Project Description

The purpose of our project was to create a program which would encrypt a plain text message into a completely unrelated, grammatically correct English message. The program would also be able to decrypt the message back into the original message. For example if one were to write, " Check your e-mail for viruses," the recipient would receive an encrypted message in English still that might say something like, "*The dog ran across the street.*" While seemingly an easy task, this encryption would require a number of complicated algorithms to enable the user to safely encrypt and decrypt various messages.

Summary

The project took some time and we faced many challenges as we attempted to encrypt a plain text message into another English message using C + +. At first, the idea seemed a bit too ambitious, for we discovered that we would have to discover a method for putting together grammatically correct English through a computer program based on already grammatically correct messages. Specifically speaking, the hardest portion of our problem came in how to put unrelated sentences (in terms of the original message) together by use of certain combinations of nouns, verbs, adjectives, and any other regular English part of speech. Of course the encrypted message would have to make sense, and ideally we planned on not having repeating sentences.

In our C+ + class, the members of our team had previously done a little work with encryption using a Caesar Shift and a Vigenere Shift, but the project we had decided to embark on would be far more difficult. With the information we had obtained in our C+ +

class we began to piece together a simple encryption program using a Caesar Shift to give us a skeletal structure of what our code needed to resemble. Our lead coder, Kelvin Dunn took it into his own hands to manifest most of the code and piece together our algorithm. The structure of the original Caesar Shift proved to be quite valuable for it was a sort of missing link we had gained, in order to link our knowledge of encryption with an actual coded representation in C+ +.

Method

The problem of forming a coherent English message from any other message seems a daunting one until you consider substitution. Using substitution, in combination with a set sentence output structure, fairly realistic encrypted messages can be produced. We decided, for the sake of simplicity, to substitute words for letters, and further decided to break the message into only its component letters, and perhaps move on to groups of letters later, increase the complexity of the encryption, and the variation among outputs. Since each letter will become a word, and the words must fit together to make an understandable sentence, we defined a basic sentence structure: Subject(person), verb(saying verb), quotation(what the person said). Then we created separate arrays for each type of word or each phrase composing the sentence, in this case there are three. Finally, as far as data storage goes, we created an array with all the characters the program expects to receive in any given message. With these set up, we used nested for loops with enclosed conditionals to do the actual encryption. The for loops compare each letter in the inputted message to the characters in the character array, when in finds a match it adds the corresponding word, the one with the same element number as the

character to the end of the output string, the encrypted messages. Whether or not that word is a subject, verb, or predicate is determined by the position of the letter in the original message.

With this basic structure defined for encryption, one can create many different sentence types and alternate between them randomly when encrypting the message, making it more realistic and harder to decode.

Progress and Challenges

In the beginning stages of our project, most of the thinking and brainstorming that we put on the table was simply far too ambitious, as we found out later in our project, while other ideas became successes. There was a lot of pseudocode which was stated, yet the thought of coding the actual project into C+ + felt like an unreal manifestation. For a few weeks the project began to die and most of us did not really even think about it. When mentioned that our AiS interim reports were due soon, our minds began to once again focus on the project we had committed ourselves to. We decided on a team meeting, and at our first meeting we came into synch once again and we also decided to continue the meetings, once a week. The project began to pick up speed and the Final Report date seemed to creep upon us. Each week we progressed in our code, finding new solutions and problems; what worked and didn’t work.

One problem we had was time while another, I remember specifically was a large scale idea we had in the early stages that didn’t make it through to the end. Originally, we had planned to write code for a user ID, which would recognize the user and give them a certain key for encrypting and decrypting messages between certain groups of users. This

idea, however was easier said than done, and the time constraints simply would not allow it.

For example as of the time of this report, we have not yet made another sentence type. As a result the outputs are repetitive enough to be suspicious. We also have discovered that decrypting the messages created by this program is extremely difficult to do, and have yet created working code for decryption. We do however have ideas concerning the way it might be done and plan to implement them before the final presentation.

Uses and Implementation

The most apparent and obvious implementation for the project is through the use of regular chat-rooms or e-mail which the user does not wish for outsiders to access the data in his/her message. The code acts as a privacy utility for users. The type of users would most likely be ordinary citizens chatting with each other. The usefulness of this type of encryption would be very successful in these cases because other users would be deceived by the simplicity and relative irrelevance of the message.

However, from a military or international standpoint the level of security of these messages would not be as high because the information being sent and received would be obviously of topic, and the encoded message would be discovered rather quickly.

Conclusions and Results

We discovered, as we were doing the project, that in order for the sentences to vary to a satisfying degree, vocabulary subsets must be created. For example the sentence, “She went ____” will be followed by a different phrase than “She said ___” and both have the same sentence structure, so two different vocabulary arrays need to be created and implemented into the encryption algorithm with additional conditionals. We learned many useful coding techniques along the way as well, for example writing code in short segments and testing them separates to make sure they work before putting the whole thing together. Also, saving the project each time the code is complicated but still works is almost a necessity so you can go back to those “guaranteed to work” states whenever you find yourself receiving errors that seem to come from nowhere.

We have, at the time of this report, a working encryption algorithm except for one logical error that we have been unable to correct. Some verbs to not appear, specifically the latter verbs of the verb array, when the message is encoded. Once we overcome this problem we plan to continue working on the decryption algorithm and on more encoding functions for different sentence types. Although the allotted time to finish this project is quite large, we have still experienced the “time squeeze” feeling, but we feel we are almost to the crest of a mental hill, and once we reach it we will be able to quickly modify expand the program and its functions.

Code

```
#include<iostream>
#include<stdlib.h>
```

```
#include<time.h>
using namespace std;
string prompt(string,int &);

void encode1(string);

int main()
{
 string message;
 int pause;
 int counter1;
 int approxlength;
 message = prompt(message,approxlength);
 int length = message.length();
 int sentences = length/3;
 int remainder = length%3;
 cout<<message<<endl;
 encode1(message);
 cin>>pause;
 return 0;
}
string prompt(string message, int &approxlength)//takes message from user
{
      cout<<"Please enter the approximate length of your message in characters."<<endl;
      cin>>approxlength;
      cout<<"Please enter your message."<<endl;
      while (message == "")
      {
      getline(cin,message);//anything after a newline will be cut off
      }
      return message;
}


void encode1(string message)
{
    int counter1;
    int pause;
    int counter2;
    string output;
    int length = message.length();
    string characters[28] =
{"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","
x","y","z"," ","."};
    string subject[28] =
{"Ann","Mary","Kim","Alex","Bob","Joe","Bill","Andrew","Jill","Sam","Kenneth","Pau
```

```
l","Tom","Sue","Carrie","Pat","Kyle","John","Ben","Jarrod","Jim","Jessica","Jordan","R
obert","Norbert","Duncan","Carl"};
    string verb[28] = {"went to","will go to","has gone to","already saw","might
see","saw","glimpsed","wants to go to","left for the","wants to see","left the","searched
the","said","told her","told
him","shouted","exclaimed","yelled","screamed","whispered","caught Ted saying","let
slip that","was like","made clear that","retorted","replied","responded","stated"};
    string togo[28] = {"the mall.","the grovery store.","the store.","the zoo.","Alpine
Park.","Terra Park.","the city.","the country.","the forest.","the club.","the circus.","the
photo finishers.","Lake Michigan.","New York.","Carolina.","New
Mexico.","Alabama.","the post office.","a restaurant.","Mcdonalds.","Wendy's .
","Burger King.","the sushi place.","Grandpa's house.","Grandma's house.","a
concert.","a crater.","the mountains."};
    string tosay[28] = {"'No!'","'Yes'!","'maybe'","'we'll see'","'who can
know?'","'perhaps'","'your terrible!'","'I love you'","'Amazing!'","'I have to leave.'","'I
don't want to!'","'You can't make me!'","'Sure!'","'What about insurance?'","'I'm off to
college'","'Wow, that is ridiculous'","'Too late'","'Watch me unravel.'","'Time for
fishing!'","'Didn't you know.'","'I forgot to tell you.'","'The mailman came.'","'Wanna
swingdance?'","'Completely ridiculous.'","'That doesn't matter.'","'Sounds
important'","'Come here.'","'Go there'"};

    for (counter1 = 0; counter1 < length; counter1 = counter1+3)
    {
       for (counter2 = 0; counter2 < 28; counter2 ++)
       {
          if(message.substr(counter1,1) == characters[counter2])
          {
             output = output + " ";
             output = output + subject[counter2];

          }
       }
       for (counter2 = 0; counter2 < 28; counter2 ++)
       {
          if(message.substr(counter1 + 1,1) == characters[counter2])
          {
           output = output + " ";
             output = output + verb[counter2];

             if (counter2 < 12)
             {
        for (counter2 = 0; counter2 < 28; counter2 ++)
                   {
                    if(message.substr(counter1 + 2,1) == characters[counter2])
                          {
                           output = output + " ";
```

```
                                output = output + togo[counter2];
                            }
                        }

                    }
                }
                else
                {
                if (counter2 > 11)
                {
                for (counter2 = 0; counter2 < 28; counter2 ++)
                {
                  if(message.substr(counter1 + 2,1) == characters[counter2])
                  {
                    output = output + " ";
                        output = output + tosay[counter2];
                  }
                }
            }
                }
             }
      }

     cout<<output;
}
```