
   z = 1
   ENumber = Val(Right(Encryption, 4))
   Encryption = Left(Encryption, Len(Encryption) - 4)
   Do 'Creates array dimensioning variable
      n = n + 1
   Loop Until (n ^ 6) >= Len(Encryption) / 4
   EncryptionNum = n
   ReDim EncryptionArray(n, n, n, n, n, n) As String 'Creates best fit array
   For e = 0 To n - 1
      i = 0
      For i = 0 To n - 1
         f = 0
         For f = 0 To n - 1
            g = 0
            For g = 0 To n - 1
               X = 0
               For X = 0 To n - 1
                  Y = 0
                  For Y = 0 To n - 1
                     EncryptionArray(Y, X, g, f, i, e) = StrReverse(Mid(Encryption, z, 4)) 'Assigns next piece of ciphertext to current array element
                     If Val(EncryptionArray(Y, X, g, f, i, e)) > (ENumber + 1) * 255 Then EncryptionArray(Y, X, g, f, i, e) = "" 'If element is random data, empties it
                     z = z + 4
                  Next Y
               Next X
            Next g
         Next f
```

```
        Next i
    Next e
End Sub


Private Sub Placer()
    Dim i As Long, e As Long, f As Long, g As Long, X As Long, Y As Long
    Dim t As Integer
    For Y = 0 To EncryptionNum - 1
        X = 0
        For X = 0 To EncryptionNum - 1
            g = 0
            For g = 0 To EncryptionNum - 1
                f = 0
                For f = 0 To EncryptionNum - 1
                    i = 0
                    For i = 0 To EncryptionNum - 1
                        e = 0
                        For e = 0 To EncryptionNum - 1
                            If EncryptionArray(Y, X, g, f, i, e) <> "" Then
                                If t = Len(Name) Then 'Selects next character in password
                                    t = 1
                                Else:
                                    t = t + 1
                                End If
                                Password = Password & Chr((Val(EncryptionArray(Y, X, g, f, i, e)) - Asc(Mid(Name, t, 1))) / ENumber) 'Decrypts data via
the NMR and adds to plaintext
                            End If
                        Next e
                    Next i
                Next f
            Next g
        Next X
    Next Y
End Sub
```

Appendix E: EncryptionBuilder Code

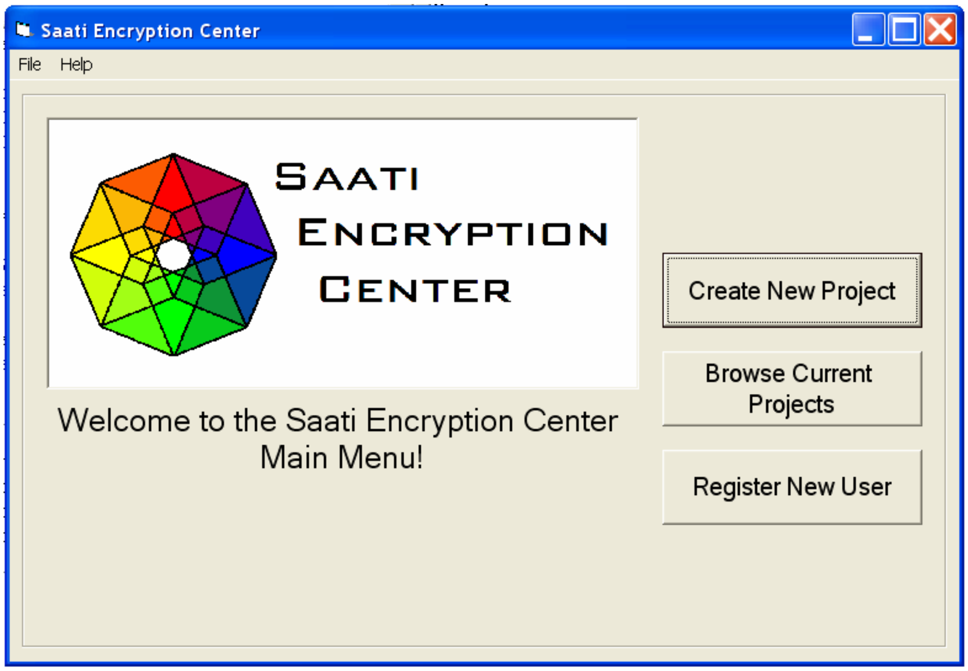
```
'Public variable dimensioning

Public Encryptee As String 'Used to store the plaintext and ciphertext during encryption and decryption
Public Pass As String 'Password used in encryption and decryption
Public EncrypteeLen As Long 'Length of Encryptee, used in encryption and decryption
Public ENumber As Long 'Random number involved in the NMR

Public Sub Encrypt()
   Dim i As Integer, e As Integer 'Counter variables
   Dim Temp As String 'Used to temporarily hold Encryptee's value
   Dim L As String * 4
   If Encryptee = "" Then Exit Sub
   Temp = Encryptee
   Encryptee = ""
   For e = 1 To EncrypteeLen 'Forms ciphertext (Encryptee) via the CSM and the NMR
      If i = Len(Pass) Then
         i = 1
      Else:
         i = i + 1
      End If
      L = LengthCheck((Asc(Mid(Temp, e, 1)) * ENumber) + Asc(Mid(Pass, i, 1)))
      Encryptee = Encryptee & StrReverse(L)
   Next e
End Sub

Public Sub Decrypt()
   Dim i As Integer, e As Integer 'Counter variables
   Dim Temp As String 'Used to temporarily hold Encryptee's value
   If Encryptee = "" Then Exit Sub
   Temp = Encryptee
   Encryptee = ""
   For e = 1 To EncrypteeLen 'Forms plaintext (Encryptee) via the CSM and the NMR
      If i = Len(Pass) Then
         i = 1
      Else:
         i = i + 1
      End If
      Encryptee = Encryptee & Chr((Val(StrReverse(Left(Temp, 4))) - Asc(Mid(Pass, i, 1))) / ENumber) 'Adds decrypted character of
ciphertext to plaintext
      Temp = Right(Temp, Len(Temp) - 4) 'removes used info. from ciphertext
   Next e
End Sub

Public Function LengthCheck(ByVal L As String) As String
   L = Trim(L) 'Removes leading and trailing spaces from L
   Do While Len(L) < 4 'Adds zeros to the beginning of L until L's total length is 4
      L = "0" & L
   Loop
   LengthCheck = L
End Function
```

Appendix F: ProjectLoader Code

```
'Public variable dimensioning

Public Insat As String 'Stores program's installation directory

Public Sub LoadUsers()
    Dim Temp As String
    Dim User1 As CUser 'Creates new User
    ProjectBrowser.NameList.Clear 'Updates user interface
    Open (Insat & "Users\Registered Users") For Input As #1
        Do While Not EOF(1)
            Set User1 = New CUser
            Input #1, Temp
            User1.Name = Temp
            Open (Insat & "Users\" & User1.Name & "\" & User1.Name) For Input As #2 'Opens user's encrypted file
                Input #2, Temp
                User1.Encryption = Temp
                User1.EntryAttempts = 0 'Assigns various automatic properties
                User1.LockedDown = False
                Call User1.Decrypt
                ProjectBrowser.Users.Add User1 'Adds user to collection of users
            Close #2
            ProjectBrowser.NameList.AddItem User1.Name 'Updates user interface
        Loop
    Close #1
End Sub

Public Sub Load_Project(SelectedProject As CProject)
    Dim Temp As String
    ProjectBrowser.EncryptionList.Clear
    Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryption Names") For Input As #1 'Opens file
containing a list of the Project's Encryptions
        Do While Not EOF(1)
            Input #1, Temp
            ProjectBrowser.EncryptionList.AddItem (Temp) 'Updates user interface
        Loop
    Close #1
    ProjectBrowser.Label4.Caption = "Project's current status: " & SelectedProject.Status 'Updates user interface
End Sub

Public Sub LoadProjects()
    ProjectBrowser.ProjectList.Clear 'Updates user interface
    Open (Insat & "Projects\Projects") For Input As #1
        Do While Not EOF(1)
            Call PC
        Loop
    Close #1
End Sub

Private Sub PC() 'Stands fo Project Creation
    Dim NewProject As New CProject
    Dim Temp As String
    Input #1, Temp
    NewProject.Name = Temp
    ProjectBrowser.ProjectList.AddItem Temp 'Updates user interface
    Input #1, Temp
    NewProject.Owner = Temp
    Open (Insat & "Projects\" & NewProject.Owner & "\" & NewProject.Name & "\" & NewProject.Name) For Input As #2 'Opens Project's
encrypted file
        Input #2, Temp
    Close #2
    NewProject.Encryption = Temp
    Call NewProject.Decrypt
    ProjectBrowser.Projects.Add NewProject 'Adds project to collection of projects
End Sub
```

```
Public Sub Load_Encryptions(SelectedProject As CProject)
    Dim i As Integer
    Do While ProjectBrowser.Encryptions.Count > 0
        ProjectBrowser.Encryptions.Remove (1)
    Loop
    If ProjectBrowser.EncryptionList.ListCount > 0 Then
        For i = 1 To ProjectBrowser.EncryptionList.ListCount '1 to number of Project's Encryptions
            Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryptions\" &
ProjectBrowser.EncryptionList.List(i - 1)) For Input As #1 'Opens Encryption's encrypted file
                Call EC(SelectedProject)
            Close #1
        Next i
    End If
End Sub

Private Sub EC(SelectedProject As CProject)
    Dim NewEncryption As New CEncryption
    Dim Temp As String
    Input #1, Temp
    NewEncryption.Encryption = Temp
    NewEncryption.ProjectOwner = SelectedProject.Owner 'Assigns automatic properties
    NewEncryption.ProjectName = SelectedProject.Name
    Call NewEncryption.DecryptData
    ProjectBrowser.Encryptions.Add NewEncryption 'Adds Encryption to collection of Encryptions
End Sub
```

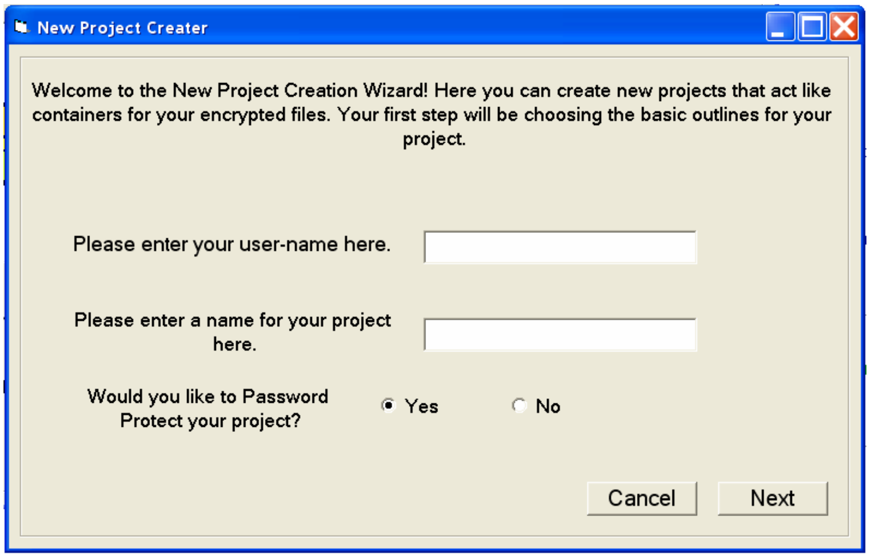
Appendix G: MainMenu Pictures and Code



Welcome to the Saati Encryption Center Main Menu!

```
'Startup screen, Purpose: Menu

Private Sub About_Click() 'Menu bar button
   AboutForm.Show
End Sub

Private Sub BrowseCurrentProjects_Click()
   ProjectBrowser.SelectedUser.Name = "" 'Prepares ProjectBrowser for use
   ProjectBrowser.SelectedUser.Password = ""
   ProjectBrowser.Show
   Call ProjectLoader.LoadUsers
   Call ProjectLoader.LoadProjects
   ProjectBrowser.NameList.SetFocus
   ProjectBrowser.RemoveUser.Enabled = False
End Sub

Private Sub BrowseCurrentProjects_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If InStr(Explanation.Caption, "new") > 0 Or InStr(Explanation.Caption, "Welcome") > 0 Then Explanation.Caption = "Click on this option
to browse through previously created Projects." 'Displays text explaining about BrowseCurrentProjects
End Sub

Private Sub CreateNewProject_Click()
   NewProjectCreation.Show
   NewProjectCreation.Frame1.Visible = True
   NewProjectCreation.NUName.SetFocus
End Sub

Private Sub CreateNewProject_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If InStr(Explanation.Caption, "browse") > 0 Or InStr(Explanation.Caption, "Welcome") > 0 Then
      Explanation.Caption = "Click on this option to create a new Project that can contain related encrypted items. Password Protected, a
project is a completely safe workstation from which you can manage all of your important encrypted files." 'Displays text explaining
about CreateNewProject
      Explanation.Font.Size = 14
   End If
End Sub

Private Sub Exit_Click() 'Menu bar button
   If MsgBox("Are you sure that you want to exit?", vbYesNo, "[>_<]") = vbNo Then End
End Sub

Private Sub Form_Load()
   Open "C:\Program Files\Saati Encryption Center\Insat" For Input As #1
      Input #1, Insat 'Retrieves installation directory
   Close #1
   Open (Insat & "Users\Registered Users") For Input As #1 'Checks  to see if there are any registered users
      If EOF(1) Then 'If there are none, makes it so new user can only register
         BrowseCurrentProjects.Enabled = False
         CreateNewProject.Enabled = False
      End If
   Close #1
   Call ProjectLoader.LoadUsers
   Open (Insat & "Projects\Projects") For Input As #1 'If there are no registered Projects, makes it so user can only create new projects
      If EOF(1) = True Then BrowseCurrentProjects.Enabled = False
   Close #1
   HelpForm.Timer1.Enabled = True
   Exit Sub
ERROR:
   MsgBox "Saati Encryption Center must be installed before you can use it!", , "[>_<] ERROR" 'In case the program has not been
installed or the directory file on the C drive has been damaged or deleted
   End
End Sub

Private Sub GreetingScreen_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If InStr(Explanation.Caption, "browse") > 0 Or InStr(Explanation.Caption, "new") > 0 Then 'Displays Greeting
      Explanation.Caption = "Welcome to the Saati Encryption Center Main Menu!"
      Explanation.Font.Bold = False
      Explanation.Font.Size = 16
   End If
End Sub

Private Sub Help_Click() 'Menu bar button
   HelpForm.Show
End Sub

Private Sub MSplashScreen_Click() 'Menu bar button
   SplashScreen.Show
End Sub

Private Sub RNU_Click() 'Stands for Register New User
   NewUserRegistry.Show
   NewUserRegistry.Text1.SetFocus
End Sub

Private Sub RNU_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Explanation.Caption = "Click on this button to register yourself as a new user who can use Saati Encryption Center." 'Displays text
explaining about RNU
End Sub
```

Appendix H: NewProjectCreation Pictures and Code



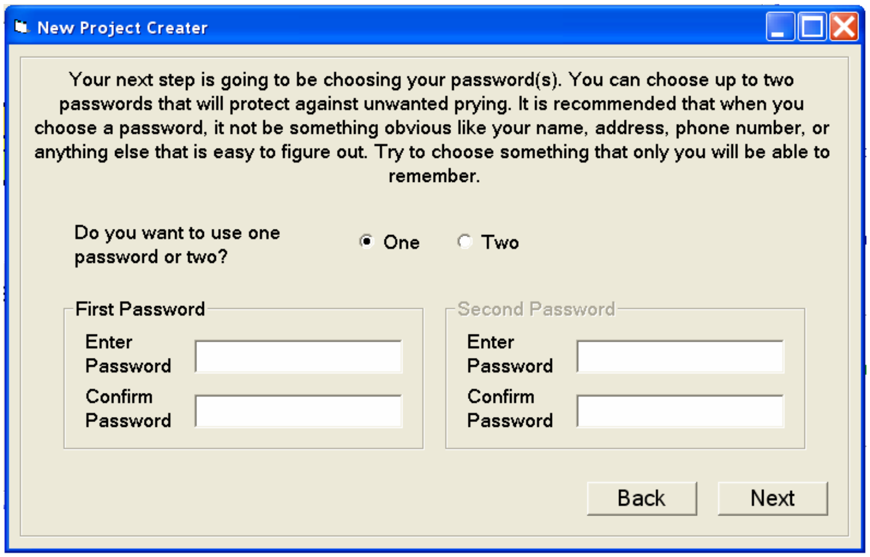
Welcome to the New Project Creation Wizard! Here you can create new projects that act like containers for your encrypted files. Your first step will be choosing the basic outlines for your project.

Please enter your user-name here.

Please enter a name for your project here.

Would you like to Password Protect your project? ● Yes ○ No

Figure 1:



Your next step is going to be choosing your password(s). You can choose up to two passwords that will protect against unwanted prying. It is recommended that when you choose a password, it not be something obvious like your name, address, phone number, or anything else that is easy to figure out. Try to choose something that only you will be able to remember.

Do you want to use one password or two? ● One ○ Two

First Password

Enter Password

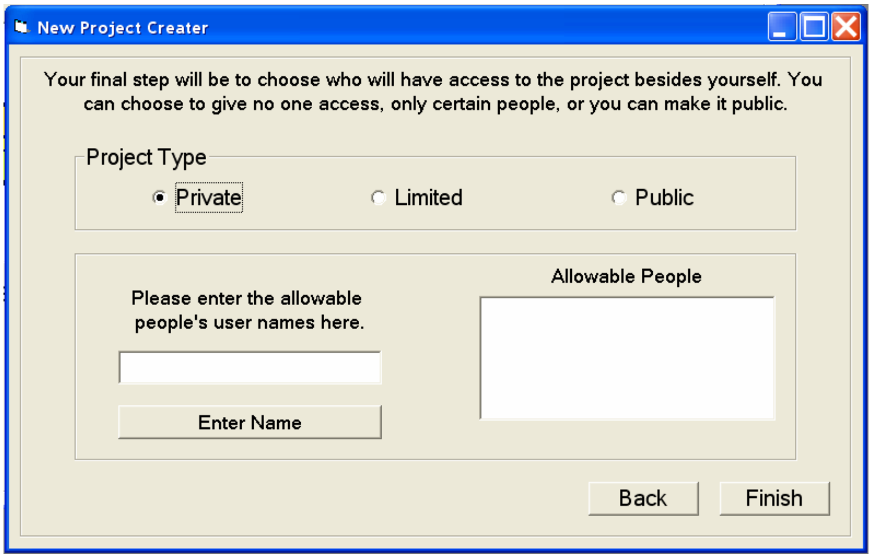
Confirm Password

Second Password

Enter Password

Confirm Password

Figure 2:



Your final step will be to choose who will have access to the project besides yourself. You can choose to give no one access, only certain people, or you can make it public.

Project Type

● Private ○ Limited ○ Public

Please enter the allowable people's user names here.

Figure 3:

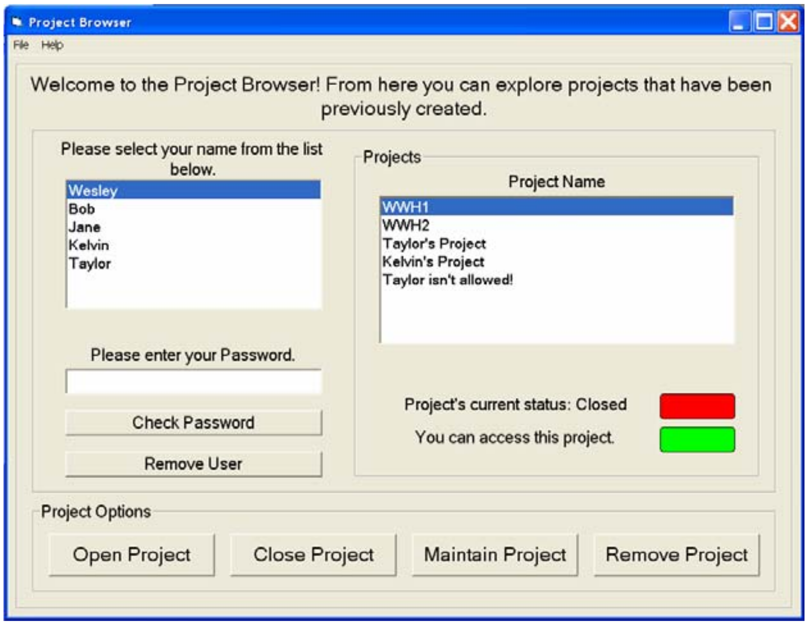
```
'Public Object dimensioning

Public NewProject As New CProject

Private Sub ANE_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then Call Enter_Name_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub Back1_Click()
   Frame2.Visible = False
   Frame1.Visible = True
End Sub

Private Sub Back2_Click()
   Frame5.Visible = False
   Frame2.Visible = True
End Sub

Private Sub Cancel_Click() 'Closes form
   NUName.Text = ""
   NPName.Text = ""
   NewProjectCreation.Hide
End Sub

Private Sub CFPW_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then 'Enables "Tab" function when enter is pressed
      If Frame4.Enabled = True Then 'Decides wat to "Tab" to depending on if the user is using the optional second password
         SPW.SetFocus
      Else:
         Call Next2_Click
      End If
   End If
End Sub

Private Sub Create_Project_Click()
   Dim i As Integer
   If NewProject.EType = "Limited" And List1.ListCount = 0 Then
      MsgBox "You have chosen to make a Limited Project. There must be at least one allowable person.", , "[>_<] Missing People!"
      ANE.SetFocus
      Exit Sub
   End If
   NewProject.CreationDate = Format(Now, "mm/dd/yy") & " " & Format(Now, "hh:mm:ss AM/PM") 'Assigns properties
   NewProject.Name = NPName.Text
```

```
    NewProject.Owner = NUName.Text
    NewProject.Password1 = FPW.Text
    If Option4.Value = True Then NewProject.Password2 = SPW.Text 'Assigns second password if user has a second password
    NewProject.Status = "Closed"
    Open (Insat & "Projects\Projects") For Append As #1 'Registers Project in file
        Write #1, NewProject.Name
        Write #1, NewProject.Owner
    Close #1
    Call NewProject.Encrypt
    Open (Insat & "Projects\" & NewProject.Owner & "\" & NewProject.Name & "\Users") For Output As #1 'Records Project's mode
        Write #1, NewProject.EType
        If NewProject.EType = "Limited" Then 'if mode is limited, records all allowable users' names
            For i = 0 To List1.ListCount - 1
                Write #1, List1.List(i)
            Next i
        End If
    Close #1
    Call Reset 'Resets and hides form
    Frame5.Visible = False
    Frame1.Visible = True
    MainMenu.BrowseCurrentProjects.Enabled = True
    NewProjectCreation.Hide
End Sub

Private Sub Reset() 'Resets form's controls for next use
    NUName.Text = ""
    NPName.Text = ""
    Option1.Value = True
    FPW.Text = ""
    CFPW.Text = ""
    SPW.Text = ""
    CSPW.Text = ""
    Option3.Value = True
    Option5.Value = True
    ANE.Text = ""
    List1.Clear
End Sub

Private Sub CSPW_Change()
    If KeyAscii = 13 Then Call Next2_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub Enter_Name_Click()
    List1.AddItem (ANE.Text) 'Adds "Allowable" User's name to list
    ANE.Text = ""
    ANE.SetFocus
End Sub

Private Sub FPW_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then CFPW.SetFocus 'Enables "Tab" function when enter is pressed
End Sub

Private Sub Next1_Click()
    Dim i As Integer
    Dim Temp As String
    Dim Good As Boolean
    NUName.Text = Trim(NUName.Text)
    NPName.Text = Trim(NPName.Text)
    If NUName.Text = "" Then 'Checks whether or not the user has entered their user-name
        MsgBox "Please enter a valid User Name", , "[>_<] Missing Information"
        NUName.Text = ""
        NUName.SetFocus
        Exit Sub
    ElseIf NPName.Text = "" Or NPName.Text = "Encryption Names" Or NPName.Text = "Users" Then 'Checks whether or not the user has
entered a valid name for the Project
        MsgBox "Please enter a valid Project Name", , "[>_<] Missing Information"
        NPName.Text = ""
        NPName.SetFocus
        Exit Sub
    End If
    Open (Insat & "Users\Registered Users") For Input As #1 'Checks to make sure the User is registered
```

```
        Do While Not EOF(1)
            Input #1, Temp
            If Temp = NUName.Text Then
                Good = True
                Exit Do
            End If
        Loop
    Close #1
    If Good = False Then
        For i = 1 To ProjectBrowser.Users.Count
            If ProjectBrowser.Users.Item(i).Name = NUName.Text Then
                If ProjectBrowser.Users.Item(i).LockedDown = True Then 'If User has been locked out of the program, then it resets and hides
the form
                    MsgBox "Sorry " & NUName.Text & ", but you have been locked out and are not able to make any new projects as of now.", ,
"[>_<] Access Error"
                    Call Reset
                    NewProjectCreation.Hide
                    Exit Sub
                End If
                Exit For
            End If
        Next i
        MsgBox "That User-Name is not recognized. Please enter a valid name, or return to the menu and register as a new user.", , "[>_<]
Unfamiliar Name"
        NUName.Text = ""
        NUName.SetFocus
        Exit Sub
    End If
    Good = True
    Open (Insat & "Projects\Projects") For Input As #1
        Do While Not EOF(1) 'Checks to see if the Project's name has been taken
            Input #1, Temp 'Reads Project's name from file.
            If Temp = NPName.Text Then
                Good = False
                Exit Do
            End If
            Input #1, Temp 'Reads Project's owner's name from file
        Loop
    Close #1
    If Good = False Then
        MsgBox "That Project-Name has been taken. Please enter another name.", , "[>_<] Name already taken"
        NPName.Text = ""
        NPName.SetFocus
        Exit Sub
    End If
    Frame1.Visible = False
    If Option1.Value = True Then
        Frame2.Visible = True
        FPW.SetFocus
    Else:
        Frame5.Visible = True
    End If
End Sub

Private Sub Next2_Click()
    FPW.Text = Trim(FPW.Text)
    CFPW.Text = Trim(CFPW.Text)
    SPW.Text = Trim(SPW.Text)
    CSPW.Text = Trim(CSPW.Text)
    If FPW.Text <> CFPW.Text Then 'Makes sure that the User confirmed their first password correctly
        MsgBox "Please re-enter your first password.", , "[>_<] Missing/Invalid Password Information"
        FPW.SetFocus
        Exit Sub
    End If
    If Option4.Value = True Then
        If SPW.Text = "" Or CSPW.Text = "" Or SPW.Text <> CSPW.Text Then 'Makes sure that the User confirmed their second password
correctly
            MsgBox "Please re-enter your Second password.", , "[>_<] Missing/Invalid Password Information"
            SPW.SetFocus
            Exit Sub
        End If
```

```
   End If
   NewProject.EType = "Private"
   Frame2.Visible = False
   Frame5.Visible = True
   Frame7.Enabled = False
End Sub

Private Sub NPName_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then Call Next1_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub NUName_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then NPName.SetFocus 'Enables "Tab" function when enter is pressed
End Sub

Private Sub Option3_Click() 'Allows user to enter only first password
   Frame4.Enabled = False
End Sub

Private Sub Option4_Click() 'Allows user to enter optional second password
   Frame4.Enabled = True
End Sub

Private Sub Option5_Click() 'Sets NewProject's mode to "Private"
   Frame7.Enabled = False
   NewProject.EType = "Private"
End Sub

Private Sub Option6_Click() 'Sets NewProject's mode to "Limited"
   Frame7.Enabled = True
   NewProject.EType = "Limited"
End Sub

Private Sub Option7_Click() 'Sets NewProject's mode to "Public"
   Frame7.Enabled = False
   NewProject.EType = "Public"
End Sub

Private Sub SPW_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then CSPW.SetFocus 'Enables "Tab" function when enter is pressed
End Sub
```

Appendix I: ProjectBrowser Pictures and Code



Welcome to the Project Browser! From here you can explore projects that have been previously created.

Please select your name from the list below.

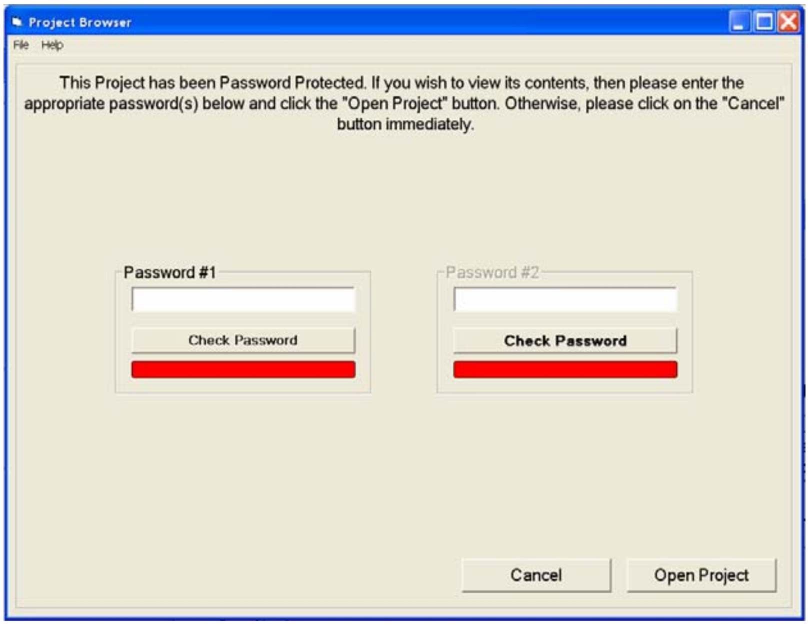
Wesley
Bob
Jane
Kelvin
Taylor

WWH1
WWH2
Taylor's Project
Kelvin's Project
Taylor isn't allowed!

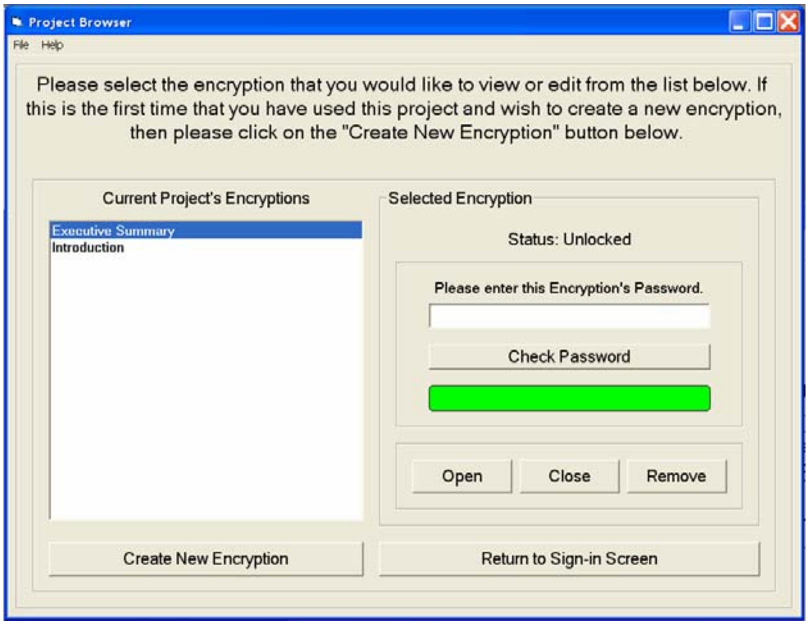
Please enter your Password.

Project's current status: Closed 🟥

You can access this project. 🟩



This Project has been Password Protected. If you wish to view its contents, then please enter the appropriate password(s) below and click the "Open Project" button. Otherwise, please click on the "Cancel" button immediately.



Please select the encryption that you would like to view or edit from the list below. If this is the first time that you have used this project and wish to create a new encryption, then please click on the "Create New Encryption" button below.

Status: Unlocked

Please enter this Encryption's Password.

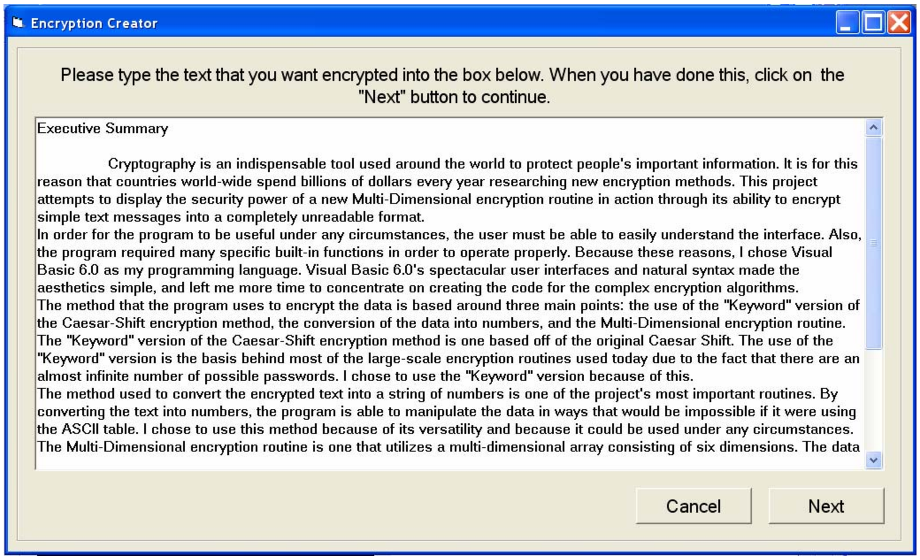
```
'Public Object and Collection dimensioning

Public Users As New Collection
Public Projects As New Collection
Public Encryptions As New Collection
Public SelectedUser As New CUser
Public SelectedProject As New CProject
Public SelectedEncryption As New CEncryption

Private Sub Cancel_Click()
  PE1.Text = ""
  PE2.Text = ""
  Shape1.BackColor = vbRed
  Shape2.BackColor = vbRed
  Frame2.Visible = False
  Frame1.Visible = True
End Sub

Private Sub EncryptionList_DblClick()
  Dim i As Integer
  MF8.Enabled = True
  If Encryptions.Count > 0 Then
    For i = 1 To Encryptions.Count 'Searches encryptions for name match
      If Encryptions.Item(i).Name = EncryptionList.List(EncryptionList.ListIndex) Then
        SelectedEncryption.CreationDate = Encryptions.Item(i).CreationDate 'Selects matching Encryption
        SelectedEncryption.EDate = Encryptions.Item(i).EDate
        SelectedEncryption.Encryption = Encryptions.Item(i).Encryption
        SelectedEncryption.ENumber = Encryptions.Item(i).ENumber
        SelectedEncryption.EScript = Encryptions.Item(i).EScript
        SelectedEncryption.Name = Encryptions.Item(i).Name
        SelectedEncryption.Password = Encryptions.Item(i).Password
        SelectedEncryption.ProjectName = Encryptions.Item(i).ProjectName
        SelectedEncryption.ProjectOwner = Encryptions.Item(i).ProjectOwner
        SelectedEncryption.BackupPassword = Encryptions.Item(i).BackupPassword
        SelectedEncryption.Status = Encryptions.Item(i).Status
        Label9.Caption = "Status: " & SelectedEncryption.Status 'Updates user interface
        If SelectedEncryption.Status = "Locked" Then
          Shape3.BackColor = vbRed
          MF9.Enabled = False
```

```
                Else:
                    Shape3.BackColor = vbGreen
                    MF9.Enabled = True
                    Call OpenEncryption_Click
                End If
                If SelectedEncryption.Password = "" Then 'If Encryption has no user-defined password, opens the Encryption
                    MF8.Enabled = False
                    SelectedEncryption.Status = "Unlocked"
                    Encryptions.Item(i).Status = "Unlocked"
                    Label9.Caption = "Status: Unlocked"
                    Shape3.BackColor = vbGreen
                    Call OpenEncryption_Click
                End If
                Exit For
            End If
        Next i
    End If
End Sub

Private Sub RemoveProject_Click()
    Dim i As Integer, e As Integer
    Set fso = CreateObject("Scripting.FileSystemObject")
    i = MsgBox("Are you sure that you want to delete this Project?", vbYesNo, "[-_-]")
    If i = vbNo Then Exit Sub
    i = MsgBox("FINAL WARNING! Are you absolutely sure that you want to delete this Project?", vbYesNo, "[-_-] FINAL WARNING!")
    If i = vbNo Then Exit Sub
    i = 1
    If Projects.Count > 0 Then
        For i = 1 To Projects.Count
            If Projects.Item(i).Name = ProjectList.List(ProjectList.ListIndex) Then 'Searches for Project name match
                Open (Insat & "Projects\projects") For Output As #1
                    For e = 1 To Projects.Count
                        If Projects.Item(e).Name <> ProjectList.List(ProjectList.ListIndex) Then Write #1, ProjectList.List(ProjectList.ListIndex)
                    Next e
                Close #1 'Removes all data concerning the Project
                fso.DeleteFolder (Insat & "Projects\" & Projects.Item(i).Owner & "\" & ProjectList.List(ProjectList.ListIndex))
                Projects.Remove (i)
                ProjectList.Clear
                If Projects.Count > 0 Then
                    e = 1
                    For e = 1 To Projects.Count
                        ProjectList.AddItem (Projects.Item(e).Name)
                    Next e
                End If
                MsgBox "Project removed successfully.", , "[^_^]"
                Exit For
            End If
        Next i
    End If
End Sub

Private Sub RemoveUser_Click()
    Dim i As Integer, e As Integer
    Set fso = CreateObject("Scripting.FileSystemObject")
    i = MsgBox("Are you sure that you want to delete this User Profile?", vbYesNo, "[-_-]")
    If i = vbNo Then Exit Sub
    i = MsgBox("FINAL WARNING! Are you absolutely sure that you want to delete this User Profile?", vbYesNo, "[-_-] FINAL WARNING!")
    If i = vbNo Then Exit Sub
    i = 1
    If Users.Count > 0 Then
        For i = 1 To Users.Count 'Searches for User name match
            If Users.Item(i).Name = NameList.List(NameList.ListIndex) Then
                Open (Insat & "Users\Registered Users") For Output As #1 'Removes User-name from User registry file
                    For e = 1 To Users.Count
                        If Users.Item(e).Name <> NameList.List(NameList.ListIndex) Then Write #1, Users.Item(e).Name
                    Next e
                Close #1
                If Projects.Count > 0 Then
                    e = 0
                    For e = 1 To Projects.Count
                        If e > Projects.Count Then Exit For
                        If Projects.Item(e).Owner = NameList.List(NameList.ListIndex) Then 'Removes all of User's Projects
                            Projects.Remove (e)
                            e = e - 1
                        End If
                    Next e
                    ProjectList.Clear
                    If Projects.Count > 0 Then
                        e = 1
                        Open (Insat & "Projects\Projects") For Output As #1
                            For e = 1 To Projects.Count
                                Write #1, Projects.Item(e).Name
                                Write #1, Projects.Item(e).Owner
                            Next e
                        Close #1
                        e = 1
                        For e = 1 To Projects.Count
                            ProjectList.AddItem (Projects.Item(e).Name)
                        Next e
                    End If
                End If
                e = 1
                For e = 1 To Users.Count
                    If Users.Item(e).Name = NameList.List(NameList.ListIndex) Then 'Removes User from Users collection
                        Users.Remove (e)
                        Exit For
                    End If
                Next e
                fso.DeleteFolder (Insat & "Users\" & NameList.List(NameList.ListIndex)) 'Deletes User's folders
                fso.DeleteFolder (Insat & "Projects\" & NameList.List(NameList.ListIndex))
                NameList.Clear
                If Users.Count > 0 Then
                    e = 1
                    For e = 1 To Users.Count 'Replenishes User listing on form
                        NameList.AddItem (Users.Item(e).Name)
                    Next e
                End If
                MsgBox "User profile removed successfully.", , "[^_^]"
                Exit Sub
            End If
        Next i
    End If
End Sub

Private Sub RTSIS_Click() 'Returns user to Sign-in screen
    Frame3.Visible = False
    Frame1.Visible = True
End Sub

Private Sub CEP_Click()
    Dim i As Integer
    If EPE.Text = SelectedEncryption.Password Then 'Checks to see if password is correct
        Shape3.BackColor = vbGreen
        SelectedEncryption.Status = "Unlocked"
        If Encryptions.Count > 0 Then
            For i = 1 To Encryptions.Count 'Unlocks Encryption
                If Encryptions.Item(i).Name = SelectedEncryption.Name Then
                    Encryptions.Item(i).Status = "Unlocked"
                End If
            Next i
        End If
        MF9.Enabled = True
        Label9.Caption = "Status: Unlocked"
        Call RecordEncryptionChange
    Else:
        MsgBox "Sorry, that was the wrong Password. Please remember that Passwords are Case Sensitive.", , "[>_<] Access Error"
        If SelectedEncryption.Status = "Locked" Then Shape3.BackColor = vbRed
    End If
    EPE.Text = ""
End Sub

Private Sub CheckPassword_Click()
```

```
    Dim i As Integer
    If Users.Count > 0 Then
        For i = 1 To Users.Count 'Searches for correct User
            If Users.Item(i).Name = NameList.List(NameList.ListIndex) Then
                Users.Item(i).EntryAttempts = Users.Item(i).EntryAttempts + 1
                If Users.Item(i).LockedDown = True Or Users.Item(i).EntryAttempts > 3 Then
                    MsgBox "Sorry " & Users.Item(i).Name & ", but you have been locked out and are now unable to log in.", , "[>_<] Access Error"
                    Exit Sub
                End If
                If PasswordEntry.Text = Users.Item(i).Password Then 'Checks to see if the password is correct
                    MF3.Enabled = True
                    RemoveUser.Enabled = True
                    Users.Item(i).EntryAttempts = 0
                    SelectedUser.Name = Users.Item(i).Name
                    SelectedUser.Password = Users.Item(i).Password
                    SelectedUser.EntryAttempts = 0
                Else:
                    If Users.Item(i).EntryAttempts >= 3 Then
                        Users.Item(i).LockedDown = True
                        MsgBox "PLEASE NOTE! You have entered three wrong passwords and " & Users.Item(i).Name & " will now be locked out!",
, "[>_<] Access Error"
                        PasswordEntry.Text = ""
                    Else:
                        MsgBox "Sorry, that's not the right password.", , "[>_<] Invalid Password"
                    End If
                End If
                Exit For
            End If
        Next i
    End If
    PasswordEntry.Text = ""
End Sub

Private Sub CloseEncryption_Click()
    Dim i As Integer
    If Encryptions.Count > 0 Then
        For i = 1 To Encryptions.Count 'Finds correct Encryption and Locks it
            If Encryptions.Item(i).Name = SelectedEncryption.Name Then Encryptions.Item(i).Status = "Locked"
        Next i
    End If
    If SelectedEncryption.Password = "" Then 'Updates User interface
        MF8.Enabled = False
        Label9.Caption = "Status: Unlocked"
        Shape3.BackColor = vbGreen
    Else:
        SelectedEncryption.Status = "Locked"
        Label9.Caption = "Status: Locked"
        Shape3.BackColor = vbRed
    End If
    MF9.Enabled = False
    Call RecordEncryptionChange
End Sub

Private Sub CloseProject_Click()
    Dim i As Integer
    SelectedProject.Status = "Closed"
    If Projects.Count > 0 Then
        For i = 1 To Projects.Count 'Searches for correct project
            If Projects.Item(i).Name = SelectedProject.Name Then
                Projects.Item(i).Status = "Closed" 'closes project
                Exit For
            End If
        Next i
    End If
    Call RecordProjectChange
    Call ProjectLoader.Load_Project(SelectedProject)
End Sub

Private Sub CP1_Click() 'Checks password against Project's first password
    If PE1.Text = SelectedProject.Password1 Then
        Shape1.BackColor = vbGreen
        If MF5.Enabled = True Then
            PE2.SetFocus
        Else:
            Shape2.BackColor = vbGreen
            Call OpenProject_Click
        End If
    Else:
        MsgBox "The Password entered is invalid. Please remember that passwords are Case Sensitive.", , "[>_<] Wrong Password"
        PE1.Text = ""
        PE1.SetFocus
    End If
End Sub

Private Sub CP2_Click() 'Checks password against Project's second password
    If PE2.Text = SelectedProject.Password2 Then
        Shape2.BackColor = vbGreen
    Else:
        MsgBox "The Password entered is invalid. Please remember that passwords are Case Sensitive.", , "[>_<] Wrong Password"
        PE2.Text = ""
        PE2.SetFocus
    End If
End Sub

Private Sub DeleteEncryption_Click()
    Dim i As Integer
    i = MsgBox("Are you sure that you want to delete this Encryption?", vbYesNo, "[-_-]")
    If i = vbNo Then Exit Sub
    i = MsgBox("FINAL WARNING! Are you absolutely sure that you want to delete this Encryption?", vbYesNo, "[-_-] FINAL WARNING!")
    If i = vbNo Then Exit Sub
    i = 0
    Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryption Names") For Output As #1 'Removes
Encryption
        For i = 0 To EncryptionList.ListCount - 1
            If EncryptionList.List(i) <> SelectedEncryption.Name Then Write #1, EncryptionList.List(i)
        Next i
    Close #1
    Kill (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryptions\" & SelectedEncryption.Name)
'Deletes Encryptions encrypted file
    Call ProjectLoader.Load_Project(SelectedProject)
    SelectedEncryption.BackupPassword = "" 'Clears out all of Selected Encryption's properties
    SelectedEncryption.CreationDate = ""
    SelectedEncryption.EDate = ""
    SelectedEncryption.EScript = ""
    SelectedEncryption.Encryption = ""
    SelectedEncryption.ENumber = 0
    SelectedEncryption.Name = ""
    SelectedEncryption.Password = ""
    SelectedEncryption.ProjectName = ""
    SelectedEncryption.ProjectOwner = ""
    SelectedEncryption.Status = ""
    MsgBox "Encryption deleted successfully!", , "[^_^]"
    Shape3.BackColor = vbRed
End Sub

Private Sub EncryptionList_Click()
    Dim i As Integer
    MF8.Enabled = True
    If Encryptions.Count > 0 Then
        For i = 1 To Encryptions.Count 'Searches for correct Encryption
            If Encryptions.Item(i).Name = EncryptionList.List(EncryptionList.ListIndex) Then
                SelectedEncryption.CreationDate = Encryptions.Item(i).CreationDate 'Loads correct Encryption's properties
                SelectedEncryption.EDate = Encryptions.Item(i).EDate
                SelectedEncryption.Encryption = Encryptions.Item(i).Encryption
                SelectedEncryption.ENumber = Encryptions.Item(i).ENumber
                SelectedEncryption.EScript = Encryptions.Item(i).EScript
                SelectedEncryption.Name = Encryptions.Item(i).Name
                SelectedEncryption.ProjectName = Encryptions.Item(i).ProjectName
                SelectedEncryption.ProjectOwner = Encryptions.Item(i).ProjectOwner
                SelectedEncryption.Password = Encryptions.Item(i).Password
                SelectedEncryption.BackupPassword = Encryptions.Item(i).BackupPassword
                SelectedEncryption.Status = Encryptions.Item(i).Status
                Label9.Caption = "Status: " & SelectedEncryption.Status
                If SelectedEncryption.Status = "Locked" Then 'Updates User interface
                    Shape3.BackColor = vbRed
                    MF9.Enabled = False
                Else:
                    Shape3.BackColor = vbGreen
                    MF9.Enabled = True
                End If
                If SelectedEncryption.Password = "" Then
                    MF8.Enabled = False
                    MF9.Enabled = True
                    SelectedEncryption.Status = "Unlocked"
                    Encryptions.Item(i).Status = "Unlocked"
                    Label9.Caption = "Status: Unlocked"
                    Shape3.BackColor = vbGreen
                End If
                Exit For
            End If
        Next i
    End If
End Sub

Private Sub EPE_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Call CEP_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub Form_Load()
    Call ProjectLoader.LoadUsers
    Call ProjectLoader.LoadProjects
End Sub

Private Sub Maintain_Click()
    If SelectedProject.Status = "Open" Then
        MsgBox "The project must be closed in order for you to maintain it!", , "[>_<] Invalid Project Status"
        Exit Sub
    End If
    ProjectMaintenance.Show 'Loads Project into ProjectMaintenance
    ProjectMaintenance.PNE.Text = SelectedProject.Name
    ProjectMaintenance.PPWE1.Text = SelectedProject.Password1
    ProjectMaintenance.PPWE2.Text = SelectedProject.Password2
    ProjectMaintenance.SelectedProject.CreationDate = SelectedProject.CreationDate
    ProjectMaintenance.SelectedProject.EDate = SelectedProject.EDate
    ProjectMaintenance.SelectedProject.Encryption = SelectedProject.Encryption
    ProjectMaintenance.SelectedProject.ENumber = SelectedProject.ENumber
    ProjectMaintenance.SelectedProject.EType = SelectedProject.EType
    ProjectMaintenance.SelectedProject.Name = SelectedProject.Name
    ProjectMaintenance.SelectedProject.Owner = SelectedProject.Owner
    ProjectMaintenance.SelectedProject.Password1 = SelectedProject.Password1
    ProjectMaintenance.SelectedProject.Password2 = SelectedProject.Password2
    ProjectMaintenance.SelectedProject.Status = SelectedProject.Status
    Select Case SelectedProject.EType
        Case "Private"
            ProjectMaintenance.Option1.Value = True
        Case "Limited"
            ProjectMaintenance.Option2.Value = True
        Case "Public"
            ProjectMaintenance.Option3.Value = True
    End Select
    Call ProjectMaintenance.LoadEncryptions
End Sub

Private Sub MakeEncryption_Click()
    Encrypter.CurrentEncryption.CreationDate = "" 'Clears all of Encrypter's Encryption properties
    Encrypter.CurrentEncryption.EDate = ""
    Encrypter.CurrentEncryption.Encryption = ""
    Encrypter.CurrentEncryption.ENumber = 0
    Encrypter.CurrentEncryption.EScript = ""
    Encrypter.CurrentEncryption.Name = ""
    Encrypter.CurrentEncryption.Password = ""
    Encrypter.CurrentEncryption.BackupPassword = ""
    Encrypter.CurrentEncryption.Status = ""
    Encrypter.CurrentProject.CreationDate = SelectedProject.CreationDate 'Loads Project data to Encrypter
    Encrypter.CurrentProject.EDate = SelectedProject.EDate
    Encrypter.CurrentProject.EType = SelectedProject.EType
    Encrypter.CurrentProject.Name = SelectedProject.Name
    Encrypter.CurrentProject.Owner = SelectedProject.Owner
    Encrypter.CurrentProject.Password1 = SelectedProject.Password1
    Encrypter.CurrentProject.Password2 = SelectedProject.Password2
    Encrypter.CurrentProject.Status = SelectedProject.Status
    Encrypter.Show
    Encrypter.Text1.SetFocus
End Sub

Private Sub MExit_Click()
    End
End Sub

Private Sub MHelp_Click()
    HelpForm.Show
End Sub

Private Sub MMainScreen_Click()
    Call Reset
End Sub

Private Sub MReturnToMenu_Click()
    Call Reset
    ProjectBrowser.Hide
    MainMenu.Show
End Sub

Private Sub NameList_Click()
    Dim i As Integer
    If SelectedUser.Name <> "" Then i = MsgBox(SelectedUser.Name & ", do you wish to log out?", vbYesNo, "[>_<]")
    If i = vbNo Then Exit Sub
    SelectedUser.Name = "" 'Resets controls to log new User in
    SelectedUser.Password = ""
    MF2.Enabled = False
    MF3.Enabled = False
    RemoveUser.Enabled = False
    Label4.Caption = "Project's current status:"
    Label11.Caption = ""
    Shape4.BackColor = &H8000000F
    Shape5.BackColor = &H8000000F
    PasswordEntry.SetFocus
End Sub

Private Sub OP_Click()
    Dim i As Integer
    If Projects.Count > 0 Then
        For i = 1 To Projects.Count 'Searches for correct project
            If Projects.Item(i).Name = ProjectList.List(ProjectList.ListIndex) Then
                SelectedProject.CreationDate = Projects.Item(i).CreationDate
                SelectedProject.EDate = Projects.Item(i).EDate
                SelectedProject.Encryption = Projects.Item(i).Encryption
                SelectedProject.EType = Projects.Item(i).EType
                SelectedProject.Name = Projects.Item(i).Name
                SelectedProject.Owner = Projects.Item(i).Owner
                SelectedProject.Password1 = Projects.Item(i).Password1
                SelectedProject.Password2 = Projects.Item(i).Password2
                SelectedProject.Status = Projects.Item(i).Status
                Exit For
            End If
        Next i
    End If
    If SelectedProject.Status = "Open" Then 'Updates User interface
        Shape1.BackColor = vbGreen
        Shape2.BackColor = vbGreen
        If SelectedProject.Password1 <> "" Then MF4.Enabled = True
        If SelectedProject.Password2 <> "" Then MF5.Enabled = True
        Call OpenProject_Click
    Else:
        Frame1.Visible = False
        If SelectedProject.Password1 <> "" Or SelectedProject.Password2 <> "" Then
            Frame2.Visible = True
            If SelectedProject.Password1 <> "" Then MF4.Enabled = True
            If SelectedProject.Password2 <> "" Then MF5.Enabled = True
            PE1.SetFocus
        Else:
            Call ProjectLoader.Load_Project(SelectedProject)
            Call ProjectLoader.Load_Encryptions(SelectedProject)
            Call RecordProjectChange
            Frame3.Visible = True
        End If
    End If
End Sub

Private Sub OpenEncryption_Click()
    If SelectedEncryption.Status = "Unlocked" Then
        Decrypter.SelectedEncryption.BackupPassword = SelectedEncryption.BackupPassword 'Loads Encryption into Decrypter
        Decrypter.SelectedEncryption.CreationDate = SelectedEncryption.CreationDate
        Decrypter.SelectedEncryption.EDate = SelectedEncryption.EDate
        Decrypter.SelectedEncryption.Encryption = SelectedEncryption.Encryption
        Decrypter.SelectedEncryption.ENumber = SelectedEncryption.ENumber
        Decrypter.SelectedEncryption.EScript = SelectedEncryption.EScript
        Decrypter.SelectedEncryption.Name = SelectedEncryption.Name
        Decrypter.SelectedEncryption.Password = SelectedEncryption.Password
        Decrypter.SelectedEncryption.ProjectName = SelectedEncryption.ProjectName
        Decrypter.SelectedEncryption.ProjectOwner = SelectedEncryption.ProjectOwner
        Decrypter.SelectedEncryption.Status = SelectedEncryption.Status
        Decrypter.Show
        Call SelectedEncryption.DecryptScript 'Decrypts Encryption
    End If
End Sub

Private Sub OpenProject_Click()
    Dim i As Integer
    If MF4.Enabled = True And MF5.Enabled = True Then 'Chooses what to do by how many passwords the Project uses
        If Shape1.BackColor = vbGreen And Shape2.BackColor = vbGreen Then 'Checks to see if both passwords have been entered
            Frame1.Visible = False
            Frame2.Visible = False
            Frame3.Visible = True
            SelectedProject.Status = "Open"
            If Projects.Count > 0 Then
                For i = 1 To Projects.Count 'Searches for correct Project
                    If Projects.Item(i).Name = SelectedProject.Name Then
                        Projects.Item(i).Status = "Open"
                        Exit For
                    End If
                Next i
            End If
            Label4.Caption = "Project's Current Status: Open"
            Call ProjectLoader.Load_Project(SelectedProject)
            Call ProjectLoader.Load_Encryptions(SelectedProject)
            Call RecordProjectChange
            PE1.Text = "" 'Resets controls
            PE2.Text = ""
            Shape1.BackColor = vbRed
            Shape2.BackColor = vbRed
        Else:
            MsgBox "A password is not correct. Please remember that passwords are Case Sensitive.", , "[>_<] Wrong Password(s)"
            PE1.Text = "" 'Resets control
            PE2.Text = ""
            PE1.SetFocus
        End If
    ElseIf MF4.Enabled = True Then
        If Shape1.BackColor = vbGreen Then 'Checks to see if the password has been entered
            Frame1.Visible = False
            Frame2.Visible = False
            Frame3.Visible = True
            SelectedProject.Status = "Open"
            If Projects.Count > 0 Then
                For i = 1 To Projects.Count 'Searches for correct project
```

```
               If Projects.Item(i).Name = SelectedProject.Name Then
                  Projects.Item(i).Status = "Open"
                  Exit For
               End If
            Next i
         End If
         Label4.Caption = "Project's Current Status: Open"
         Call ProjectLoader.Load_Project(SelectedProject)
         Call ProjectLoader.Load_Encryptions(SelectedProject)
         Call RecordProjectChange
         PE1.Text = "" 'Resets control
         Shape1.BackColor = vbRed
      Else:
         MsgBox "A password is not correct. Please remember that passwords are Case Sensitive.", , "[>_<] Wrong Password(s)"
         PE1.Text = ""
         PE1.SetFocus
      End If
   Else:
      Frame1.Visible = False
      Frame2.Visible = False
      Frame3.Visible = True
      Call ProjectLoader.Load_Project(SelectedProject)
      Call ProjectLoader.Load_Encryptions(SelectedProject)
      Call RecordProjectChange
   End If
End Sub

Private Sub PasswordEntry_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then Call CheckPassword_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub PE1_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then 'Enables "Tab" function when enter is pressed
      If MF5.Enabled = True Then
         PE2.SetFocus
         Exit Sub
      End If
      Call CP1_Click
   End If
End Sub

Private Sub PE2_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then Call CP2_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub ProjectList_Click()
   Dim i As Integer
   If Projects.Count > 0 Then
      For i = 1 To Projects.Count 'Searches for correct Project
         If Projects.Item(i).Name = ProjectList.List(ProjectList.ListIndex) Then
            SelectedProject.CreationDate = Projects.Item(i).CreationDate 'Loads Project
            SelectedProject.EDate = Projects.Item(i).EDate
            SelectedProject.ENumber = Projects.Item(i).ENumber
            SelectedProject.EType = Projects.Item(i).EType
            SelectedProject.Name = Projects.Item(i).Name
            SelectedProject.Owner = Projects.Item(i).Owner
            SelectedProject.Password1 = Projects.Item(i).Password1
            SelectedProject.Password2 = Projects.Item(i).Password2
            SelectedProject.Status = Projects.Item(i).Status
            Label4.Caption = "Project's current status: " & SelectedProject.Status
            If SelectedProject.Status = "Open" Then 'Updates User interface
               Shape5.BackColor = vbGreen
            Else:
               Shape5.BackColor = vbRed
            End If
            Maintain.Enabled = False
            Call UserCheck
            Exit For
         End If
      Next i
   End If
End Sub

Private Sub UserCheck() 'Checks to see if the User can access the Project
   Dim i As Integer
   Dim Good As Boolean
   Dim Temp As String
   If Users.Count > 0 Then
      Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Users") For Input As #1
         Input #1, Temp 'Extracts Projec's mode
         Select Case Temp
            Case "Private"
               If SelectedUser.Name = SelectedProject.Owner Then 'Updates User interface
                  MF2.Enabled = True
                  Maintain.Enabled = True
                  RemoveProject.Enabled = True
                  Label11.Caption = "You can access this project."
                  Shape4.BackColor = vbGreen
               Else:
                  MF2.Enabled = False
                  Label11.Caption = "You can't access this project."
                  Shape4.BackColor = vbRed
               End If
            Case "Limited"
               Do While Not EOF(1) 'Checks to see if current user can access the Project
                  Input #1, Temp
                  If Temp = SelectedUser.Name Then
                     Good = True
                     Exit Do
                  End If
               Loop
               If Good = True Or SelectedUser.Name = SelectedProject.Owner Then
                  MF2.Enabled = True 'User is allowed access to the project
                  Label11.Caption = "You can access this project."
                  Shape4.BackColor = vbGreen
                  If SelectedUser.Name = SelectedProject.Owner Then
                     Maintain.Enabled = True
                     RemoveProject.Enabled = True
                  End If
               Else:
                  MF2.Enabled = False
                  Label11.Caption = "You can't access this project."
                  Shape4.BackColor = vbRed
               End If
            Case "Public"
               MF2.Enabled = True 'User is allowed access to the project
               Label11.Caption = "You can access this project."
               Shape4.BackColor = vbGreen
               If SelectedUser.Name = SelectedProject.Owner Then
                  Maintain.Enabled = True
                  RemoveProject.Enabled = True
               End If
         End Select
      Close #1
   End If
End Sub

Public Sub RecordProjectChange()
   Dim i As Integer
   Call SelectedProject.Encrypt
   If Projects.Count > 0 Then
      For i = 1 To Projects.Count 'Searches for correct Project
         If Projects.Item(i).Name = SelectedProject.Name Then
            Projects.Item(i).EDate = SelectedProject.EDate 'Synchronizes Projects
            Projects.Item(i).ENumber = SelectedProject.ENumber
            Exit For
         End If
      Next i
   End If
   Label4.Caption = "Project's current status: " & SelectedProject.Status 'Updates User interface
   If SelectedProject.Status = "Open" Then
      Shape5.BackColor = vbGreen
   Else:
      Shape5.BackColor = vbRed
   End If
End Sub

Public Sub RecordEncryptionChange()
   Dim i As Integer
   Dim Temp As String * 4
   Randomize
   SelectedEncryption.EDate = Format(Now, "mm/dd/yy") & " " & Format(Now, "hh:mm:ss AM/PM")
   SelectedEncryption.ProjectName = SelectedProject.Name 'Synchronizes Encryptions
   SelectedEncryption.ProjectOwner = SelectedProject.Owner
   If Encryptions.Count > 0 Then
      For i = 1 To Encryptions.Count 'Searches for correct Encryption
         If Encryptions.Item(i).Name = SelectedEncryption.Name Then
            Encryptions.Item(i).EDate = SelectedEncryption.EDate 'Synchronizes Encryptions
            Exit For
         End If
      Next i
   End If
   Call SelectedEncryption.EncryptData
   Call SelectedEncryption.SmallRecorder
End Sub

Private Sub Reset() 'Resets form's controls for next use
   ProjectList.Clear
   NameList.Clear
   PasswordEntry.Text = ""
   MF2.Enabled = False
   MF3.Enabled = False
   Label4.Caption = "Project's current status:"
   EncryptionList.Clear
   EPE.Text = ""
   Shape3.BackColor = vbRed
   Label9.Caption = "Status:"
   PE1.Text = ""
   PE2.Text = ""
   Shape1.BackColor = vbRed
   Shape2.BackColor = vbRed
   Frame3.Visible = False
   Frame2.Visible = False
   Frame1.Visible = True
   Call ProjectLoader.LoadUsers
   Call ProjectLoader.LoadProjects
End Sub
```

Appendix J: Encrypter Pictures and Code



Please type the text that you want encrypted into the box below. When you have done this, click on the "Next" button to continue.

Executive Summary

Cryptography is an indispensable tool used around the world to protect people's important information. It is for this reason that countries world-wide spend billions of dollars every year researching new encryption methods. This project attempts to display the security power of a new Multi-Dimensional encryption routine in action through its ability to encrypt simple text messages into a completely unreadable format.

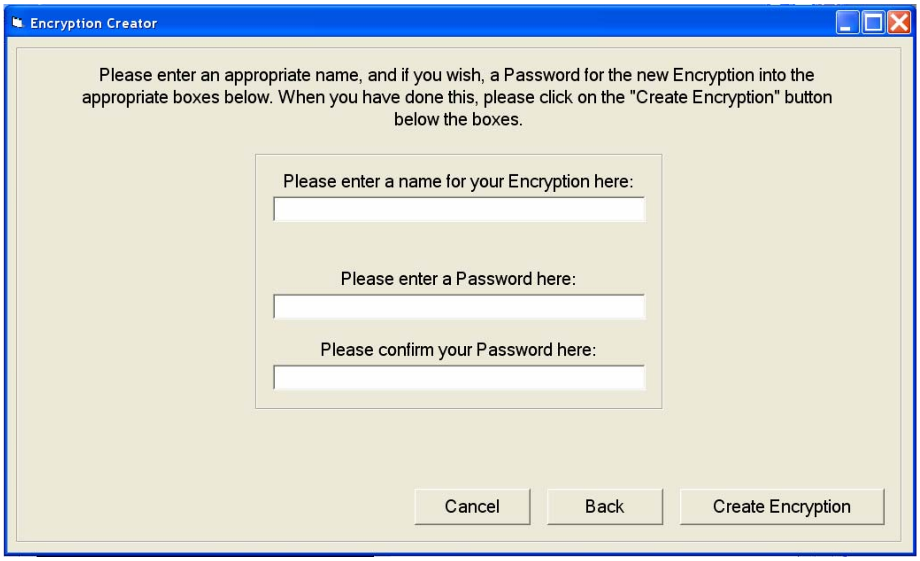
In order for the program to be useful under any circumstances, the user must be able to easily understand the interface. Also, the program required many specific built-in functions in order to operate properly. Because these reasons, I chose Visual Basic 6.0 as my programming language. Visual Basic 6.0's spectacular user interfaces and natural syntax made the aesthetics simple, and left me more time to concentrate on creating the code for the complex encryption algorithms.

The method that the program uses to encrypt the data is based around three main points: the use of the "Keyword" version of the Caesar-Shift encryption method, the conversion of the data into numbers, and the Multi-Dimensional encryption routine.

The "Keyword" version of the Caesar-Shift encryption method is one based off of the original Caesar Shift. The use of the "Keyword" version is the basis behind most of the large-scale encryption routines used today due to the fact that there are an almost infinite number of possible passwords. I chose to use the "Keyword" version because of this.

The method used to convert the encrypted text into a string of numbers is one of the project's most important routines. By converting the text into numbers, the program is able to manipulate the data in ways that would be impossible if it were using the ASCII table. I chose to use this method because of its versatility and because it could be used under any circumstances.

The Multi-Dimensional encryption routine is one that utilizes a multi-dimensional array consisting of six dimensions. The data

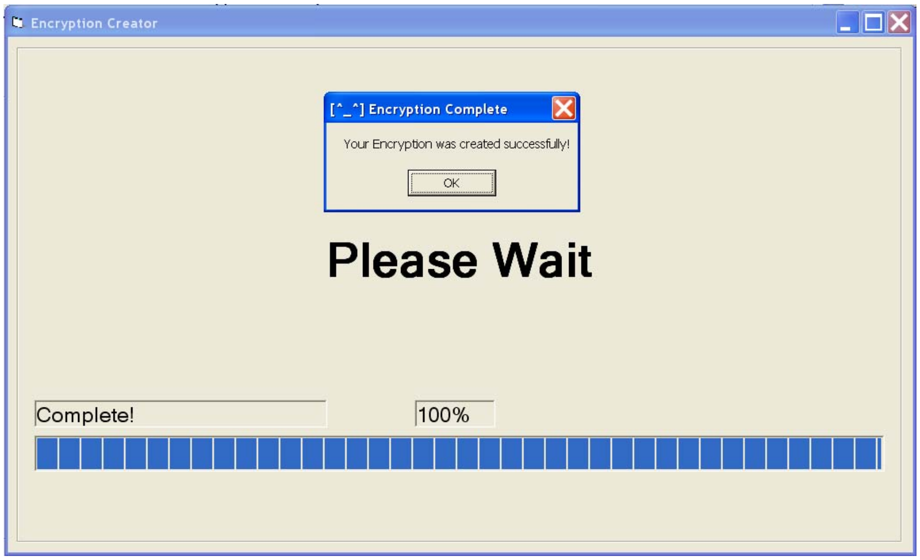


Please enter an appropriate name, and if you wish, a Password for the new Encryption into the appropriate boxes below. When you have done this, please click on the "Create Encryption" button below the boxes.

Please enter a name for your Encryption here:

Please enter a Password here:

Please confirm your Password here:



Your Encryption was created successfully!

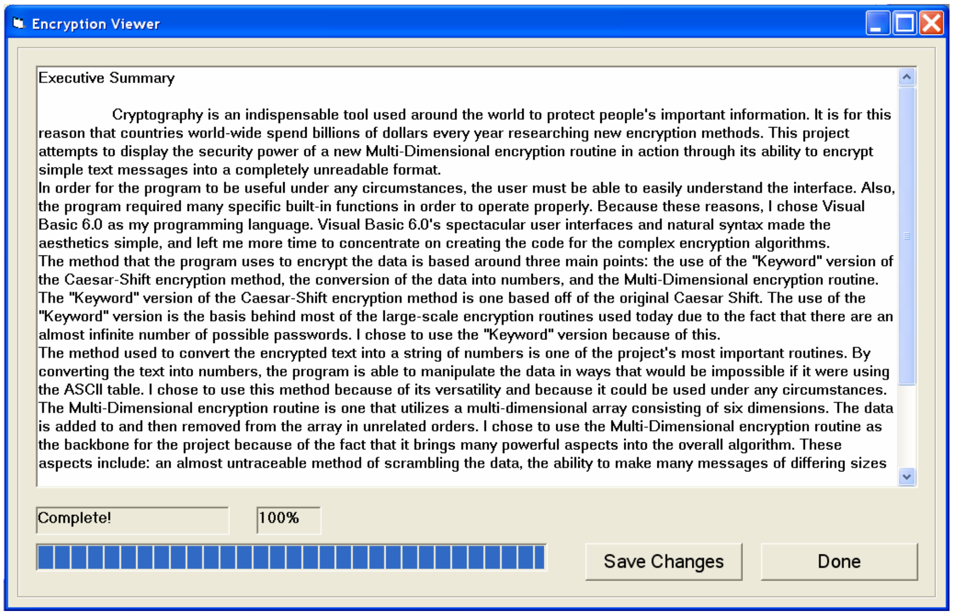
Please Wait

Complete!

100%

```
'Public Collection dimensioning

Public CurrentProject As New CProject
Public CurrentEncryption As New CEncryption

Private Sub Back_Click()
   Frame2.Visible = False
   Frame1.Visible = True
End Sub

Private Sub Cancel_Click(Index As Integer) 'Hides form
   Call Reset
   Encrypter.Hide
End Sub

Private Sub Reset() 'Resets form's controls for next use
   Frame3.Visible = False
   Frame2.Visible = False
   Frame1.Visible = True
   Percent.Caption = ""
   Progress.Caption = ""
   ENE.Text = ""
   PWE.Text = ""
   CPWE.Text = ""
   Text1.Text = ""
   PB1.Value = 0
End Sub

Private Sub CPWE_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then Call Encrypt_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub ENE_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then PWE.SetFocus 'Enables "Tab" function when enter is pressed
End Sub

Private Sub Next1_Click()
   If Text1.Text = "" Then 'Makes sure that the FER has something to encrypt
      MsgBox "Please enter something to encrypt.", , "[>_<]"
      Text1.SetFocus
      Exit Sub
   End If
   Frame1.Visible = False
   Frame2.Visible = True
   ENE.SetFocus
End Sub

Private Sub Encrypt_Click()
   Dim Temp As String
   ENE.Text = Trim(ENE.Text) 'Removes leading and treiling spaces from ENE.Text
   If ENE.Text = "" Then 'Makes sure the user has entered a proper name for the Encryption
      MsgBox "Please enter a name for your Encryption.", , "[>_<] Missing Name"
      ENE.SetFocus
      Exit Sub
   End If
   Open (Insat & "Projects\" & CurrentProject.Owner & "\" & CurrentProject.Name & "\Encryption Names") For Input As #1
      Do While Not EOF(1) 'Makes sure that the new name has not been taken
         Input #1, Temp
         If ENE.Text = Temp Then
            MsgBox "That Name has been taken! Please choose another one!", , "[>_<] Name Taken"
            Close #1
            ENE.Text = ""
            ENE.SetFocus
            Exit Sub
         End If
      Loop
   Close #1
   If PWE.Text <> "" Or CPWE.Text <> "" Then 'Makes sure that the User confirmed the password correctly
      PWE.Text = Trim(PWE.Text)
      CPWE.Text = Trim(CPWE.Text)
      If PWE.Text = "" And CPWE.Text <> "" Then PWE.Text = CPWE.Text
      If PWE.Text <> CPWE.Text Then
         MsgBox "Invalid Password", , "[>_<]"
         PWE.Text = ""
         CPWE.Text = ""
         PWE.SetFocus
         Exit Sub
      End If
   End If
   Frame2.Visible = False
   Frame3.Visible = True
   CurrentEncryption.CreationDate = Format(Now, "mm/dd/yy") & " " & Format(Now, "hh:mm:ss AM/PM") 'Loads New Encryption's
properties
   CurrentEncryption.Name = ENE.Text
   CurrentEncryption.ProjectName = CurrentProject.Name
   CurrentEncryption.ProjectOwner = CurrentProject.Owner
   CurrentEncryption.Password = PWE.Text
   CurrentEncryption.Status = "Locked"
   CurrentEncryption.EScript = Text1.Text
   Call CurrentEncryption.EncryptScript
   MsgBox "Your Encryption was created successfully!", , "[^_^] Encryption Complete"
   Call Reset
   Encrypter.Hide
   Call ProjectLoader.Load_Project(ProjectBrowser.SelectedProject)
   Call ProjectLoader.Load_Encryptions(ProjectBrowser.SelectedProject)
End Sub

Private Sub PWE_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then CPWE.SetFocus 'Enables "Tab" function when enter is pressed
End Sub
```

Appendix K: Decrypter Pictures and Code



Executive Summary

Cryptography is an indispensable tool used around the world to protect people's important information. It is for this reason that countries world-wide spend billions of dollars every year researching new encryption methods. This project attempts to display the security power of a new Multi-Dimensional encryption routine in action through its ability to encrypt simple text messages into a completely unreadable format.

In order for the program to be useful under any circumstances, the user must be able to easily understand the interface. Also, the program required many specific built-in functions in order to operate properly. Because these reasons, I chose Visual Basic 6.0 as my programming language. Visual Basic 6.0's spectacular user interfaces and natural syntax made the aesthetics simple, and left me more time to concentrate on creating the code for the complex encryption algorithms.

The method that the program uses to encrypt the data is based around three main points: the use of the "Keyword" version of the Caesar-Shift encryption method, the conversion of the data into numbers, and the Multi-Dimensional encryption routine.

The "Keyword" version of the Caesar-Shift encryption method is one based off of the original Caesar Shift. The use of the "Keyword" version is the basis behind most of the large-scale encryption routines used today due to the fact that there are an almost infinite number of possible passwords. I chose to use the "Keyword" version because of this.

The method used to convert the encrypted text into a string of numbers is one of the project's most important routines. By converting the text into numbers, the program is able to manipulate the data in ways that would be impossible if it were using the ASCII table. I chose to use this method because of its versatility and because it could be used under any circumstances.

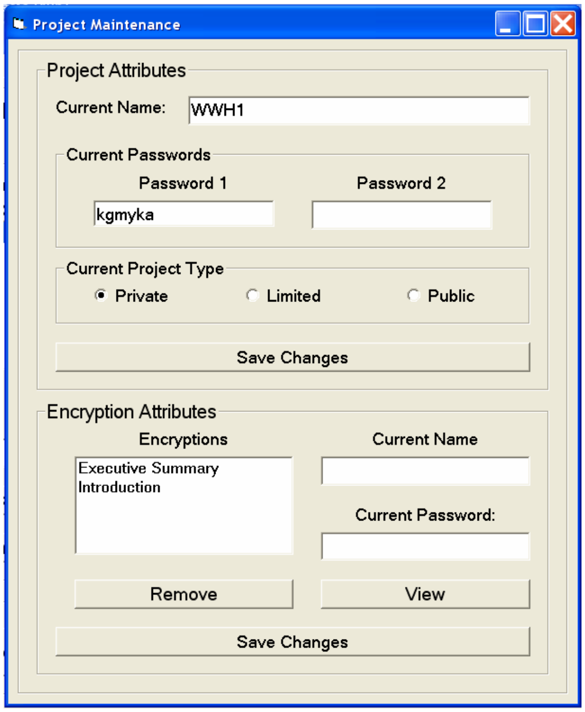
The Multi-Dimensional encryption routine is one that utilizes a multi-dimensional array consisting of six dimensions. The data is added to and then removed from the array in unrelated orders. I chose to use the Multi-Dimensional encryption routine as the backbone for the project because of the fact that it brings many powerful aspects into the overall algorithm. These aspects include: an almost untraceable method of scrambling the data, the ability to make many messages of differing sizes

Complete!

100%

```
Public SelectedEncryption As New CEncryption

Private Sub Done_Click()
   TextDisplay.Text = ""
   Decrypter.Hide
End Sub

Private Sub Save_Click() 'Saves User's changes to the Encryption
   SelectedEncryption.EScript = TextDisplay.Text
   Call SelectedEncryption.EncryptScript
   Encrypter.Hide
   Call ProjectLoader.Load_Project(ProjectBrowser.SelectedProject)
   Call ProjectLoader.Load_Encryptions(ProjectBrowser.SelectedProject)
   MsgBox "Changes Saved Successfully!", , "[^_^] Save Successful"
End Sub

Private Sub Form_Resize() 'Adjusts controls' placement and size properties accordingly
   On Error Resume Next
   Read.Height = Decrypter.ScaleHeight - 120
   Read.Width = Decrypter.ScaleWidth - 240
   TextDisplay.Height = Read.Height - 1800
   TextDisplay.Width = Read.Width - 480
   Progress.Top = TextDisplay.Height + 600
   Percent.Top = TextDisplay.Height + 600
   PB1.Top = TextDisplay.Height + 1080
   PB1.Width = Read.Width - (Done.Width + Save.Width + 1200)
   Done.Top = TextDisplay.Height + 1080
   Done.Left = Read.Width - (Done.Width + 240)
   Save.Top = TextDisplay.Height + 1080
   Save.Left = Done.Left - (Save.Width + 240)
End Sub
```

Appendix L: ProjectMaintenance Pictures and Code



Project Attributes

Current Name: WWH1

Current Passwords

Password 1: kgmyka

Password 2:

Current Project Type

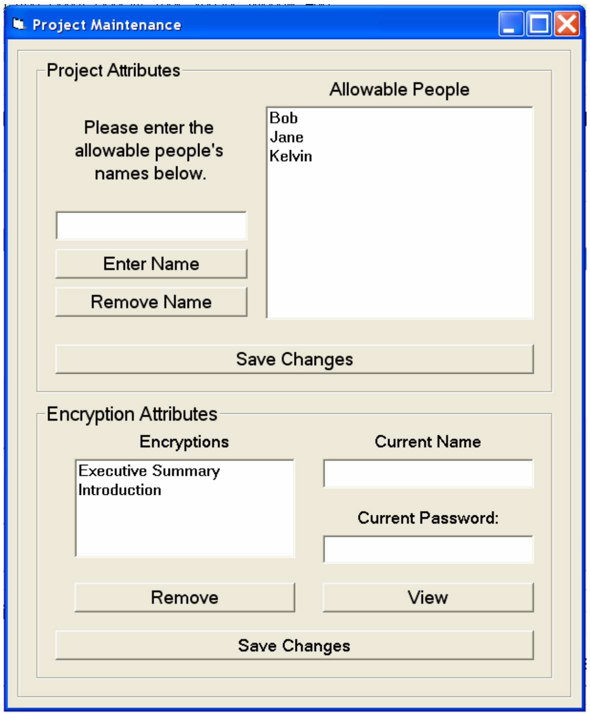
● Private ○ Limited ○ Public

Encryption Attributes

Encryptions:
Executive Summary
Introduction

Current Name:

Current Password:



Project Attributes

Please enter the allowable people's names below.

Allowable People:
Bob
Jane
Kelvin

Encryption Attributes

Encryptions:
Executive Summary
Introduction

Current Name:

Current Password:

```
Public SelectedProject As New CProject
Public SelectedEncryption As New CEncryption
Public Encryptions As New Collection


Public Sub LoadEncryptions()
  Dim i As Integer
  Dim Temp As String
  Do While Encryptions.Count > 0 'Empties Encryptions Collection
    Encryptions.Remove (1)
  Loop
  EncryptionList.Clear
  Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryption Names") For Input As #1
    Do While Not EOF(1) 'Adds all of the Project's Encryptions to a list
      Input #1, Temp
      EncryptionList.AddItem (Temp)
    Loop
  Close #1
  If EncryptionList.ListCount > 0 Then
    For i = 1 To EncryptionList.ListCount 'Loads all of the Project's Encryptions
      Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryptions\" & EncryptionList.List(i - 1))
For Input As #1
        Call EC
      Close #1
    Next i
  End If
End Sub

Private Sub EC()
  Dim NewEncryption As New CEncryption
  Dim Temp As String
  Input #1, Temp 'Extracts ciphertext from file
  NewEncryption.Encryption = Temp 'Sets automatic properties
  NewEncryption.ProjectOwner = SelectedProject.Owner
  NewEncryption.ProjectName = SelectedProject.Name
  Call NewEncryption.DecryptData
  Encryptions.Add NewEncryption
End Sub

Private Sub APNE_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then Call NameEntry_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub EncryptionList_Click()
  Dim i As Integer
  If Encryptions.Count > 0 Then
    For i = 1 To Encryptions.Count 'Searches for correct Encryption
      If Encryptions.Item(i).Name = EncryptionList.List(EncryptionList.ListIndex) Then
        SelectedEncryption.BackupPassword = Encryptions.Item(i).BackupPassword 'Loads Encryption
        SelectedEncryption.CreationDate = Encryptions.Item(i).CreationDate
        SelectedEncryption.EDate = Encryptions.Item(i).EDate
        SelectedEncryption.Encryption = Encryptions.Item(i).Encryption
        SelectedEncryption.ENumber = Encryptions.Item(i).ENumber
        SelectedEncryption.Name = Encryptions.Item(i).Name
        SelectedEncryption.EScript = Encryptions.Item(i).EScript
        SelectedEncryption.ProjectName = Encryptions.Item(i).ProjectName
        SelectedEncryption.ProjectOwner = Encryptions.Item(i).ProjectOwner
        SelectedEncryption.Password = Encryptions.Item(i).Password
        SelectedEncryption.Status = Encryptions.Item(i).Status
        ENE.Text = Encryptions.Item(i).Name
        EPWE.Text = Encryptions.Item(i).Password
        Exit For
      End If
    Next i
  End If
End Sub

Private Sub EncryptionList_DblClick()
  Dim i As Integer
  If Encryptions.Count > 0 Then
    For i = 1 To Encryptions.Count
```

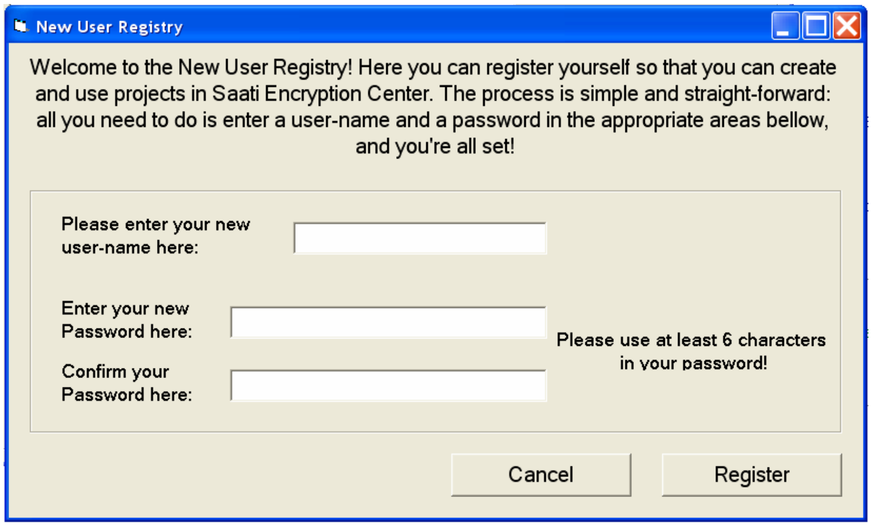
```
            If Encryptions.Item(i).Name = EncryptionList.List(EncryptionList.ListIndex) Then
                SelectedEncryption.BackupPassword = Encryptions.Item(i).BackupPassword 'Loads Encryption
                SelectedEncryption.CreationDate = Encryptions.Item(i).CreationDate
                SelectedEncryption.EDate = Encryptions.Item(i).EDate
                SelectedEncryption.Encryption = Encryptions.Item(i).Encryption
                SelectedEncryption.ENumber = Encryptions.Item(i).ENumber
                SelectedEncryption.Name = Encryptions.Item(i).Name
                SelectedEncryption.EScript = Encryptions.Item(i).EScript
                SelectedEncryption.ProjectName = Encryptions.Item(i).ProjectName
                SelectedEncryption.ProjectOwner = Encryptions.Item(i).ProjectOwner
                SelectedEncryption.Password = Encryptions.Item(i).Password
                SelectedEncryption.Status = Encryptions.Item(i).Status
                ENE.Text = Encryptions.Item(i).Name
                EPWE.Text = Encryptions.Item(i).Password
                Call ViewEncryption_Click 'Opens Encryption in Encryption viewer
                Exit For
            End If
        Next i
    End If
End Sub

Private Sub ENE_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Call SaveEncryption_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub EPWE_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Call SaveEncryption_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub NameEntry_Click()
    If APNE.Text = "" Then Exit Sub 'Adds "Allowable" User's name to a list
    APNL.AddItem (APNE.Text)
    APNE.Text = ""
    APNE.SetFocus
End Sub

Private Sub PNE_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Call SaveProject1_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub PPWE1_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Call SaveProject1_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub PPWE2_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Call SaveProject1_Click 'Enables "Tab" function when enter is pressed
End Sub

Private Sub RemoveEncryption_Click()
    Dim i As Integer
    i = MsgBox("Are you sure that you want to delete this encryption?", vbYesNo, "[-_-]")
    If i = vbNo Then Exit Sub
    i = MsgBox("FINAL WARNING! Are you absolutely sure that you want to delete this Encryption?", vbYesNo, "[-_-] FINAL WARNING!")
    If i = vbNo Then Exit Sub
    i = 0
    Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryption Names") For Output As #1 'Removes
Encryption's name from Registry file
        For i = 0 To EncryptionList.ListCount - 1
            If EncryptionList.List(i) <> SelectedEncryption.Name Then Write #1, EncryptionList.List(i)
        Next i
    Close #1
    EncryptionList.Clear
    Kill (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryptions\" & SelectedEncryption.Name)
'Deletes Encryption's encrypted file
    If Encryptions.Count > 0 Then
        i = 1
        For i = 1 To Encryptions.Count 'Searches for correct Encryption
            If Encryptions.Item(i).Name = SelectedEncryption.Name Then
                Encryptions.Remove i 'Removes Encryption from Encryptions Collection
                Exit For
            End If
```

```
      Next i
      i = 1
      For i = 1 To Encryptions.Count 'Replenishes Encryption List on form
         EncryptionList.AddItem (Encryptions.Item(i).Name)
      Next i
   End If
   SelectedEncryption.BackupPassword = "" 'Clears Encryptions data
   SelectedEncryption.CreationDate = ""
   SelectedEncryption.EDate = ""
   SelectedEncryption.Encryption = ""
   SelectedEncryption.ENumber = 0
   SelectedEncryption.Name = ""
   SelectedEncryption.EScript = ""
   SelectedEncryption.ProjectOwner = ""
   SelectedEncryption.ProjectName = ""
   SelectedEncryption.Password = ""
   SelectedEncryption.Status = ""
   ENE.Text = ""
   EPWE.Text = ""
   MsgBox "Encryption deleted successfully!", , "[^_^]"
End Sub

Private Sub RemoveName_Click()
   Dim i As Integer, e As Integer
   For i = 0 To APNL.ListCount - 1
      If APNL.List(i) = APNE.Text Then APNL.RemoveItem (i)
   Next
End Sub

Private Sub SaveEncryption_Click() 'Saves changes that the User made to the Encryption
   Dim i As Integer, e As Integer
   Dim Temp As String
   If SelectedEncryption.Name <> "" Then
      For i = 1 To Encryptions.Count 'Searches for correct Encryption
         If Encryptions.Item(i).Name = SelectedEncryption.Name Then
            Kill (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryptions\" &
SelectedEncryption.Name) 'Deletes Encryption's encrypted folder
            SelectedEncryption.Name = ENE.Text 'Loads Encryption's new name and password
            SelectedEncryption.Password = EPWE.Text
            ProjectBrowser.SelectedEncryption.BackupPassword = SelectedEncryption.BackupPassword 'Loads all of Encryption's data to
Encrypter
            ProjectBrowser.SelectedEncryption.CreationDate = SelectedEncryption.CreationDate
            ProjectBrowser.SelectedEncryption.Encryption = SelectedEncryption.Encryption
            ProjectBrowser.SelectedEncryption.EDate = SelectedEncryption.EDate
            ProjectBrowser.SelectedEncryption.Name = SelectedEncryption.Name
            ProjectBrowser.SelectedEncryption.EScript = SelectedEncryption.EScript
            ProjectBrowser.SelectedEncryption.ProjectName = SelectedEncryption.ProjectName
            ProjectBrowser.SelectedEncryption.ProjectOwner = SelectedEncryption.ProjectOwner
            ProjectBrowser.SelectedEncryption.ENumber = SelectedEncryption.ENumber
            ProjectBrowser.SelectedEncryption.Password = SelectedEncryption.Password
            ProjectBrowser.SelectedEncryption.Status = SelectedEncryption.Status
            Encryptions.Item(i).BackupPassword = SelectedEncryption.BackupPassword 'Loads all of Encryption's data into Encryptions
collection
            Encryptions.Item(i).CreationDate = SelectedEncryption.CreationDate
            Encryptions.Item(i).Encryption = SelectedEncryption.Encryption
            Encryptions.Item(i).EDate = SelectedEncryption.EDate
            Encryptions.Item(i).Name = SelectedEncryption.Name
            Encryptions.Item(i).EScript = SelectedEncryption.EScript
            Encryptions.Item(i).ProjectName = SelectedEncryption.ProjectName
            Encryptions.Item(i).ProjectOwner = SelectedEncryption.ProjectOwner
            Encryptions.Item(i).ENumber = SelectedEncryption.ENumber
            Encryptions.Item(i).Password = SelectedEncryption.Password
            Encryptions.Item(i).Status = SelectedEncryption.Status
            Call ProjectBrowser.RecordEncryptionChange 'Saves change to Encryption
            Exit For
         End If
      Next i
      EncryptionList.Clear
      i = 1
      Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Encryption Names") For Append As #1
         For i = 1 To Encryptions.Count 'Re-Writes Project's Encryption registry file
```

```
            EncryptionList.AddItem (Encryptions.Item(i).Name) 'Refreshes list of Encryptions on form
            Write #1, Encryptions.Item(i).Name
         Next i
      Close #1
   End If
End Sub

Private Sub SaveProject2_Click()
   Dim i As Integer
   NewProjectCreation.NewProject.CreationDate = SelectedProject.CreationDate 'Loads all of the Project's data to NewProjectCreation
   NewProjectCreation.NewProject.EDate = SelectedProject.EDate
   NewProjectCreation.NewProject.Encryption = SelectedProject.Encryption
   NewProjectCreation.NewProject.ENumber = SelectedProject.ENumber
   NewProjectCreation.NewProject.EType = SelectedProject.EType
   NewProjectCreation.NewProject.Name = SelectedProject.Name
   NewProjectCreation.NewProject.Owner = SelectedProject.Owner
   NewProjectCreation.NewProject.Password1 = SelectedProject.Password1
   NewProjectCreation.NewProject.Password2 = SelectedProject.Password2
   NewProjectCreation.NewProject.Status = SelectedProject.Status
   Call SelectedProject.Encrypt
   Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Users") For Output As #1
      Write #1, SelectedProject.EType 'Writes Project's current mode to file
      For i = 0 To APNL.ListCount - 1 'Writes all "Allowable" Users' names to file
         Write #1, APNL.List(i)
      Next i
   Close #1
   MF2.Visible = False
   MF1.Visible = True
End Sub

Private Sub SaveProject1_Click()
   Dim i As Integer
   SelectedProject.Name = PNE.Text 'Loads Project's new properties
   SelectedProject.Password1 = PPWE1.Text
   SelectedProject.Password2 = PPWE2.Text
   If Option1.Value = True Then
      SelectedProject.EType = "Private"
   ElseIf Option2.Value = True Then
      SelectedProject.EType = "Limited"
      MF1.Visible = False
      MF2.Visible = True
      APNE.SetFocus
      Exit Sub
   Else:
      SelectedProject.EType = "Public"
   End If
   Open (Insat & "Projects\" & SelectedProject.Owner & "\" & SelectedProject.Name & "\Users") For Output As #1
      Write #1, SelectedProject.EType
      If SelectedProject.EType = "Limited" Then
         For i = 0 To APNL.ListCount - 1 'Writes Project's current mode to file
            Write #1, APNL.List(i) 'Writes all "Allowable" Users' names to file
         Next i
      End If
   Close #1
   i = 1
   Call SelectedProject.Encrypt
   For i = 1 To ProjectBrowser.Projects.Count 'Searches for correct Project in Projects collection
      If ProjectBrowser.Projects.Item(i).Name = SelectedProject.Name Then
         ProjectBrowser.Projects.Item(i).CreationDate = SelectedProject.CreationDate 'Loads Projects data to ProjectBrowser
         ProjectBrowser.Projects.Item(i).EDate = SelectedProject.EDate
         ProjectBrowser.Projects.Item(i).Encryption = SelectedProject.Encryption
         ProjectBrowser.Projects.Item(i).ENumber = SelectedProject.ENumber
         ProjectBrowser.Projects.Item(i).EType = SelectedProject.EType
         ProjectBrowser.Projects.Item(i).Owner = SelectedProject.Owner
         ProjectBrowser.Projects.Item(i).Password1 = SelectedProject.Password1
         ProjectBrowser.Projects.Item(i).Password2 = SelectedProject.Password2
         ProjectBrowser.Projects.Item(i).Status = SelectedProject.Status
         Exit For
      End If
   Next i
End Sub
```

```
Private Sub ViewEncryption_Click()
    Decrypter.Show
    Decrypter.SelectedEncryption.CreationDate = SelectedEncryption.CreationDate 'Loads all Encryption's data to Decrypter
    Decrypter.SelectedEncryption.EDate = SelectedEncryption.EDate
    Decrypter.SelectedEncryption.Encryption = SelectedEncryption.Encryption
    Decrypter.SelectedEncryption.ENumber = SelectedEncryption.ENumber
    Decrypter.SelectedEncryption.Name = SelectedEncryption.Name
    Decrypter.SelectedEncryption.EScript = SelectedEncryption.EScript
    Decrypter.SelectedEncryption.ProjectName = SelectedEncryption.ProjectName
    Decrypter.SelectedEncryption.ProjectOwner = SelectedEncryption.ProjectOwner
    Decrypter.SelectedEncryption.Password = SelectedEncryption.Password
    Decrypter.SelectedEncryption.BackupPassword = SelectedEncryption.BackupPassword
    Decrypter.SelectedEncryption.Status = SelectedEncryption.Status
    Call SelectedEncryption.DecryptScript 'Decrypts Encryption
End Sub
```

Appendix M: NewUserRegistry Pictures and Code



Welcome to the New User Registry! Here you can register yourself so that you can create and use projects in Saati Encryption Center. The process is simple and straight-forward: all you need to do is enter a user-name and a password in the appropriate areas bellow, and you're all set!

Please enter your new user-name here:

Enter your new Password here:

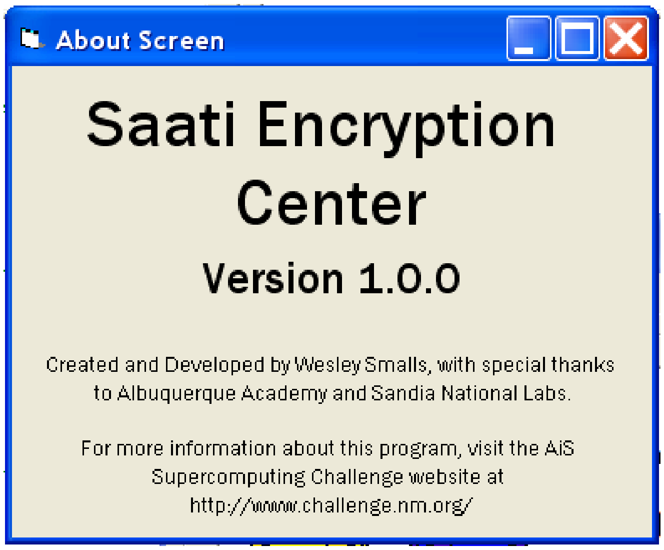
Confirm your Password here:

Please use at least 6 characters in your password!

```
Private Sub Cancel_Click()
   Text1.Text = ""
   Text2.Text = ""
   Text3.Text = ""
   NewUserRegistry.Hide
End Sub

Private Sub Register_Click()
   On Error Resume Next 'Error handler
   Dim User1 As New CUser
   Dim Temp As String
   Dim Good As Boolean
   If Text1.Text = "" Then
      MsgBox "Please enter another user-name.", , "[>_<] Invalid user-name"
      Text1.SetFocus
      Exit Sub
   End If
   If Len(Text2.Text) < 6 Then
      MsgBox "Please make your Password longer!", , "[>_<] Invalid Password"
      Text2.Text = ""
      Text3.Text = ""
      Text2.SetFocus
      Exit Sub
   End If
   If Text2.Text = "" Or Text3.Text = "" Or Text2.Text <> Text3.Text Then
      Text2.Text = ""
      Text3.Text = ""
      MsgBox "Please re-enter your password.", , "[>_<] Invalid Password"
      Text2.SetFocus
      Exit Sub
   End If
   Open (Insat & "\Users\Registered Users") For Input As #1
      Good = True
      Do While Not EOF(1)
         Input #1, Temp
         If Text1.Text = Temp Then
            Good = False
            Exit Do
         End If
      Loop
      If Good = False Then
         MsgBox "That user-name has already been chosen, please choose a different one.", , "[>_<] Invalid user-name"
         Text1.Text = ""
         Text1.SetFocus
```

```
            Close #1
            Exit Sub
        End If
    Close #1
    Open (Insat & "Users\Registered Users") For Append As #1
        Write #1, Text1.Text
    Close #1
    Set fso = CreateObject("Scripting.FileSystemObject")
    fso.CreateFolder (Insat & "Users\" & Text1.Text)
    fso.CreateFolder (Insat & "Projects\" & Text1.Text)
    Temp = Text2.Text
    User1.Name = Text1.Text
    User1.Password = Text2.Text
    Call User1.Encrypt
    Text1.Text = ""
    Text2.Text = ""
    Text3.Text = ""
    MainMenu.CreateNewProject.Enabled = True
    MainMenu.BrowseCurrentProjects.Enabled = True
    NewUserRegistry.Hide
    Open (Insat & "Projects\Projects") For Input As #1
        If EOF(1) = True Then MainMenu.BrowseCurrentProjects.Enabled = False
    Close #1
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Text2.SetFocus 'Enables "Tab" function when enter is pressed
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Text3.SetFocus 'Enables "Tab" function when enter is pressed
End Sub

Private Sub Text3_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then Call Register_Click 'Enables "Tab" function when enter is pressed
End Sub
```

Appendix N: Pictures of Forms without Code



Saati Encryption Center

Version 1.0.0

Created and Developed by Wesley Smalls, with special thanks to Albuquerque Academy and Sandia National Labs.

For more information about this program, visit the AiS Supercomputing Challenge website at http://www.challenge.nm.org/

Figure 1: The About Screen

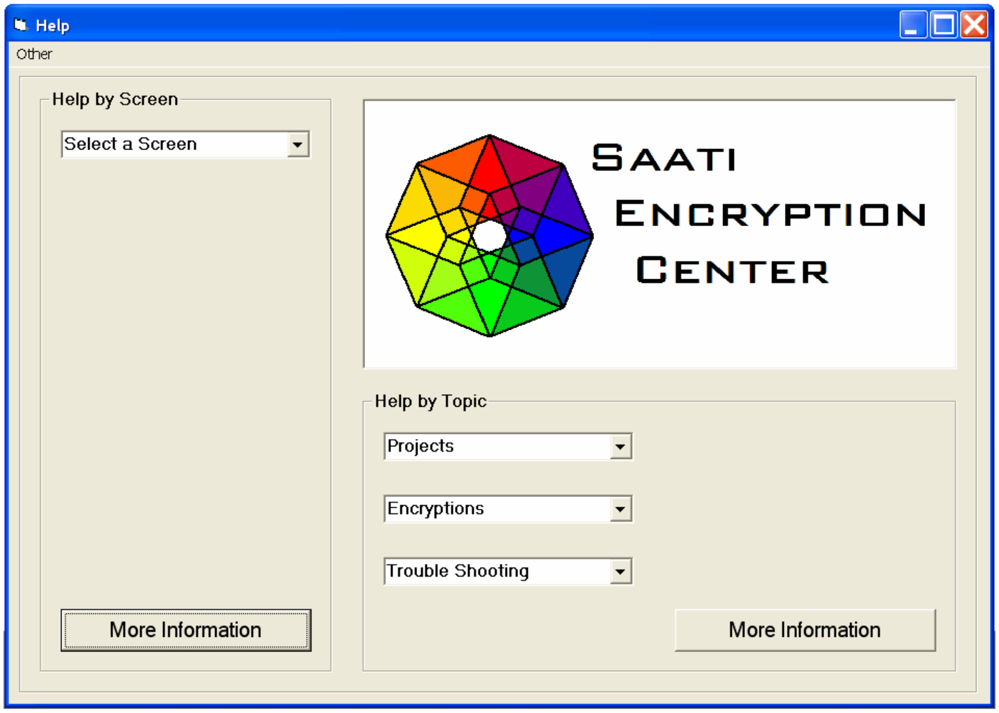
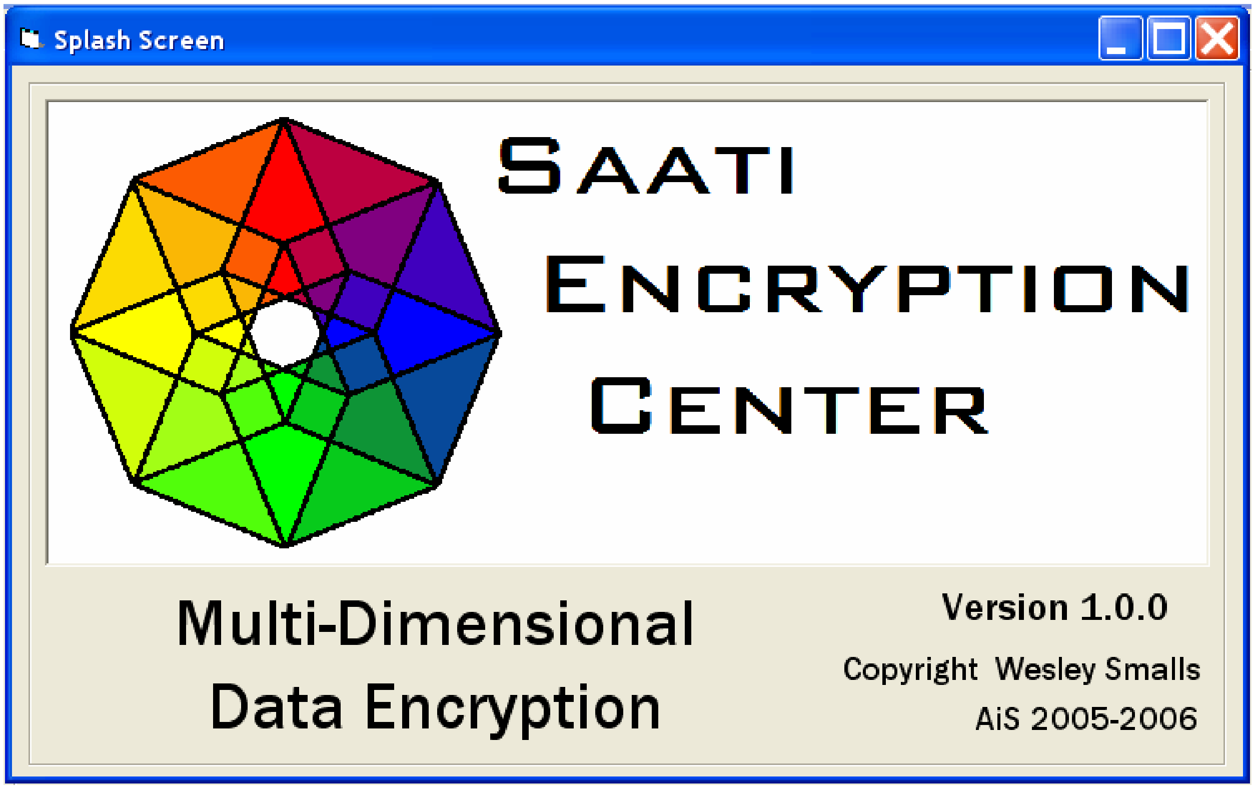


Figure2: The Help Screen



Splash Screen

Saati Encryption Center

Multi-Dimensional Data Encryption

Version 1.0.0

Copyright Wesley Smalls

AiS 2005-2006

Figure 3: The Splash Screen

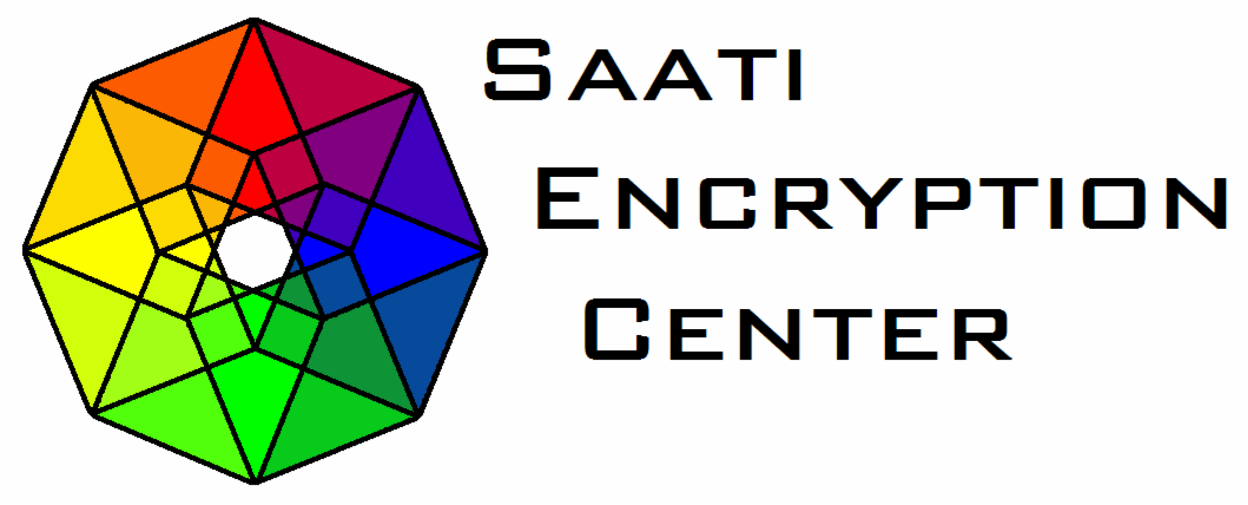


Figure 4: The project symbol, developed by me, appears on several forms. The shape is actually a 4-dimensional hypercube.