VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

CSE4003-CYBER SECURITY

Project Component

Creating and managing of strong passwords and Modifying the Caesar cipher

SUBMITTED TO:

Prof. JOSHVA DEVADAS T

SUBMITTED BY:

PAVITRA GOYAL: 19BCE373

DEVANSH SAINI: 19BCE0412

SHATAYU MITTAL: 19BCE0446

ANURAJ AGARWAL: 19BCE2026

INDEX

ACKNOWLEDGEMENT

Primarily I would like to thank god for being able to complete this project with success. Then I would like to thank my Cyber Security teacher Prof. Joshva Devadas, whose valuable guidance has been the ones that helped our group patch this project and make it full proof success. His suggestions and instructions has served as the major contributor towards the completion of the project.

ABSTRACT

Recent studies indicate that many users have difficulties managing online passwords for the increasing number of accumulated accounts. As a result, users often adopt strategies to simplify password management, such as selecting weak passwords and reusing passwords across multiple accounts, which unfortunately can cause security vulnerabilities. In user studies on password habits it was found that while users accumulate more and more accounts as a function of the time, they stay active online, the number of different passwords they maintain for accessing these accounts remains more or less constant. With the help of our application we are able to provide a strong password and also able to create a unique database for a user to use it in multiple platforms.

Caesar cipher is an ancient, elementary method of encrypting plain text message into cipher text protecting it from adversaries. However, with the advent of powerful computers, there is a need for increasing the complexity of such techniques. This paper contributes in the area of classical cryptography by providing a modified and expanded version for Caesar cipher using knowledge of mathematics and computer science. By our application we are providing the basic methodology of how all the types of Caesar cipher works by encrypting the user's personal information.

PROBLEM STATEMENT

In the latest decade, there is a growth in internet services which allows us to communicate interact and collaborate with others all over the world. This led users to have abundant personal information on various sites all over the internet. This calls for security, the recent stats show that this theft is caused mostly due to password hack. Our project deals with this problem only generating and managing strong passwords for the users so that they don’t have to worry about it again.

RELATED WORK

Sr. No.	Author	Title	Year
1.	N. Kshetri	The simple economics of cybercrimes	2006
2.	M. Bidgoli and J. Grossklags	End user cybercrime reporting: what we know and what we can do to improve it	2016
3.	S. Noimanee, K. Noimanee, S. Krisanachinda and W. Senavongse	Study of cybercrime and security in medical devices	2016
4.	A. Riaz and A. Riaz	Causes and consequences of cybercrimes: An exploratory study of Pakistan	2015
5.	M. M. Taha, T. A. Alhaj, A. E. Moktar, A. H. Salim and S. M. Abdullah	On password strength measurements: Password entropy and password quality	2013
6.	S. Zhang, J. Zeng and Z. Zhang	Password guessing time based on guessing entropy and long-tailed password distribution in the large-scale password dataset	2017
7.	P. Tsokkis and E. Stavrou	A password generator tool to increase users' awareness on bad password construction strategies	2018
8.	Deepanshu, M. Sharma, S. Som and S. K. Khatri	Enhancing password security using cyclic group matrix	2017
9.	T. Sugai, T. Ohigashi, Y. Kakizaki and A. Kanaoka	Password Strength Measurement without Password Disclosure	2019
10.	C. Luevanos, J. Elizarraras, K. Hirschi and J. Yeh	Analysis on the Security and Use of Password Managers	2017

After reviewing all the above research papers, we were able to create a systematic algorithm which not only generate strong passwords but also helps to manage them and customize them in future.

We also done a thorough study on different types of Caesar ciphers and implemented them in our application.

PROPOSED WORK

- Development and testing of all cipher algorithms.
- Development of password generation algorithm using randomized hashing in python.
- Creating the database for password management portal
- Implementing the developed algorithms in JavaScript and then linking them with backend.
- Implementation of user authentication on the Password Management Portal.
- Implementation of all the different types of Caesar ciphers to generate cipher text.

METHODOLOGY

ALGORITHM FOR PASSWORD GENERATION

- The password generation algorithm uses random library of Python to randomize the generated list of all characters on the standard US keyboard.
- The algorithm works as follows:
- Say length =N
- For every iteration from [1, N]:
- The whole list is randomized at every step and then again, a character is randomly picked from the list. At last again the password list is randomized.
- At last the generated list after conversion is stored into a database with a master password and is also given to the user.
- At the end there are in total 1+(N) +1 => (N+2) rounds of randomization which might end up in (N+2)! * (94!) rounds in case of a brute force attack.
- 94 is the total number of characters on the keyboard.
- The specialty of the algorithm is that it can generate a large number of passwords with any repetitions.

ALGORITHM FOR 1D SUBSTITUTION CIPHER

Input:

- A String of both lower- and upper-case letters, called Plaintext.
- An Integer denoting the required key.

Procedure:

- Create a list of all the characters.
- Create a dictionary to store the substitution for all characters.
- For each character, transform the given character as per the rule, depending on whether we're encrypting or decrypting the text.
- Print the new string generated.

ALGORITHM FOR POLYGRAPHIC SUBSTITUTION CIPHER

ENCRYPTION

- Each letter is first encoded as a number. The most common scheme used to be:
- Then, the message that is to be encrypted will be held in a block of n letters considered as a vector of n dimensions. It will then be multiplied by an n x n matrix known as the key matrix.
- Then the result will be converted with modulo 26

DECRYPTION

- we hold the cipher text in a vector of n dimensions, like we did with the plain text.
- Then we multiply by the inverse of the key matrix. This in reality is not a regular inverse matrix.
- It is heavily dependent in the modulo being used. Then we will convert the resultant matrix with modulo 26.

ALGORITHM FOR HYBRID CIPHER

ENCRYPTION

- Assign synthetic value for message.
- Multiply synthetic value with random selected natural number.
- Calculate with modulo 37.
- Again, select random negative number and multiply with it.
- Again, calculate with modulo 37 $CT = (PT \times n \times n1) \bmod 1$.

DECRYPTION

- Multiply received text with key 1 & key 2.
- Calculate with modulo 37.
- Remainder is Revealed Text or Plain Text $PT = (CT \times n{-}1 \times n1{-}1) \bmod 1$.

ALGORITHM FOR SHIFT CIPHER

ENCRYPTION

- Convert the letter into the number that matches its order in the alphabet starting from 0, and call this number X.
- Calculate: $Y = (X + K) \bmod 26$
- Convert the number Y into a letter that matches its order in the alphabet starting from 0.

DECRYPTION

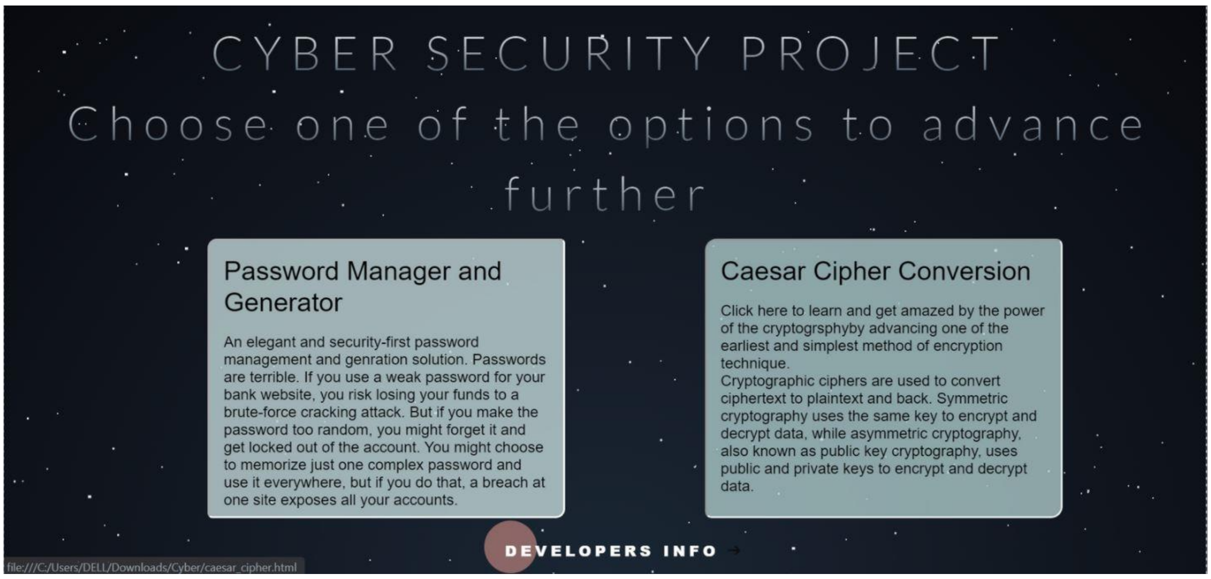
For every letter in the cipher text C :

- Convert the letter into the number that matches its order in the alphabet starting from 0, and call this number Y.
- Calculate: $X= (Y - K) \bmod 26$
- Convert the number X into a letter that matches its order in the alphabet starting from 0.

RESULT AND ANALYSIS

The snapshots of the completed work are:

1.Landing Page



CYBER SECURITY PROJECT

Choose one of the options to advance further

Password Manager and Generator

An elegant and security-first password management and genration solution. Passwords are terrible. If you use a weak password for your bank website, you risk losing your funds to a brute-force cracking attack. But if you make the password too random, you might forget it and get locked out of the account. You might choose to memorize just one complex password and use it everywhere, but if you do that, a breach at one site exposes all your accounts.

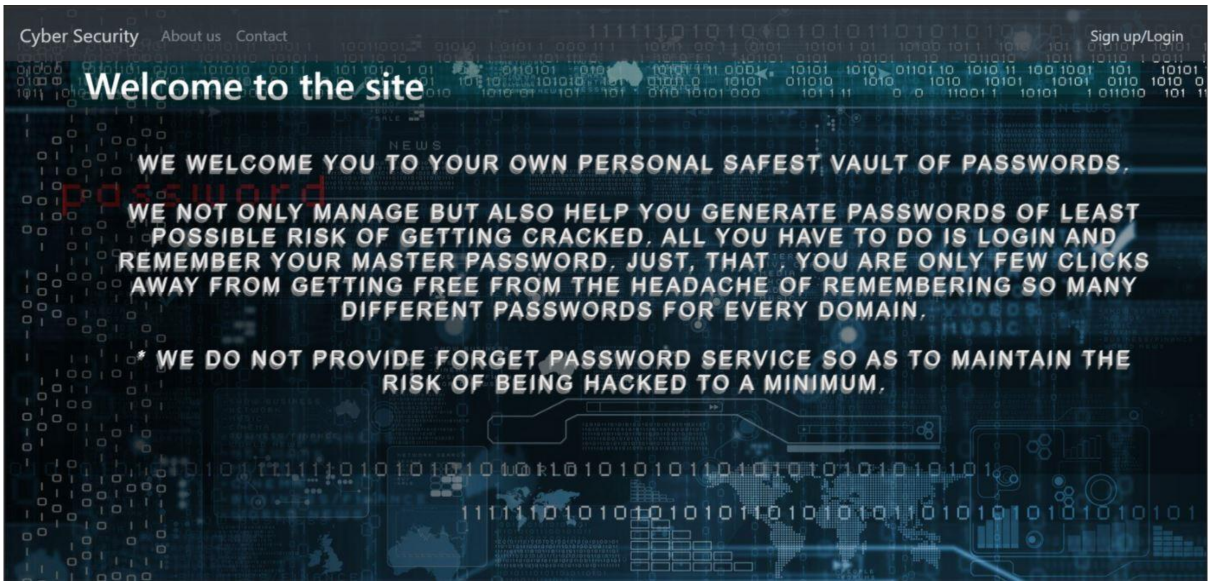
Caesar Cipher Conversion

Click here to learn and get amazed by the power of the cryptogrsphyby advancing one of the earliest and simplest method of encryption technique.
Cryptographic ciphers are used to convert ciphertext to plaintext and back. Symmetric cryptography uses the same key to encrypt and decrypt data, while asymmetric cryptography, also known as public key cryptography, uses public and private keys to encrypt and decrypt data.

DEVELOPERS INFO →

file:///C:/Users/DELL/Downloads/Cyber/caesar_cipher.html

2. Password Management Landing Page



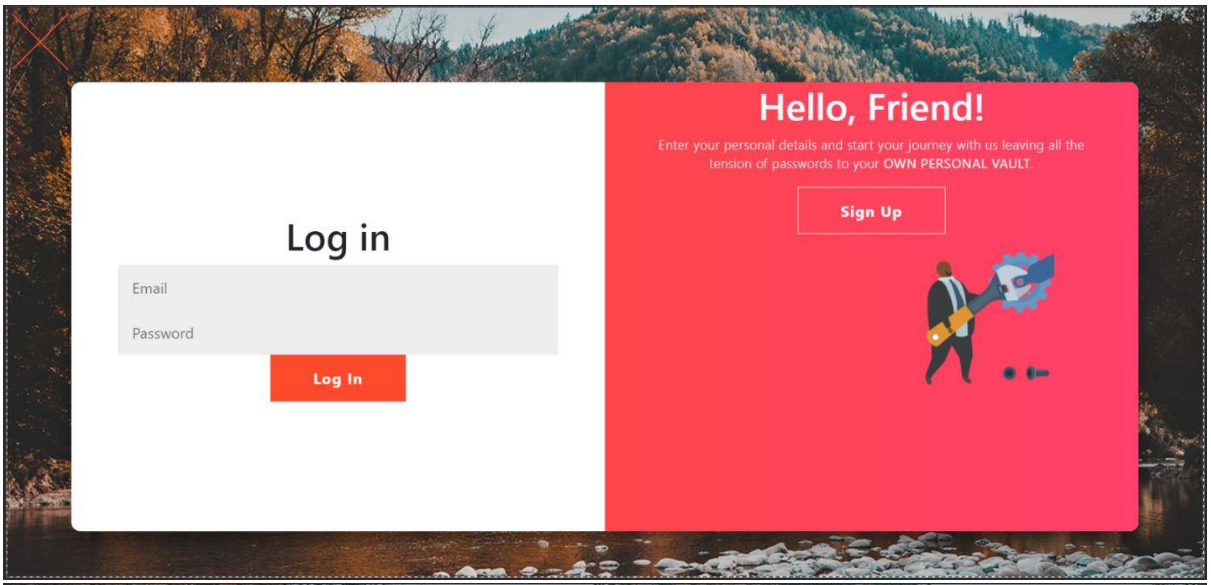
Welcome to the site

WE WELCOME YOU TO YOUR OWN PERSONAL SAFEST VAULT OF PASSWORDS.

WE NOT ONLY MANAGE BUT ALSO HELP YOU GENERATE PASSWORDS OF LEAST POSSIBLE RISK OF GETTING CRACKED. ALL YOU HAVE TO DO IS LOGIN AND REMEMBER YOUR MASTER PASSWORD. JUST, THAT . YOU ARE ONLY FEW CLICKS AWAY FROM GETTING FREE FROM THE HEADACHE OF REMEMBERING SO MANY DIFFERENT PASSWORDS FOR EVERY DOMAIN.

* WE DO NOT PROVIDE FORGET PASSWORD SERVICE SO AS TO MAINTAIN THE RISK OF BEING HACKED TO A MINIMUM.

3. Login and Signup Page



Log in

Email

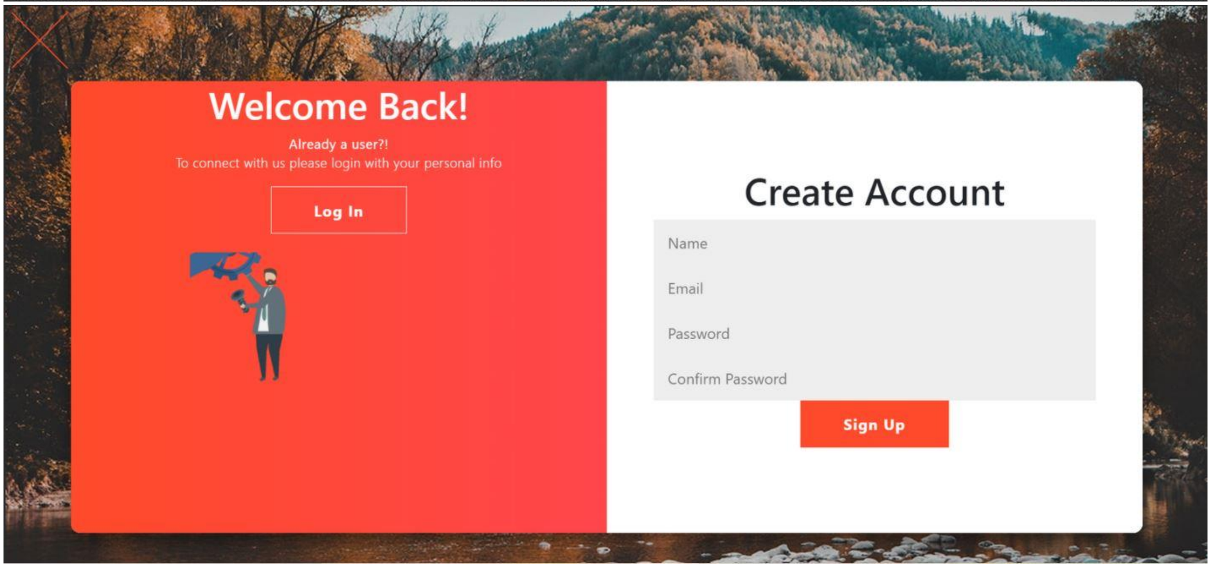
Password

Log In

Hello, Friend!

Enter your personal details and start your journey with us leaving all the tension of passwords to your OWN PERSONAL VAULT.

Sign Up



Welcome Back!

Already a user?!

To connect with us please login with your personal info

Log In

Create Account

Name

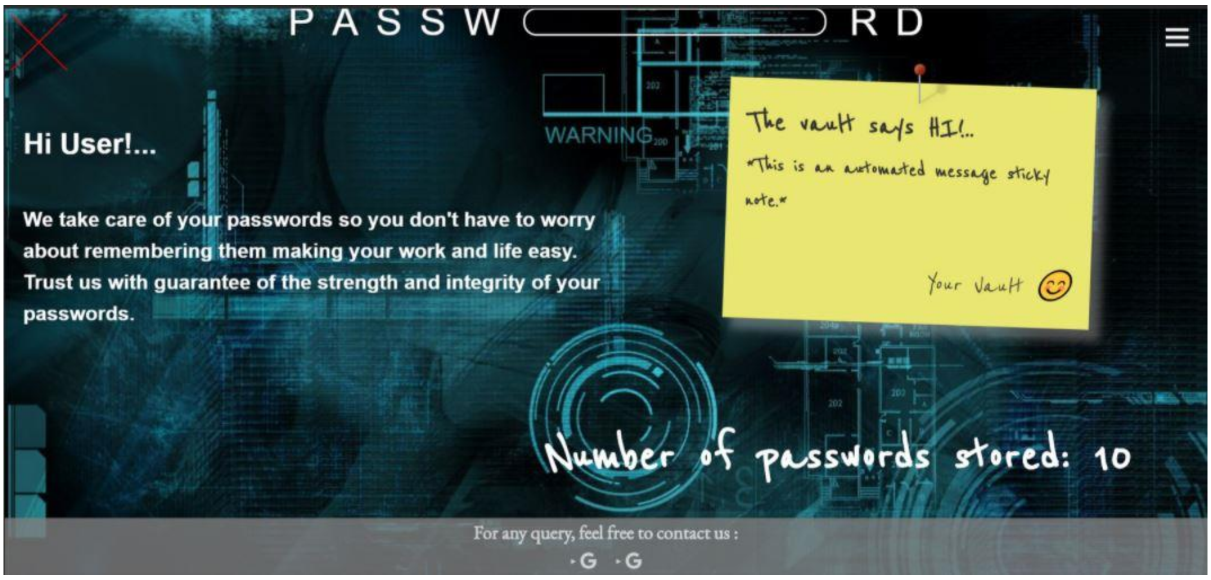
Email

Password

Confirm Password

Sign Up

4.Post login page of Password management Portal.



Hi User!...

We take care of your passwords so you don't have to worry about remembering them making your work and life easy. Trust us with guarantee of the strength and integrity of your passwords.

The vault says HI!..

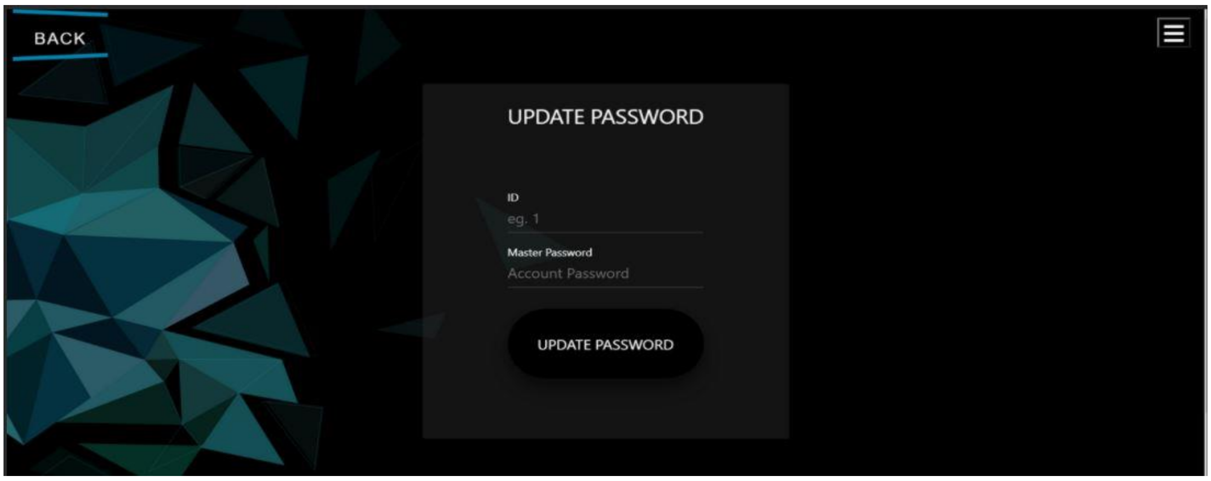
This is an automated message sticky note.

Your Vault

Number of passwords stored: 10

For any query, feel free to contact us :

5.Add/Delete/Update/View Portals



UPDATE PASSWORD

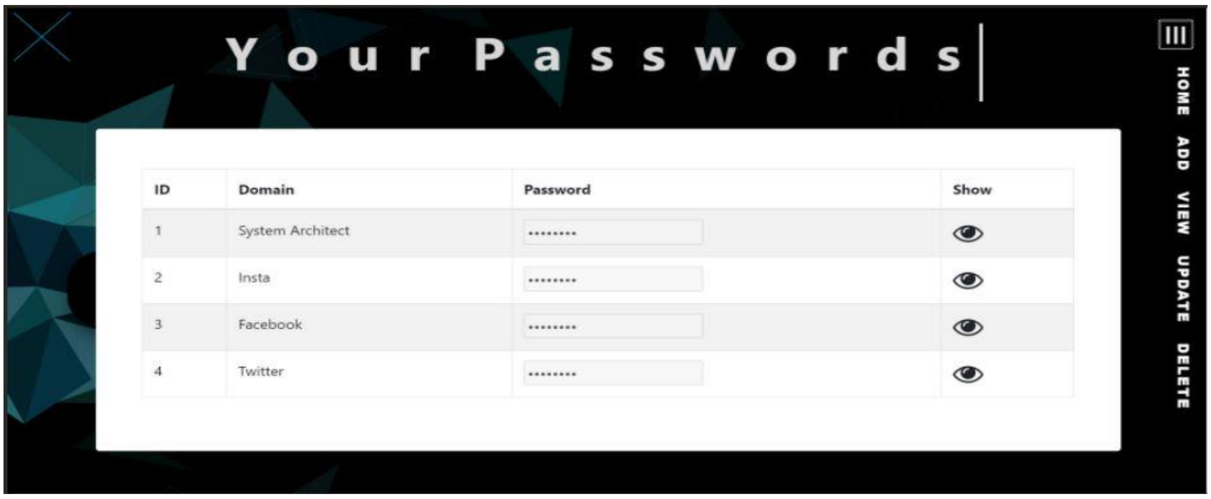
ID

eg. 1

Master Password

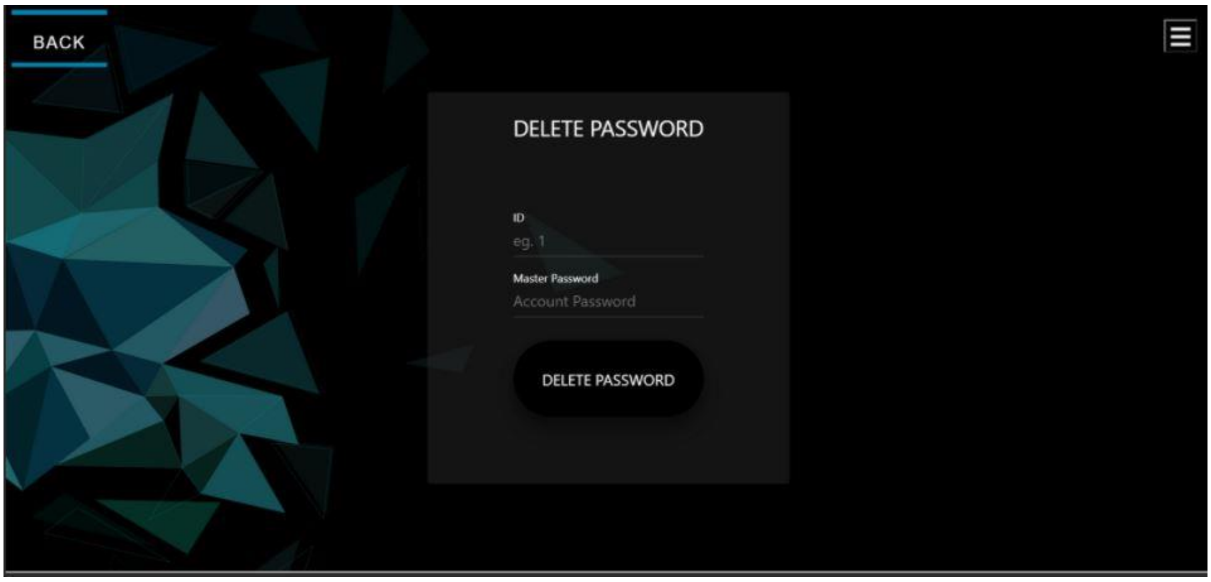
Account Password

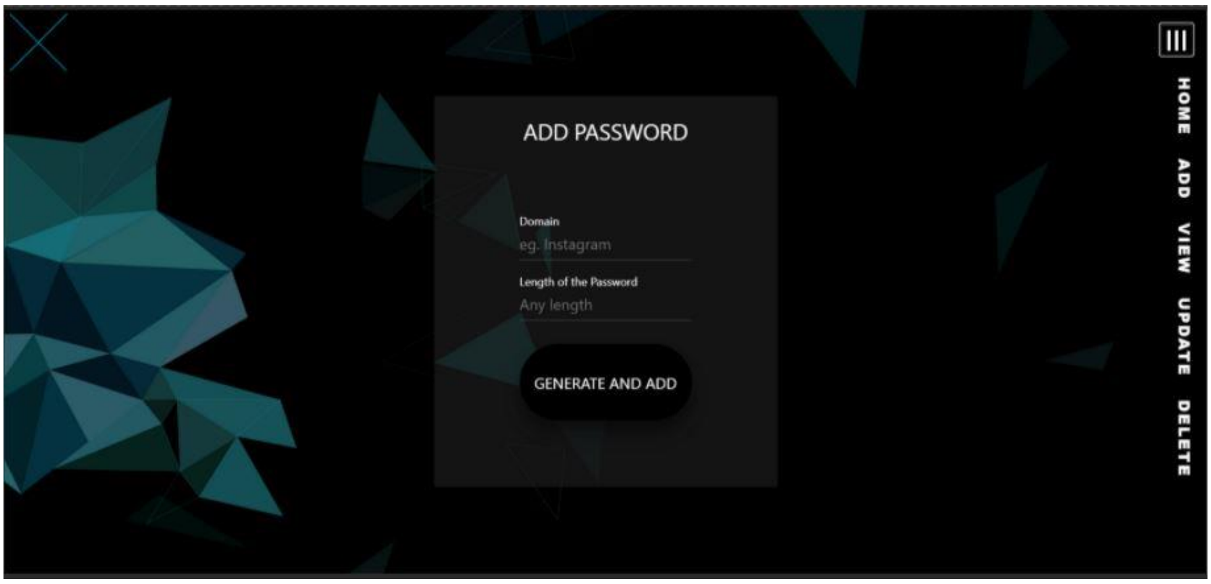
UPDATE PASSWORD



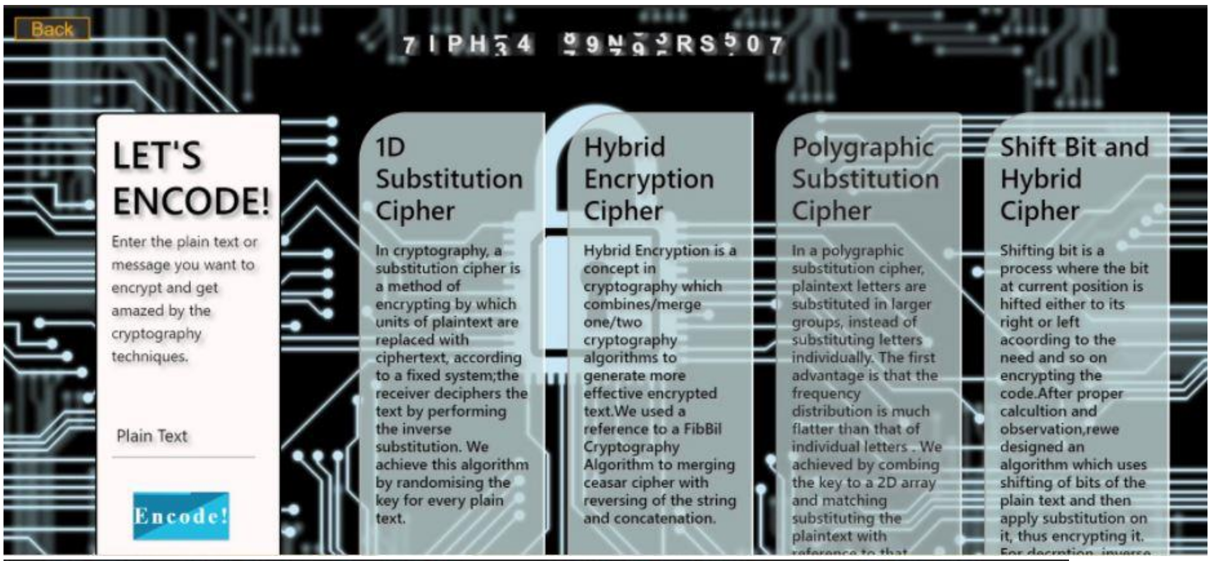
Your Passwords

ID	Domain	Password	Show
1	System Architect	••••••••	
2	Insta	••••••••	
3	Facebook	••••••••	
4	Twitter	••••••••	


BACK
DELETE PASSWORD
ID
eg. 1
Master Password
Account Password
DELETE PASSWORD


ADD PASSWORD
Domain
eg. Instagram
Length of the Password
Any length
GENERATE AND ADD
HOME
ADD
VIEW
UPDATE
DELETE

6. Cipher Generation Page


Back

7 I P H 3 4 9 N 9 R S 5 0 7

LET'S ENCODE!

Enter the plain text or message you want to encrypt and get amazed by the cryptography techniques.

Plain Text

Encode!

1D Substitution Cipher

In cryptography, a substitution cipher is a method of encrypting by which units of plaintext are replaced with ciphertext, according to a fixed system;the receiver deciphers the text by performing the inverse substitution. We achieve this algorithm by randomising the key for every plain text.

Hybrid Encryption Cipher

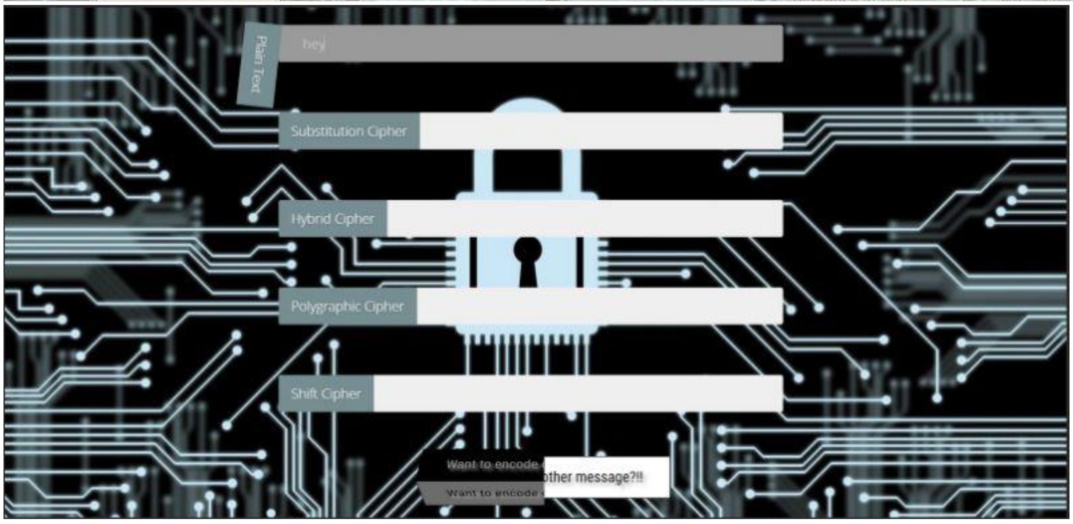
Hybrid Encryption is a concept in cryptography which combines/merge one/two cryptography algorithms to generate more effective encrypted text.We used a reference to a FibBil Cryptography Algorithm to merging ceasar cipher with reversing of the string and concatenation.

Polygraphic Substitution Cipher

In a polygraphic substitution cipher, plaintext letters are substituted in larger groups, instead of substituting letters individually. The first advantage is that the frequency distribution is much flatter than that of individual letters . We achieved by combing the key to a 2D array and matching substituting the plaintext with reference to that

Shift Bit and Hybrid Cipher

Shifting bit is a process where the bit at current position is hifted either to its right or left acoording to the need and so on encrypting the code.After proper calcultion and observation,rewe designed an algorithm which uses shifting of bits of the plain text and then apply substitution on it, thus encrypting it. For decrption, inverse


Plain Text

hey

Substitution Cipher

Hybrid Cipher

Polygraphic Cipher

Shift Cipher

Want to encode another message?!!

A Snippet of Backend Code which is running the whole website:

```
var express =require("express");
var app=express();
var bodyparser=require("body-parser");
app.use(express.static("cssresources"));
app.use(bodyparser.urlencoded({extended:true}));
app.set("view engine","ejs");
// ==>>>> basic setup ends here

// ==========================>
// DATABASE SETUP
// ==========================>
const mongoose=require("mongoose");
const { stringify } = require("querystring");
const { request, response } = require("express");
mongoose.connect('mongodb://localhost/demo_cyber', {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => console.log('Connected to DB!'))
.catch(error => console.log(error.message));
mongoose.set('useFindAndModify', false);
// ==================> Designing user schemas
var userauthschema=new mongoose.Schema({
   Name: String,
   Userid: String,
   Password: String,
   master_key: Number
});
var userdataschema=new mongoose.Schema({
   master_key: Number,
   details :[{entity: String,password: String}]
```

```
});
var Authuser=mongoose.model("Authuser",userauthschema);
var User=mongoose.model("User",userdataschema);
// ==================>
// =================>
// Paths::::
// =================>

// ==============================>
// PATHS FOR FIRST LANDING PAGE
// ==============================>


//path for landing page
app.get("/",function(request,response)
{
  response.render("landing");
});
// path for developer info
app.get("/dev",function(request,response){
  response.render("developers");
});

// ==============================>
// PATHS FOR CIPHER
// ==============================>

//path for caeser cipher landing page
app.get("/cipherintro",function(request,response){
  response.render("cipherlanding");
});
//path for caeser cipher result
app.get("/encrypt",function(request,response){
  var tobeenc=request.query.asked;
  response.render("encryptlanding",{data:tobeenc});
});

// ==========================================
==>
// PATHS FOR PASSWORD MANAGEMENT PORTAL
// ==========================================
==>


//path for password management landing
app.get("/passwordlanding",function(request,response){
  response.render("passlanding");
})

//path for contact page in password landing page
app.get("/passwordlanding/contact",function(request,response){
  response.render("contacts");
});

//path for login and sign up page in password landing page
app.get("/passwordlanding/loginsignup",function(request,response){
  var tt="";
  response.render("loginpage",{tt});
});

app.post("/register",function(request,response){
  // console.log(request.body);
  var secretkey=Math.floor((Math.random() * 1000000) + 1);
  var userauthdata={
    Name: request.body.username,
    Userid: request.body.userid,
    Password: request.body.userpass,
    master_key: secretkey
  };
  var tt="";
  Authuser.find({Userid: request.body.userid},function(err,ret){
    if (err){
      console.log(err);
    }
    else{
      if (ret.length>0){
        tt="User already exists!!.\n PLEASE LOGIN OR SIGN UP WITH A DI
FFERENT EMAIL ID!!";
        response.render("loginpage",{tt});
      }
      else{
        Authuser.create(userauthdata,function(err,user){
          if (err){
            console.log(err);
          }
          else{
            var newuser={
              master_key: secretkey
            };
            User.create(newuser,function(err,retout){
              if (err){
                console.log(err);
              }
              else{
                console.log(retout);
              }
            });
            tt="Congratulations!!, You are registered.\n Your secret key is :"
+userauthdata.master_key+".\n Please keep it safe and secure!!"
            console.log("Successfully registered!!!!");
            // console.log(user);
            response.render("loginpage",{tt});
          }
        });
      }
    }
  });
});
app.post("/validatedata",function(request,response){
  var provideddata={
    Userid: request.body.userid,
    Password: request.body.userpass
  };
  Authuser.find(provideddata,function(err,sendback){
    if (err){
      console.log(err);
    }
    else{
      if (sendback.length==0){
        var tt="Wrong Details!!.\nPlease signup first if you are not register
ed or try again!!";
        response.render("loginpage",{tt});
      }
      else{
        var tt="";
        nameuser=sendback[0].Name;
        var sk=sendback[0].master_key;
        User.find({master_key:sk},function(err,data){
          if (err){
            console.log(err);
          }
          else{
            totpass=data[0].details.length;
            // console.log(totpass);
            response.render("pwd-
welcome",{tt,nameofuser:nameuser,recordsofuser:totpass});
```

```
          }
        });
      }
    }
  });
})
//====================================>>>>>>>>
>>>>>>>>>
// Landing page for post login page
app.get("/home",function(request,response){
  var tt="";
  response.render("pwd-
welcome",{tt,nameofuser:nameuser,recordsofuser:totpass});
});

//Page for adding a password
app.get("/add",function(request,response){
  response.render("pwd-add");
});
//Path for the views page after addition
app.post("/add&view",function(request,response){
  // console.log(request.body);
  var newentry={
    entity: request.body.Domain,
    password: generate(request.body.len)
  }
  var providedkey=(parseInt(request.body.masterkey));
  // console.log(providedkey);
  Authuser.find({master_key:providedkey},function(err,retdata){
    if (err){
      console.log(err);
    }
    else{
      if (retdata.length==0){
```

```
    var tt="Provided Master key is wrong!!";
    response.render("pwd-
welcome",{tt,nameofuser:nameuser,recordsofuser:totpass});
   }
   else{
    User.find({"details.entity": request.body.Domain,master_key:provi
dedkey},function(err,res){
     if (err){
      console.log(err);
     }
     else{
      if (res.length==0){
       User.findOneAndUpdate({master_key:providedkey},{$push:{d
etails:newentry}},function(err,user){
        if (err){
         console.log(err);
        }
        else{
         // console.log(user);
         totpass+=1;
         response.render("pwd-view2");
        }
       });
      }
      else{
       var tt="Password for this domain already exists.\nIf you wish
to update,click the update link in the side menu.";
       response.render("pwd-
welcome",{tt,nameofuser:nameuser,recordsofuser:totpass});
      }
     }
    });
   }
  }
  });
});
//Page for view all passwords (stored)
app.get("/view",function(request,response){
  response.render("pwd-view2");
});
app.post("/getinfo",function(request,response){
  var dataprovided={
    master_key:request.body.masterkey,
    Password: request.body.masterpassword
  };
  Authuser.find(dataprovided,function(err,retauth){
    if (err){
      console.log(err);
    }
    else{
      if (retauth.length==0){
        var tt="Invalid Credentials";
        response.render("pwd-
welcome",{tt,nameofuser:nameuser,recordsofuser:totpass});
      }
      else{
        User.find({master_key:request.body.masterkey},function(err,data)
{
          if (err){
            console.log(err);
          }
          else{
            if (data[0].details.length==0){
              var tt="No records found!!";
              response.render("pwd-
welcome",{tt,nameofuser:nameuser,recordsofuser:totpass});
            }
            else{
              var userdetailsdata=data[0].details;
              // console.log(userdetailsdata);
              response.render("pwd-view",{displaydata:userdetailsdata});
            }
          }
        });
      }
    }
  });
});
//Path for updating a given password
app.get("/update",function(request,response){
  response.render("pwd-update");
});
//Path for the views page after updation
app.post("/upd&view",function(request,response){
  var authinfo={
    Password: request.body.pass,
    master_key: request.body.masterkey
  };
  Authuser.find(authinfo,function(err,retauth){
    if (err){
      console.log(err);
    }
    else{
      if (retauth.length==0){
        var tt="Invalid Credentials Provided!!!"
        response.render("loginpage",{tt});
      }
      else{
        User.find({"details.entity": request.body.domain,master_key:reque
st.body.masterkey},function(err,resd){
          if (err){
            console.log(err);
          }
          else{
            if (resd.length==0){
              var tt="Please generate a password for the provided domain b
efore trying to update it."
              response.render("pwd-
welcome",{tt,nameofuser:nameuser,recordsofuser:totpass});
            }
            else{
              User.findOneAndUpdate({master_key:request.body.masterkey
,"details.entity": request.body.domain},{$set :{"details.$[].password":gene
rate(request.body.passlen)}},function(err,user){
                if (err){
                  console.log(err);
                }
                else{
                  response.render("pwd-view2");
                }
              });
            }
          }
        })
      }
    }
  });
});

//Path for deleting a password
app.get("/delete",function(request,response){
  response.render("pwd-delete");
});
//Path for the views page after deletion
app.post("/delete&view",function(request,response){
  var enteredcred={
    Password: request.body.pwd,
    master_key: request.body.masterkey
  }
  Authuser.find(enteredcred,function(err,res){
    if (err){
      console.log(err);
    }
    else{
      if (res.length==0){
        var tt="Invalid Credentials Provided!!!"
        response.render("loginpage",{tt});
      }
      else{
        User.find({"details.entity": request.body.domain,master_key:reque
st.body.masterkey},function(err,resd){
          if (err){
            console.log(err);
          }
          else{
            if (resd.length==0){
              var tt="Please generate a password for the provided domain b
efore trying to delele it."
              response.render("pwd-
welcome",{tt,nameofuser:nameuser,recordsofuser:totpass});
            }
            else{
              User.updateMany({master_key:request.body.masterkey},{$pul
l :{details:{entity:request.body.domain}}},function(err,user){
                if (err){
                  console.log(err);
                }
                else{
                  totpass-=1;
                  response.render("pwd-view2");
                }
              });
            }
          }
        })
      }
    }
  });
});
//==============================================
==============
//Residual path
app.get("/passwordlanding/loginsignup/login",function(request,respons
e){
  response.render("pwd-welcome");

});
app.get("/passwordlanding/loginsignup/login/generated",function(requ
est,response){
  var reqlen=parseInt(request.query.len);
  response.render("postloginlandingans",{data:reqlen});
});
//JUST A TESTING PATH FOR ALL JS CODES
app.get("/testing",function(request,response){
  response.render("testerforalgos");
})

//Residual paths end here
//==============================================
==============




//command for starting the server
```

```
app.listen(process.env.PORT || 3000,process.env.IP,function()
{
  console.log("Server started!!!");
});
```

Code for Password Generation:

```
function generate(size){
  function random_item(list){
    return list[Math.floor(Math.random()*list.length)];
  }
  function shuffleArray(arra1) {
    var ctr = arra1.length, temp, index;
    while (ctr > 0) {
       index = Math.floor(Math.random() * ctr);
       ctr--;
       temp = arra1[ctr];
       arra1[ctr] = arra1[index];
       arra1[index] = temp;
    }
    return arra1;
  }
  var items=['1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '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', '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', '~', '`', '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '_', '+', '=', '-
', '\\', ']', '[', '|', '}', '"', "'", ':', ';', '?', '/', '>', '.', '<', ',' , '{'];
  //for time being i am taking the length of password to be 8
  size=parseInt(size);
  //no of passwords=5;
  var counter=1;
  var allpasswords=[];
  items=shuffleArray(items);
  var listshuffled=shuffleArray(items);
  var pass="";
  for(var i=0;i<size;i++){
    pass+=(random_item(listshuffled));
  }
  var finalpass=shuffleArray(pass);
  for(var i=0;i<size;i++){
    finalpass=shuffleArray(finalpass);
  }
  return finalpass;
}
```

3.Codes for the cipher Algorithms in C++

1-D substitution cipher:

```
var alpha=[4,7,13,10,9,23,0,8,22,18,17,19,21,16,6,12,5,3,25,24,15,1,20,11,14,2];
var nums=[9,4,3,0,7,6,1,8,2,5];
var message="Pavitra";
var cipheredtext="";
for(var i=0;i<message.length;i++){
  //next line of code for getting ascii value of a character;
  var charcode=message[i].charCodeAt(0);
  if (charcode>=65 && charcode<=90){
    var pos=charcode-65;
    var newc=String.fromCharCode(65+alpha[pos]);
    cipheredtext+=newc;
  }
  else if (charcode>=97 && charcode<=122)
  {
    var pos=charcode-97;
    var newc=String.fromCharCode(97+alpha[pos]);
    cipheredtext+=newc;
  }
  else if (charcode>=48 && charcode<=57)
  {
    var pos=charcode-48;
    var newc=String.fromCharCode(48+nums[pos]);
    cipheredtext+=nch;
  }
  else if (charcode==32)
  {
    cipheredtext+='#';
  }
}
console.log(cipheredtext);
```

Parity Encryption Cipher:

```
function modulo(a,b){
  var answer=(b+(a%b))%b;
  return answer;
}
function modsub(a,b,m){
  return modulo(modulo((modulo(a,m)-modulo(b,m)),m),m);
}
function getchar(coor){
  if (coor==62){
    var ret='_';
    return ret;
  }
  if (coor>=0 && coor<26){
    var ret=String.fromCharCode(65+coor);
    return ret;
  }
  if (coor>=26 && coor<52){
    var ret=String.fromCharCode(97+(coor-26));
    return ret;
  }
  if (coor>=52 && coor<62){
    var ret=String.fromCharCode(48+(coor-52));
    return ret;
  }
  return '!';
}
function pos(ch){
  var charcode=ch[0].charCodeAt(0);
  // console.log(charcode);
  if (ch=='_'){
    return 62;
  }
  if (charcode>=97){
    var ret=26+(charcode-97);
    return ret;
  }
  if (charcode>=48 && charcode<=57){
    var ret=52+(charcode-48);
    return ret;
  }
  return (charcode-65);
}
var data=[37,19,11,3,16,17,62,18,36,1,31,5,15,25,10,54,4,6,40,45,30,41,14,47,9,28,58,43,8,56,34,24,2,7,49,12,20,35,55,26,39,51,0,61,13,46,23,60,50,27,44,42,29,32,48,57,52,22,38,33,53,59,21];
var shifter=153;
var input2="Hello_world";
var n=input2.length;
var even="",odd="";
for(var i=0;i<=n-1;i++){
  if ((i+1)%2==0){
    even+=input2[i];
  }
  if ((i+1)%2!=0){
    odd+=input2[i];
  }
}
var newodd="",neweven="";
for(var i=0;i<=odd.length-1;i++){
  var currpos=pos(odd[i]);
  var newpos=(currpos+shifter)%63;
  var newchar=getchar(data[newpos]);
  newodd+=newchar;
}
for(var i=0;i<=even.length-1;i++){
  var currpos=pos(even[i]);
  var newpos=modsub(currpos,shifter,63);
  var newchar=getchar(data[newpos]);
  neweven+=newchar;
}
var output2="";
var ko=0,ke=0;
for(var i=0;i<=n-1;i++){
  if ((i+1)%2==0){
    output2+=neweven[ke];
    ke+=1;
  }
  if ((i+1)%2!=0){
    output2+=newodd[ko];
    ko+=1;
  }
}
console.log(output2);
```

CONCLUSION

Our project is successful in dealing with the problem discussed about the remembering of multiple passwords and issue regarding the security of passwords. Our project manages multiple passwords for domains for multiple users with utmost security provided by the algorithm prepared. The project also consists of a Caesar cipher portal which ciphers the plain text to four different ciphers with the algorithms prepared by us discussed in the algorithm section.

Our future objective is to make the passwords generated more secure and use the cipher algorithms prepared in any application. We can do that by a concept named session and other related keywords.

FUTURE WORK

Security provided by this algorithm can be enhanced further by using it with one or more different stronger encryption algorithms which will further strong our password generation system to provide multiple passwords which will be much harder to decrypt.

REFERENCES

1. http://web.eecs.umich.edu/~jhalderm/pub/papers/password-www05.pdf
2. https://cloud.google.com/solutions/modern-password-security-for-system-designers.pdf
3. http://searchsecurity.techtarget.com/definition/cipher
4. http://searchsecurity.techtarget.com/definition/cryptanalysis
5. Atul Kahate (2009), Cryptography and Network Security, 2nd edition, McGraw-Hill.