**CHAPTER 1**

**INTRODUCTION**

Diffie Hellman algorithm allows two parties who have not previously met to securely establish a key which they can use to secure their communication. This method was followed shortly afterwards by RSA, an implementation of public-key cryptography using asymmetric algorithms. Diffie–Hellman key exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network.

**1.1 Computer Graphics**

Computer graphics is a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Although the term often refers to the study of three-dimensional computer graphics. It also encompasses two-dimensional graphics and image processing. It focuses on the mathematical and computational foundations of image generation and processing rather than purely aesthetic issues. Computer graphics is often differentiated from the field of visualization.

Computer Graphics deals with generating images with aid of computers. Computer Graphics is a core technology in digital photography, film, video games, cell phone and computer display, and many specialized applications. It is a vast and recently developed area of computer science. The phrase was coined in 1960 by computer graphics researchers Verne Hudson and William Fetter of Boeing. It is often abbreviated as CG, or typically in the context of the film as computer generated imagery. The non-artistic aspects of computer graphics are the subject of computer science research.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers themselves. It has grown to include the creation, storage and manipulation of models and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive. The user controls the contents, structure and appearance of objects and their displayed images by using input devices, such as a keyboard, mouse, or touch sensitive panel on the screen. The handling of such devices is

included in the study of computer graphics, because of the close relationship between the input devices and the display.

**1.2 OpenGL**

OpenGL (Open Graphics Library) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit, to achieve hardware-accelerated rendering. Silicon Graphics, Inc. (SGI) began developing OpenGL in 1991 and released it on June 30,1992. Applications use it extensively in the fields of computer-aided design, virtual reality, scientific visualization, information visualization, flight simulation and video games.

OpenGL is an actively developed API. New versions of the OpenGL specifications are regularly released by the Khronos group each of which extends the API to support various new features. The details of each version are decided by consensus between the Group's members, including graphics card manufacturers, operating system designers, and general technology companies such as Mozilla and Google.

With OpenGL, you must build up your desired model from a small set of geometric primitives - points, lines, and polygons. A sophisticated library that provides these features could certainly be built on top of OpenGL. The OpenGL Utility Library (GLU) provides many of the modeling features, such as quadric surfaces and NURBS curves and surfaces. GLU is a standard part of every OpenGL implementation. Also, there is a higher-level, object-oriented toolkit, Open Inventor, which is built atop OpenGL, and is available separately for many implementations of OpenGL. Similarly, OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules.

**1.3 Problem Statement**

Diffie Hellman key exchange algorithm establishes a shared secret between two parties, that can be used for secret communication for exchanging data over public networks.

Exchanging the keys between two communicating parties securely, we do not just want to establish a common key but don’t want to do so in such a way that anyone who is listening to communication between the devices do not find out the key.

**1.4 Overview of the project**

Diffie–Hellman key exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network. An analogy illustrates the concept of public key exchange by using colors instead of very large numbers.

The process begins by having the two parties, Alice and Bob, agree on an arbitrary starting color that does not need to be kept secret (but should be different every time). In this example, the color is yellow. Each person also selects a secret color that they keep to themselves – in this case, red and blue. The crucial part of the process is that Alice and Bob each mix their own secret color together with their mutually shared color, resulting in orange and green mixtures respectively, and then publicly exchange the two mixed colors. Finally, each of the two mixes the color they received from the partner with their own private color. The result is a final color mixture (brown in this case) that is identical to the partner's final color mixture.

If a third party listened to the exchange, it would only know the common color (yellow) and the first mixed colors (orange and green), but it would be difficult for this party to determine the final secret color (brown). Bringing the analogy back to a real-life exchange using large numbers rather than colors, this determination is computationally expensive. It is impossible to compute in a practical amount of time even for modern supercomputers.

**1.5 Applications of Diffie Hellman Key Exchange**

* These keys can then be used with symmetric-key algorithms to transmit information in a protected manner. Symmetric algorithms tend to be used to encrypt the bulk of the data because they are more efficient than public key algorithms.
* Technically, the Diffie-Hellman key exchange can be used to establish public and private keys.
* Diffie-Hellman key exchange is frequently implemented in security protocols such as TLS, IPsec, SSH, PGP, and many others. This makes it an integral part of our secure communications.
* As part of these protocols, the Diffie-Hellman key exchange is often used to help secure your connection to a website, to remotely access another computer, and for sending encrypted emails.

**CHAPTER 2**

**SOFTWARE REQUIREMENT SPECIFICATION**

Software Requirement Specification specifies the requirements required to run the given web application. The detailed explanation of each type of requirement is given below.

**2.1 Functional Requirements**

For the execution of this project, the graphics has been written in C language and many simple user defined functions are used. The project requires access to OpenGL utility toolkit through the use of the header file “GL /glut.h”. This header file, in addition to the usual header files is needed for the working of the project. For running the program, any basic C running compatible version of Code Blocks or Linux (Ubuntu) based platform is sufficient.

**2.2 Non Functional Requirements**

The software should not accept wrong inputs and produce outputs. This should be meticulously taken care of. It should use memory as less as possible. For this, dynamic memory allocation is preferable to accomplish this task. The project needs to be memory efficient and be able to produce quality output at the same time.

**2.3 Hardware Requirement Specification**

* PROCESSOR: Intel®core™ 2 Duo
* SPEED: 2.10GHz
* RAM: 2.00 GB Minimum
* SPACE ON DISC:20GB Minimum

**2.4 Software Requirement Specification**

* OpenGL libraries
* C language compiler
* Operating System: Windows
* Code::Blocks

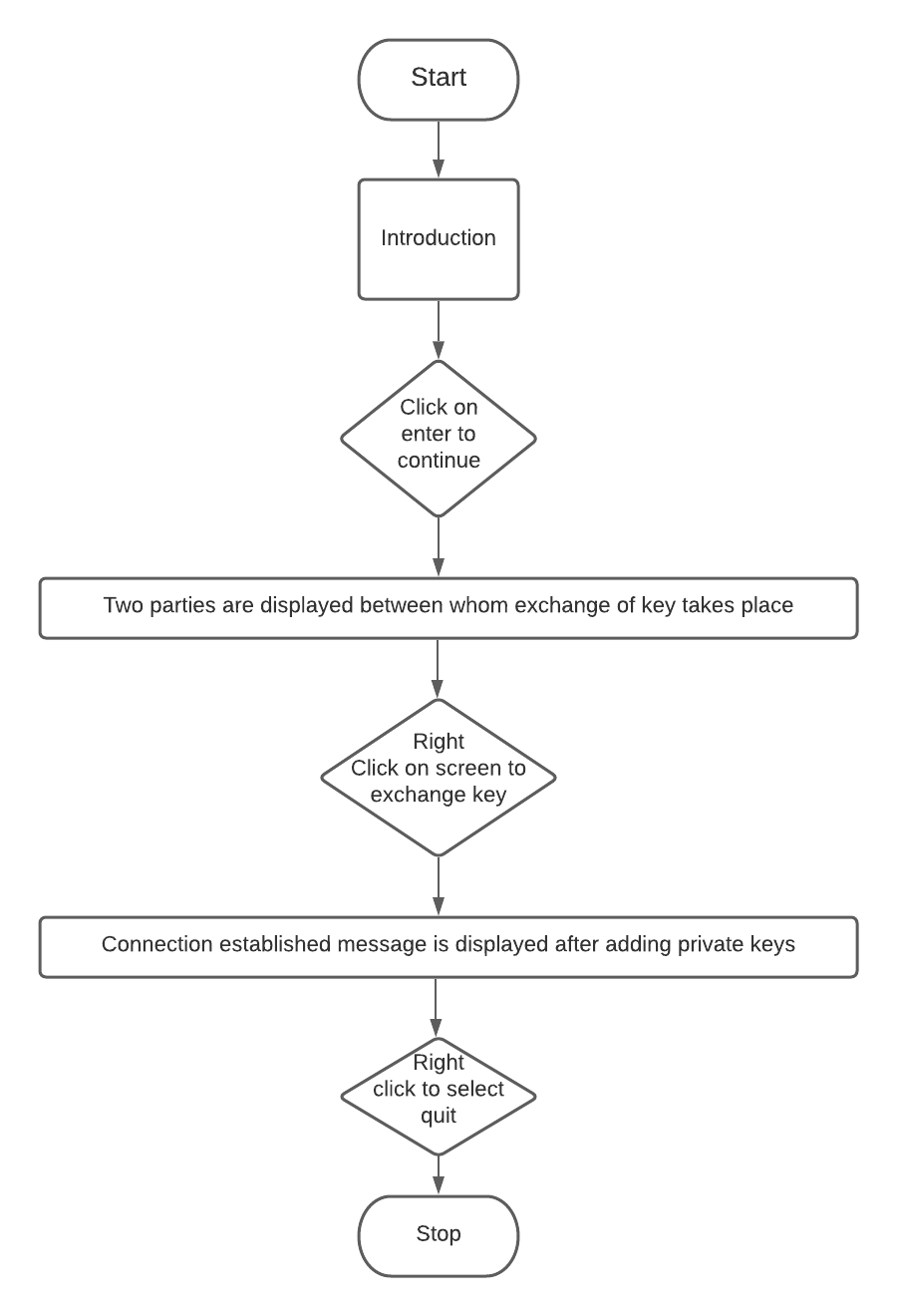
**CHAPTER 3**

**SYSTEM DESIGN**

The system design includes the description of the commands used, which gives an intuitive definition of a system. It shows how the functions are associated with the program. This project consists of many user defined functions which defines each object like drawing computers etc.

**3.1 Flowchart**

Flowchart is a common type of chart that represents an algorithm or process showing the steps as boxes of various kinds, their order by connecting these with arrows.



**Figure 3.1** Flowchart

* The above figure 3.1 shows the flow chart of the project.
* At first when we run the program, a cover page will be displayed which will have title of the project and team member name.
* When we press the S key, it goes to the simulation page.
* The simulation page will show how two parties who have not previously met to securely establish a key which they can use to secure their communications.
* By selecting the Exchange public keys menu, we can view how the keys will be exchanged on the screen between Alice and Bob.
* Alice and Bob will agree with public key(yellow) which is visible to everyone.
* Selecting the menu named Add your private key, Alice and Bob will add the private key(red and blue) which is not visible to everyone.
* They disguise their private key by mixing it with public key to obtain orange and green.
* That shows connection has been established between Alice and Bob which will be a common secret brown.

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 Description of Implementation Modules**

In this project we have created a colour analogy of Diffie Hellman Algorithm using “OpenGL” functional API. We have taken the help of OpenGL built in functions and OpenGL User Defined functions. To provide functionality to our project we have used sub functions. These functions provide us the efficient way to design the project. In this chapter we are describing the functionality of our project using OpenGL Built in functions and OpenGL User Defined functions.

**4.2 Implementation of OpenGL Built-In functions**

* **glutCreateWindow(char\*name) –** Creates the window on the display. The string title can be used to label the window.
* **glutInitWindowSize(int width, int height) –** It specifies the initial height and width of the window in pixel.
* **glutInitWindowPosition(int x, int y) –** It specifies the initial position of the top left corner of the window.
* **glColor3 [b i f d ub ui] (TYPE r, TYPE g, TYPE b) –** Sets the present RGB or RGBA colors. Valid types are byte(b), int(i), float(f), double(d), unsigned byte(ub), unsigned int(ui). The maximum and minimum values of the floating point types are 1.0 and 0.0 respectively.
* **glutInit(&argc,argv) –** It takes arguments.
* **glutKeyBoardFunc(void \*(unsigned char key, int x, int y)func) –** It registers the keyboard callback function, takes the ASCII code of the key pressed and the position of the mouse.
* **glutMainLoop() –** It causes the program to enter tan event-processing loop, It should be the last statement in the main.
* **glutDisplayFunc(void(\*func)(void)) –** Registers the display function that is executed when the window needs to be redrawn.
* **glClear(GLbitfieldmask) –** Clears the specified buffers to their current clearing values. The mask argument is a bitwise logical OR combination of values i.e. GL\_COLOR\_BUFFER\_BIT,GL\_DEPTH\_BUFFER\_BIT.
* **glLoadIdentity() –** Replaces the current matrix with the identity matrix.
* **glClearColor(GLfloatR,GLfloatG,GLfloatB,GLfloatalpha) –** This clears the background color to current clearing values.
* **VoidgluOrtho2D(GLfloatleft,GLfloatright,GLfloattop,GLfloatbotom) –** Sets the ortho as specified values in the function argument.
* **glFlush() –** Forces all previously issued OpenGL commands to begin execution thus guarantying that they complete in finite time.
* **glVertex2f(GLfloat x,GLfloat y) –** Specifies a vertex in 2D.
* **glRasterpos3f(TYPE \* coordinates) –** Specifies the raster position. Parameters are same as glVertex.
* **glutBitmapCharacter(void \*font int char) –** Renders the character with ASCII code char at the current raster position using the raster font given in the font.
* **glBegin(GL\_LINE\_LOOP) –** Draws a hollow diagram. Takes the end points from the glvertex function.
* **glBegin(GL\_POLYGON) –** Draws a filled polygon diagram. Takes the end points from the glertex function.
* **glBegin(GL\_QUADS) –** Treats each group of four vertices as an independent quadrilateral.
* **glMatrixMode(GLenum mode) –** Specifies which matrix will be affected by subsequent transformations. Mode can be GL\_MODELVIEW, GL\_PROJECTION.
* **glPushMatrix() –** It pushes the matrix stack corresponding to the current matrix mode.
* **glPopMatrix() –** It pops the matrix stack corresponding to the current matrix mode.
* **glEnd() –** Indicates the end of glBegin().

**4.3 Implementation of OpenGL User-Defined functions**

* **int main(int argc,char \*argv[]) –** This is the main routine of the program.
* **void keyfunc (unsigned char key, int x, int y) –** keyfunc() is keyboard function to take input from keyboard.
* **display() –** display function is used to display project title, credits and team members name.
* **home() –** home function is used to show the diffie-hellman key exchange simulation.
* **brownA() –** brownA function is used to create brown colored block for Alice.
* **brownB() –** brownB function is used to create brown colored block for Bob.
* **moveVertical() –** moveVertical function is used to move the public key blocks vertically.
* **moveHorizontal() –** moveHorizontal function is used to move the public key blocks horizontal.
* **yellow() –** yellow function is used to create yellow colored blocks for Alice, Bob and Intruder
* **red() –** red function is used to create red colored block for Alice.
* **blue() –** blue function is used to create blue colored block for Bob.
* **orange() –** orange function is used to create orange colored block for Alice.
* **green() –** green function is used to create green colored block for Bob.
* **success() –** success function is used to show green colored line when connection is established.
* **computer\_intruder() –** computer\_intruder function is used to draw intruders computer.
* **computer\_dest() –** computer\_dest function is used to draw Bob's computer.
* **computer() –** computer function is used to draw intruders Alice's computer.
* **output(int x,int y,char \*string) –** output function is used to display the string on the screen.
* **outputGreen(int x,int y,char \*string) –** outputGreen function is used to display the string on the screen having green color.
* **outputBG(int x,int y,char \*string) –** outputBG function is used to display the string on the screen having the background color.
* **clearArea(int u,int v,int w,int x) –** clearArea function is used to repaint the specified area with background color
* **delay(unsigned int mseconds) –** delay function is used to introduce delay between the excecution of different functions.

**4.4 Code Snippets**

void output(int x,int y,char \*string)

{

glColor3f(0.6,0.8,0.7);

int len,i;

glRasterPos2f(x,y);

len=(int) strlen(string);

for(i=0;i<len;i++)

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,string[i]);

glFlush();

}

void moveHorizontal(){

int i=0;

int j=630;

while (i<490 && j>80)

{

glColor3f(1,0.49,0); // Alice orange

glBegin(GL\_QUADS);

glVertex2f(pos1,325);

glVertex2f(pos2,325);

glVertex2f(pos2,300);

glVertex2f(pos1,300);

glEnd();

delay(2);

clearArea(pos1-70,pos2-65,325,300);

pos1+=1;

pos2+=1;

i+=1;

glColor3f(0,0.6,0.0); // Bob green

glBegin(GL\_QUADS);

glVertex2f(pos3,325);

glVertex2f(pos4,325);

glVertex2f(pos4,300);

glVertex2f(pos3,300);

glEnd();

delay(2);

clearArea(pos3+60,pos4+65,325,300);

pos3-=1;

pos4-=1;

j-=1;

glFlush();

}

}

void success(){

glColor3f(0, 1, 0);

for (int i = 180; i < 530; i++) {

glBegin(GL\_POLYGON);

glVertex2i(i, 380);

glVertex2i(i + 1, 380);

glVertex2i(i + 1, 390);

glVertex2i(i, 390);

glEnd();

glFlush();

delay(1);

}

int i=0;

while(i<50){

outputBlue(240,280+i,"CONNECTION ESTABLISHED !!");

delay(10);

outputBG(240,280+i,"CONNECTION ESTABLISHED !!");

i+=1;

}

outputBlue(240,280+i,"CONNECTION ESTABLISHED !!");

}

**CHAPTER 5**

**TESTING**

**5.1 Testing**

The primary purpose of testing is to detect software failures so that defects may be discovered and corrected. The scope of software testing often includes examination of codes as well as execution of that code in various environment.

**Table 5.1** Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Result Expected** | **Result Obtained** | **Remarks** |
| When Enter is Pressed | Simulation page appears | Simulation page appears | Successful |
| When Right click is pressed | Menu option appears | Menu option appears | Successful |
| When we select 1st option | Exchange of key takes place | Exchange of key takes place | Successful |
| When we select 2nd option | Adding of private keys takes place | Adding of private keys takes place | Successful |
| When we select Quit option | End of simulation takes place | End of simulation takes place | Successful |

**5.2 User Interface Testing**

Interface testing is done to cover all the functionalities of the system and ensure that this project’s graphical user interface meets its specification. We test the interface to perform, such as the movements of the virus and antivirus and also swapping of displays.

**5.3 System Testing**

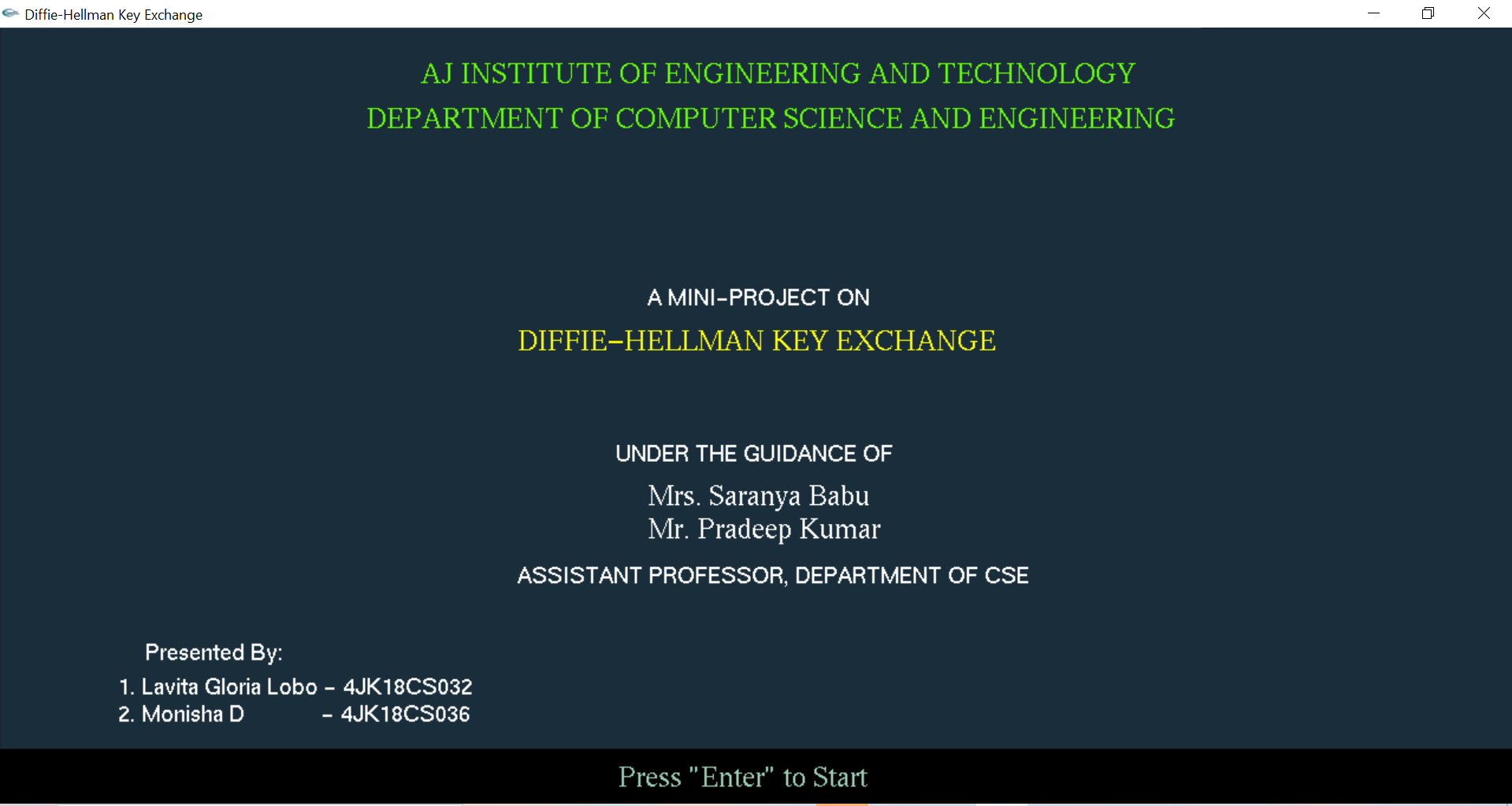
After testing the user interface, we can click into the system to test the program. We need to execute a number of times to make sure that the system does not have any bugs during execution. We check the compliance with the specified requirements.

**5.4 Performance**

The overall performance of the result is efficient and as expected. We need to test in different OS. The simulation has no delays and the total simulation time approximates to 1 minute.

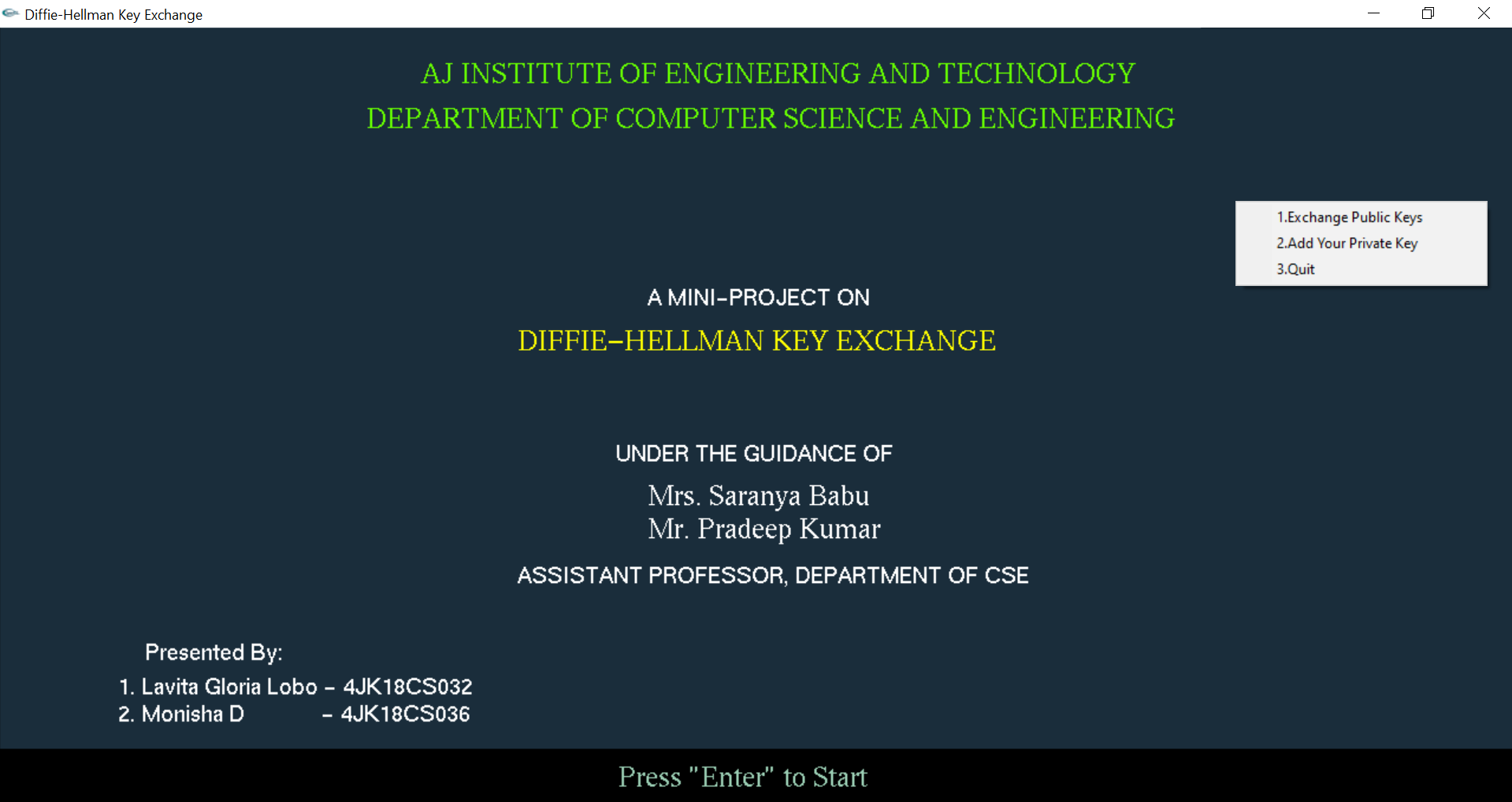
**CHAPTER 6**

**RESULTS**

****

**Figure 6.1** Project title

Figure 6.1 shows title of the project and team member’s name. When the Enter key is pressed the next screen will be displayed.

****

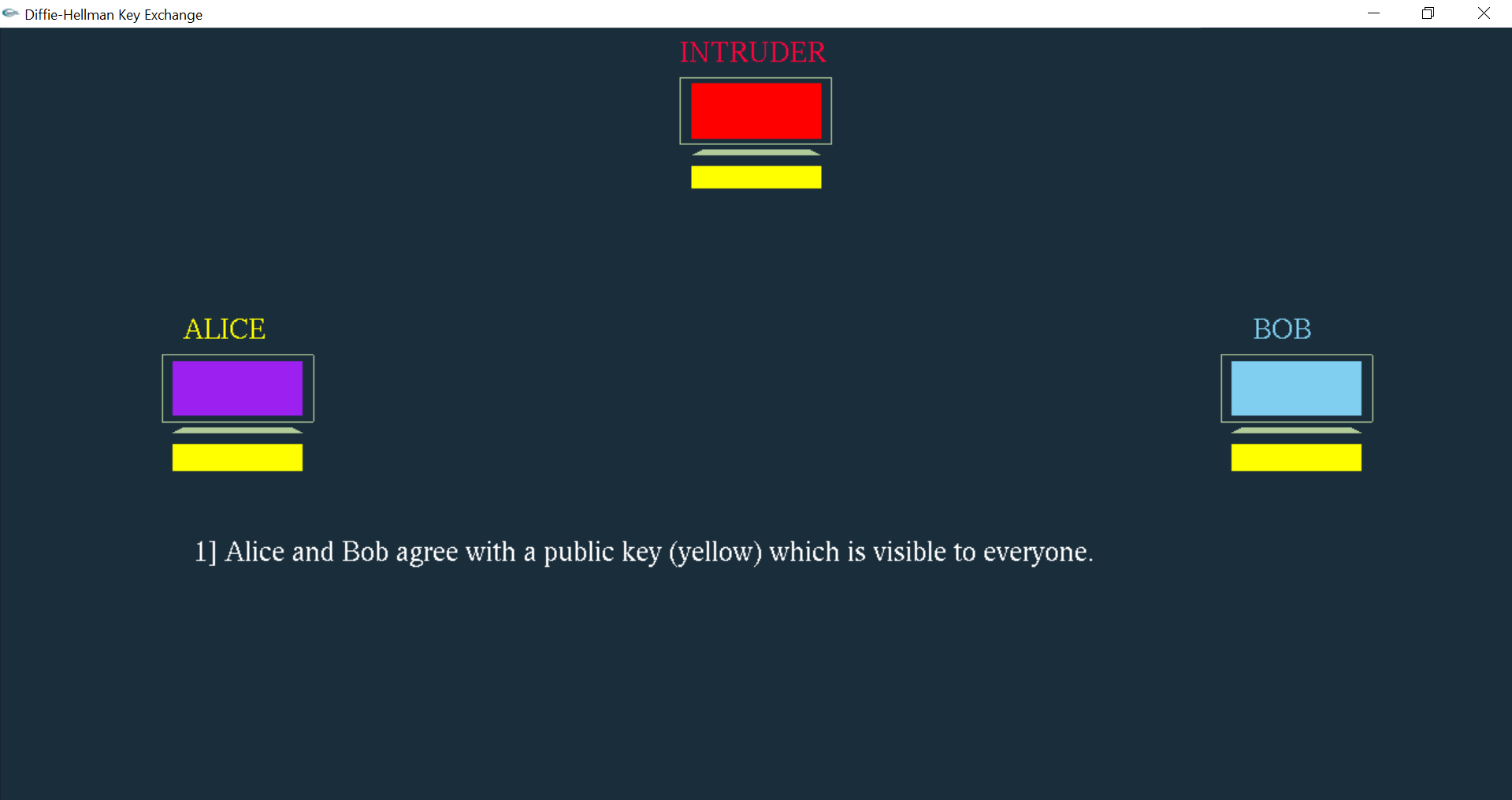
**Figure 6.2** Menus

Figure 6.2 shows the three menus which allows the user to select and see how the keys are actually exchanged.

****

**Figure 6.3** Parties before communication

Figure 6.3 shows how two parties who have not previously met to securely establish a key which they can use to secure their communications. By right clicking on the screen we can see the exchange of keys.

****

**Figure 6.4** Exchange of public keys

Figure 6.4 shows Alice and Bob agree upon a public key yellow color which is visible to everyone.

****

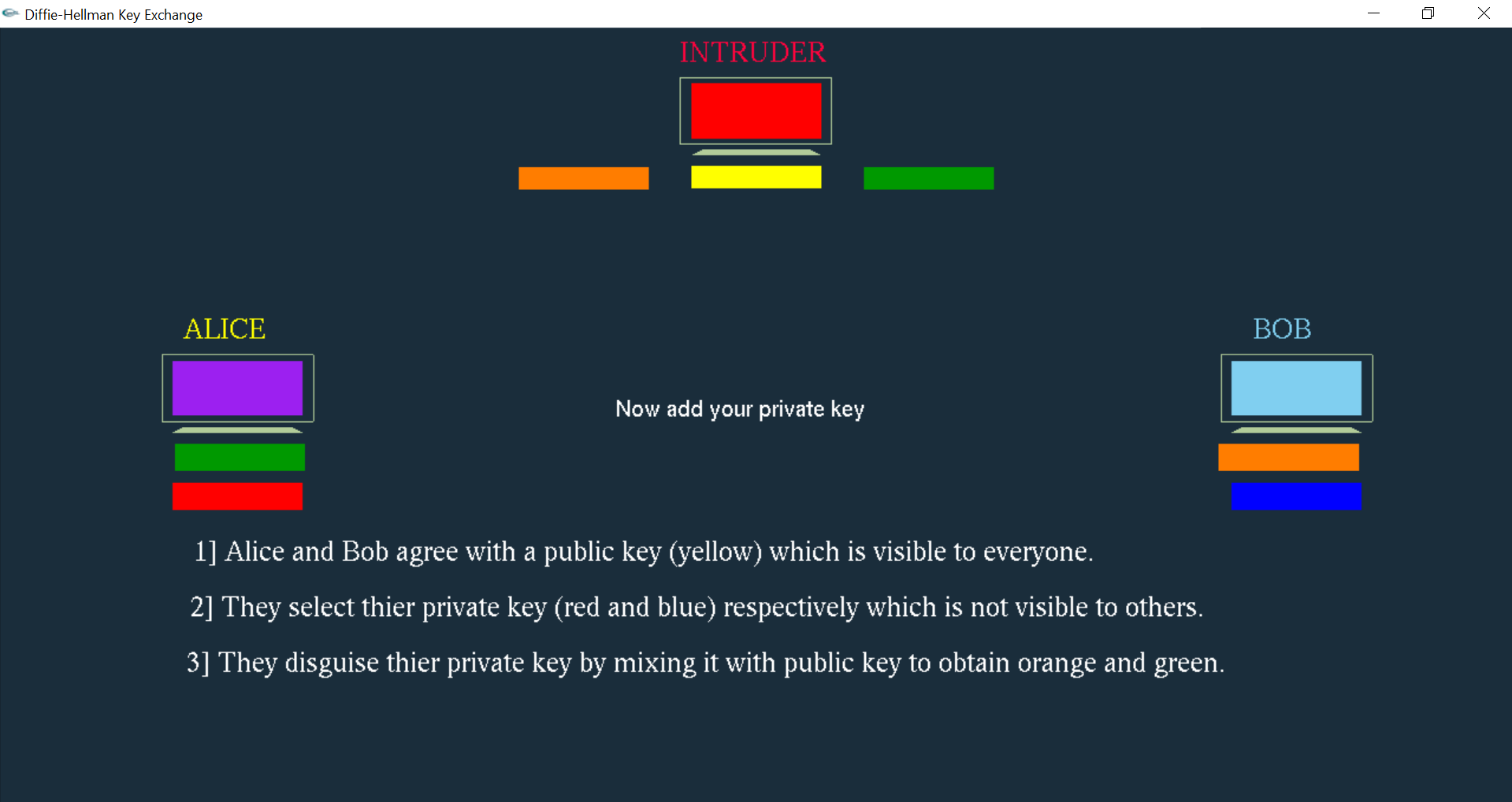
**Figure 6.5** Selection of private key

Figure 6.5 shows Alice and Bob select their private keys red and blue respectively.

****

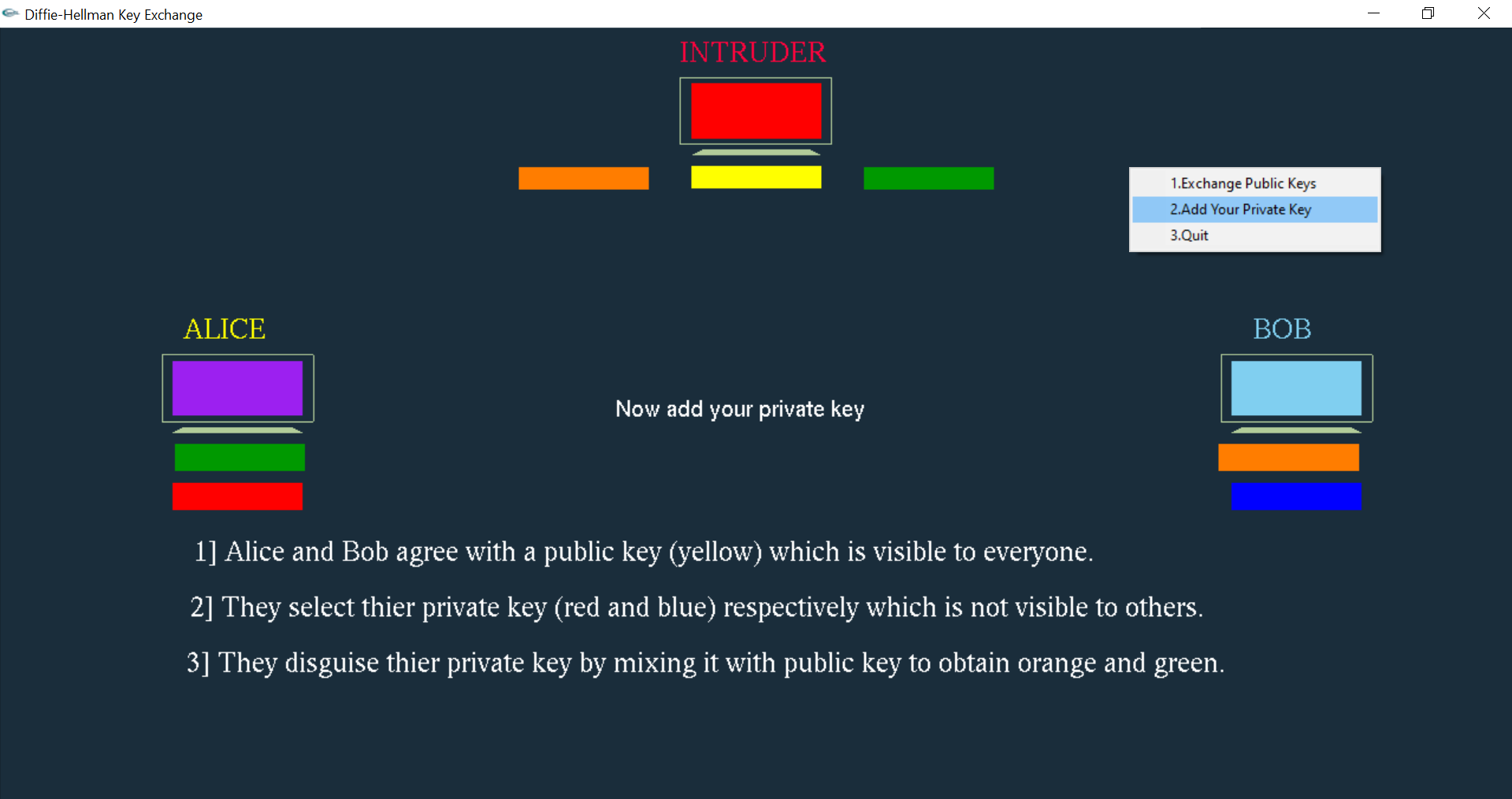
**Figure 6.6** Mixing of two different keys

Figure 6.6 shows how Alice and Bob disguise their private keys and mix it with the public key to obtain orange and green.

****

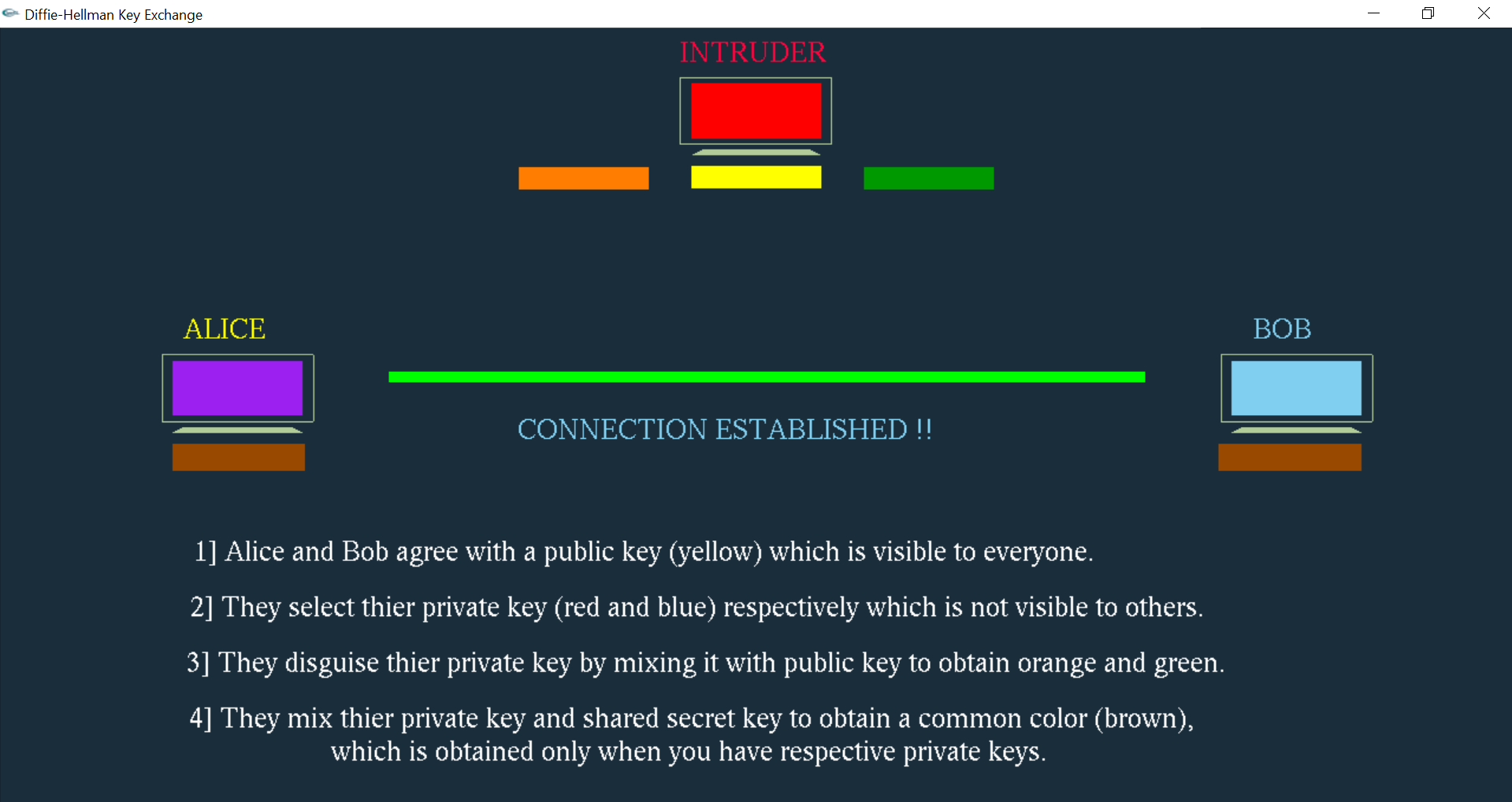
**Figure 6.7** Exchange of private keys

Figure 6.7 shows that how Alice and Bob interchange the results which they obtained from mixing two different keys which will also be visible to the intruder.

****

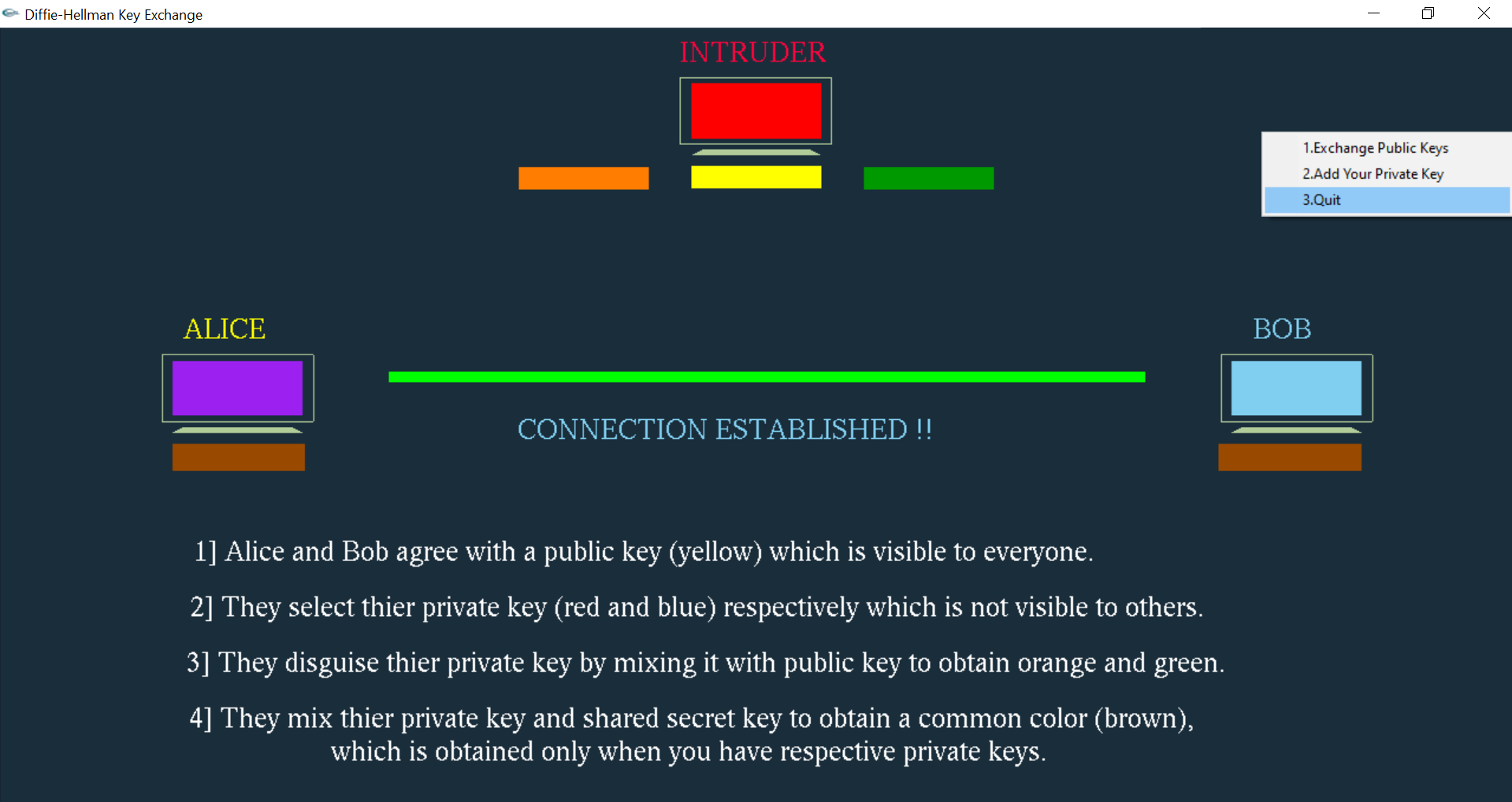
**Figure 6.8** Adding Private Key

Figure 6.8 shows how Alice and Bob add their private keys.

****

**Figure 6.9** Connection establishment

Figure 6.9 shows that connection has been established between Alice and Bob.

****

**Figure 6.10** Quit option

Figure 6.10 shows that it marks the end of the simulation.

**CHAPTER 7**

**CONCLUSION AND FUTURE ENHANCEMENT**

**7.1 Conclusion**

The project “DIFFIE-HELLMAN KEY EXCHANGE” has helped me in understanding about computer graphics using OpenGL basic functions which can be used to manipulate the data and provide some animations and various concepts and methodologies used in computer graphics. It also helped me to understand various inbuilt functions of OpenGL and allowed to implement my own. The keyboard and mouse interaction has helped me to implement this project in more easy way.

The Diffie-Hellman Key exchange was one of the most important development in public key cryptography and was the first widely used method of safety developing and exchanging keys over an insecure channel. And also, do it in such a way that anyone listening to communication between the devices don’t find the key. Developing this project helped me to learn about the computer graphics practically and to add realistic animations required for the project. The project is user friendly and has the features which can be easily understood by any user. The OpenGL is used for all the graphics display, coordinate transformations, viewport transformations and orthogonal projections.

The combination of computers, networks through computer graphics has led to new ways of displaying information, seeing virtual worlds, and communicating with people and machines. Hence, I conclude that the project ‘Diffie Hellman Key Exchange’ was successfully truest of my senses and best of my ability.

**7.2 Future Enhancement**

* I would make the scope of this project bigger by making it more interactive to the user.
* I would also like to modify the project by adding cryptographic technique using the numbers.

**REFERENCES**

[1] Edward Angle:Interactive Computer graphics A Top-Down Approach with OpenGL, 5th Edition,Addition – Wesley,2008.

[2] James D Foley, andries Van Dam, Steven K Feiner, John F Huges: Computer Graphics, Pearson Education.

[3] Donald Hearn and Pauline Baker: Computer Graphics-OpenGL Version,2 n

[4] Diffie Hellman: <https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange>.

[5] Color Analogy: <https://www.comparitech.com/blog/information-security/diffie-hellman-key-exchange/>.

[6] Working of Diffie Hellman: <https://www.youtube.com/watch?v=cM4mNVUBtHk>