1. **In search of an Unbreakable code:**

There exists many cyphers like **Caesar, Atbash, Simple substitution, Vigenère and others. But none of them are quite unbreakable. This leads to the need of an Unbreakable encryption system. If we take a look into the history of the world, during World War II, many encryption techniques were developed, which were almost unbreakable like the Vernam cipher and the Enigma machine. I was fascinated about the idea of Vernam cipher. The Vernam cipher is quite easy to create but very difficult to break. One day I was studying cross product of vectors and then I got the idea that what if we combine the Vernam cipher with the vector cross product rule thinking that the characters of the original text as axes and considering the numbers to represent the letters as the values for the corresponding axes and then apply the cross product rule with some modifications. To make the code more hard to break, I also made a decoy method. The decoy method prevents the normal people to predict the length of the original text.**

**For example: Let the word be “ARAB” and let the key be “MAIN” and decoy=5.**

**Then the encoding process will be like this:**

**0 1 2 3**

+-------+----+----+---+----+

0 | ARAB= | 0 | 17 | 0 | 1 |

+-------+----+----+---+----+

1 | MAIN= | 12 | 0 | 8 | 13 |

+-------+----+----+---+----+

**We represented each letter and the symbol ‘+’ (together called characters) in the table by numbers. The characters are represented as numbers from the following: A=0, B=1 , ……. , Z=25, a=26, b=27,…. , z=51, +=52. We will represent the values of the table by coordinates like (row number , column number). Example: (0,0) represents 0, (0,1) represents 17,…. Now first, we will close columns one by one from the left to right. We close 0 number column. Then we do the cross rule. First we do A={(0,0)+(1,1)+decoy} then B={(1,0)+(0,1)+decoy} but since the column 0 is closed and the two sets A and B contain the values of column 0, we ignore these two sets. Then A={(0,1)+(1,2)+decoy}={17+8+5}={30}={e} (as 30 is the number representation of e)**

**B={(1,1)+(0,2)+decoy}={0+0+5}={5}={F} (as 5 is the number representation of F)**

**Thus, the first and second letters of our encrypted text is e and F**

**A={(0,2)+(1,3)+decoy}={0+13+5}={18}={S} (as 18 is the number representation of S)**

**B={(1,2)+(0,3)+decoy}={8+1+5}={14}={O} (as 14 is the number representation of O)**

**Thus, the third and fourth letters of our encrypted text is S and O**

**In case, the value of the element of set A or B is greater than 52, we subtract it by 53.**

**We continue this process by closing each column one by one. And notice that, if we have added to same cells of the table earlier, we do not add it again.**

**The resultant encoded text will be: eFSOFi**

**To prevent anyone from getting the original length of key, we will add 5 random characters after the encoded text.**

**Now For decoding,**

**Since decoy=5, we will remove 5 letters from the end of the encoded text.**

**0 1 2 3**

+-------+----+----+---+----+

0 | XXXX= | | | | |

+-------+----+----+---+----+

1 | MAIN= | 12 | 0 | 8 | 13 |

+-------+----+----+---+----+

**eFSOFi** = 30 5 18 14 5 34

**We follow the similar process reversely with this.**

**Now first, we will close columns one by one from the left to right. We close 0 number column. Then we do the cross rule. First we try A={30-(1,1)-decoy} (which shall be equal to (0,0)) then B={5-(0,1)-decoy} (which shall be equal to (0,1)) but since the column 0 is closed and the set A tries to set the value of column 0, we ignore sets A and B.**

**Then A={30-(1,2)-decoy} (which shall be equal to (0,1))={30-8-5}={17}={R} (as 17 is the number representation of R)=(0,1)**

**B={5-(1,1)-decoy} (which shall be equal to (0,2))={5-0-5}={0}={A} (as 0 is the number representation of A)=(0,2)**

**A={18-(1,3)-decoy} (which shall be equal to (0,2))={18-13-5}={0}={A} (as 0 is the number representation of A)=(0,2)**

**B={14-(1,2)-decoy} (which shall be equal to (0,3))={14-8-5}={1}={B} (as 1 is the number representation of B)=(0,3)**

**In case, the value of the element of set A or B is less than 0, we add it by 53.**

**We continue this process by closing each column sequentially. Note that we have to check that we do not repeat the same cells again. To do this, suppose the cell we are putting value into is (0,j). Then we have to check that if (0,j) is empty or (0,j+1) is empty. If any one of them is empty, then we put the value, else not.**

1. **VectoRiya the java program:**

**import java.util.ArrayList;**

**class VectoRiya**

**{**

**//Main Method**

**public static void main(String[] args)**

**{**

**String toEncode="ARAB";//String to encode**

**String key="MAIN";//String to key**

**int decoy=5;**

**System.out.println("Encoding........");**

**try**

**{**

**String encoded = new VectoRiya().Encode(toEncode, key, decoy);//encoded text**

**System.out.println("Encoded.");**

**System.out.printf("Encoded message:\t%s%n",encoded);//Print encoded**

**System.out.println("Decoding........");**

**String decoded=new VectoRiya().decode(encoded,key,decoy);//Decode the encoded text**

**System.out.println("Decoded.");**

**System.out.printf("Decoded message:\t%s%n",decoded);//Print decoded**

**}**

**catch(Exception e)**

**{**

**System.out.println(e.getMessage());//errors**

**}**

**}**

**//Encode**

**public String Encode(String original,String key, int decoy) throws KeyLengthException**

**{**

**decoy=Math.abs(decoy);//decoy cannot be negative**

**//shift\_decoy is the amount of shift that has to be done to create the encoded text**

**/\*max value of shift decoy is calculated as follows:**

**the max value at an index of the calculation can be**

**52+52=104(if both original and key contains +).**

**Since it is greater than 52, 53 will be subtracted.**

**104-53=51**

**Then at the end, since 51 is greater than 26, 71 will be added to it.**

**So the max value at an index of encoded text be 51+71=122.**

**The max ascii value is 65535, we subtract 122 from it,**

**we get max value of decoy be 65413\*/**

**int shift\_decoy=(decoy>65413)?decoy-(65413\*(decoy/65413)):decoy;//shift\_decoy cannot be greater than 65413**

**original=RemoveSpecialCharacter(original);//Replace Special characters with +**

**if(key.length()!=original.length())//key length should be equal to original length**

**{**

**throw new KeyLengthException();**

**}**

**int calc[][]=new int[2][key.length()];//table to store the characters of original and key**

**for(int i=0;i<2;i++)**

**{**

**for(int j=0;j<key.length();j++)**

**{**

**if(i==0)**

**{**

**//first row to store the number representation of characters of original**

**if(original.charAt(j)=='+')**

**calc[i][j]=52;**

**else if(Character.isUpperCase(original.charAt(j)))**

**calc[i][j] = original.charAt(j) - 65;**

**else**

**calc[i][j] = original.charAt(j) - 71;**

**}**

**else**

**{**

**//second row to store the number representation of characters of key**

**if(key.charAt(j)=='+')**

**calc[i][j]=52;**

**else if(Character.isUpperCase(key.charAt(j)))**

**calc[i][j] = key.charAt(j) - 65;**

**else**

**calc[i][j] = key.charAt(j) - 71;**

**}**

**}**

**}**

**ArrayList<Integer> answer\_int=new ArrayList<>();//store the integers obtained to represent each character of the encoded text**

**ArrayList<ArrayList<Integer>> done=new ArrayList<>();//store the cells of tables that have already been done**

**//the encoding process**

**for(int i=0;i<key.length();i++)**

**{**

**for(int j=0;j<key.length()-1;j++)**

**{**

**if (j!=i && j+1!=i)**

**{**

**int first=calc[0][j]+calc[1][j+1]+shift\_decoy,second=calc[1][j]+calc[0][j+1]+shift\_decoy;**

**ArrayList<Integer> check=new ArrayList<>();**

**check.add(j);**

**check.add(j+1);**

**if (first > 52)**

**first = first - 53;**

**if (second > 52)**

**second = second - 53;**

**if(!done.contains(check))**

**{**

**answer\_int.add(first);**

**answer\_int.add(second);**

**}**

**done.add(check);**

**}**

**}**

**}**

**String answer\_final="";//represent each integer of answer\_int by characters**

**for(int i=0;i<answer\_int.size();i++)**

**{**

**if(answer\_int.get(i)==52)**

**answer\_final+='+';**

**else if(answer\_int.get(i)<26)**

**answer\_final+=(char)(answer\_int.get(i)+65);**

**else**

**answer\_final+=(char)(answer\_int.get(i)+71);**

**}**

**return decoy(answer\_final,decoy);//return final answer after putting a decoy**

**}**

**//Decode**

**public String decode(String encoded,String key,int decoy)**

**{**

**decoy=Math.abs(decoy);//decoy cannot be negative**

**//shift\_decoy is the amount of shift that has been done to create the encoded text**

**int shift\_decoy=(decoy>65413)?decoy-(65413\*(decoy/65413)):decoy;//shift\_decoy cannot be greater than 65413**

**encoded=anti\_decoy(encoded,decoy);//remove the decoyed letters**

**int calc[][]=new int[2][key.length()];//table for putting the integer representation of characters of encoded**

**for(int i=0;i<2;i++)**

**{**

**for(int j=0;j<key.length();j++)**

**{**

**if(i==0)**

**calc[i][j]=-500;//when row is 1, leave the cell with -500, which will later be used for calculating the original text**

**else**

**{**

**//second row to store the number representation of characters of key**

**if(key.charAt(j)=='+')**

**calc[i][j]=52;**

**else if(Character.isUpperCase(key.charAt(j)))**

**calc[i][j] = key.charAt(j) - 65;**

**else**

**calc[i][j] = key.charAt(j) - 71;**

**}**

**}**

**}**

**ArrayList<Integer> answer\_int=new ArrayList<>();//integer representation of characters of the encoded text**

**for(int i=0;i<encoded.length();i++)**

**{**

**//store integer representation of characters of encoded text to the ArrayList**

**if(encoded.charAt(i)=='+')**

**answer\_int.add(52);**

**else if(encoded.charAt(i)<97)**

**answer\_int.add(encoded.charAt(i)-65);**

**else**

**answer\_int.add(encoded.charAt(i)-71);**

**}**

**//decoding process**

**int counter=0;**

**for(int i=0;i<key.length();i++)**

**{**

**for(int j=0;j<key.length()-1;j++)**

**{**

**if (j!=i && j+1!=i)**

**{**

**if(calc[0][j]==-500 || calc[0][j+1]==-500)**

**{**

**calc[0][j] = answer\_int.get(counter++) - calc[1][j + 1]-shift\_decoy;**

**calc[0][j + 1] = answer\_int.get(counter++) - calc[1][j]-shift\_decoy;**

**}**

**if(calc[0][j]<0)**

**calc[0][j]=calc[0][j]+53;**

**if(calc[0][j+1]<0)**

**calc[0][j+1]=calc[0][j+1]+53;**

**}**

**}**

**}**

**String answer\_final="";//to store the answer**

**for(int i=0;i<key.length();i++)**

**{**

**//convert the integers of the answer from the table to Letters**

**if(calc[0][i]==52)**

**answer\_final+='+';**

**else if(calc[0][i]<26)**

**answer\_final+=(char)(calc[0][i]+65);**

**else**

**answer\_final+=(char)(calc[0][i]+71);**

**}**

**return answer\_final;//return answer**

**}**

**//Change all characters except letters to +**

**String RemoveSpecialCharacter(String s)**

**{**

**String answer="";**

**for(int i=0;i<s.length();i++)**

**{**

**char c=s.charAt(i);**

**if(Character.isLetter(c))**

**answer+=c;**

**else**

**answer+="+";**

**}**

**return answer;**

**}**

**//decoy**

**String decoy(String s,int n)**

**{**

**//add n number of random letters to create a decoy**

**for(int i=0;i<n;i++)**

**{**

**if(s.contains("+"))**

**{**

**if(getRandomNumber(0,10)%7==0)**

**s += "+";**

**else**

**{**

**if(getRandomNumber(0,10)%2==0)**

**s+=(char) getRandomNumber(65, 91);**

**else**

**s+=(char) getRandomNumber(97, 123);**

**}**

**}**

**else**

**{**

**if(getRandomNumber(0,10)%2==0)**

**s+=(char) getRandomNumber(65, 91);**

**else**

**s+=(char) getRandomNumber(97, 123);**

**}**

**}**

**return s;**

**}**

**//anti\_decoy**

**String anti\_decoy(String s,int n)**

**{**

**return s.substring(0,s.length()-n);//remove the decoy letters**

**}**

**//random number generator**

**int getRandomNumber(int min, int max)**

**{**

**return (int) ((Math.random() \* (max - min)) + min);**

**}**

**//exceptions**

**static class KeyLengthException extends Exception{**

**public KeyLengthException() {**

**super();**

**}**

**@Override**

**public String getMessage() {**

**return "Length of key cannot be different from the length of String";**

**}**

**}**

**}**

*Output:*

*Encoding........*

*Encoded.*

*Encoded message: eFSOFiSXOSW*

*Decoding........*

*Decoded.*

*Decoded message: ARAB*

1. **Importance:**

Let us consider, we have a text field that opens a file if the password is correct. Suppose the encoded form of the password is known to a person. Let us consider from the previous example, the password is ‘ARAB’, key is ‘MAIN’, and decoy is 5. The person knows only the encoded form which is ‘eFSOFiSXOSW’. Now,

Firstly the person doesn’t know decoy. So, he has to choose decoy from 1 to 2147483647 (theoretically).

For each decoy, he has to construct the encoded message without decoy and create Keys. Suppose he got ‘EmSxci’ as the encoded message without decoy. Now, the person has to construct keys like AAAA, AAAB, …… , AAAZ, AAAa, …….,AAAz, AAA+ You can understand this pattern. Now he has to do this for every places.

Then for every key generated, he has to create the decoded word. Then check that decoded word in the text field to open that file. Therefore, this is a vast process as millions of possibilities can exist. To enhance the safety of the file, obviously, the text field should block input for some time if suppose the user has typed 10 incorrect passwords. Therefore ensuring maximum security.

1. The greatest importance of this method is that there are two levels of security. First the key, then the decoy. If a person knows the key but doesn’t know the decoy, it will still be hard to break the code.
2. The original length of the message cannot be predicted if the decoy does not equal to zero.

But obviously this encrypting system is not totally unbreakable as we can say that there exists no single system in the world which does not have a flaw. I think that the code that can be constructed, can also be broken.