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

PROJECT REVIEW 1

FALL SEMESTER 2020-21

CSE3501-INFORMATION SECURITY ANAYSIS AND AUDIT

TOPIC: HYBRID CRYTOGRAPHY

SLOT: G2

TEAM MEMBERS:

LOGA RAKSHIKA . B 18BIT0430

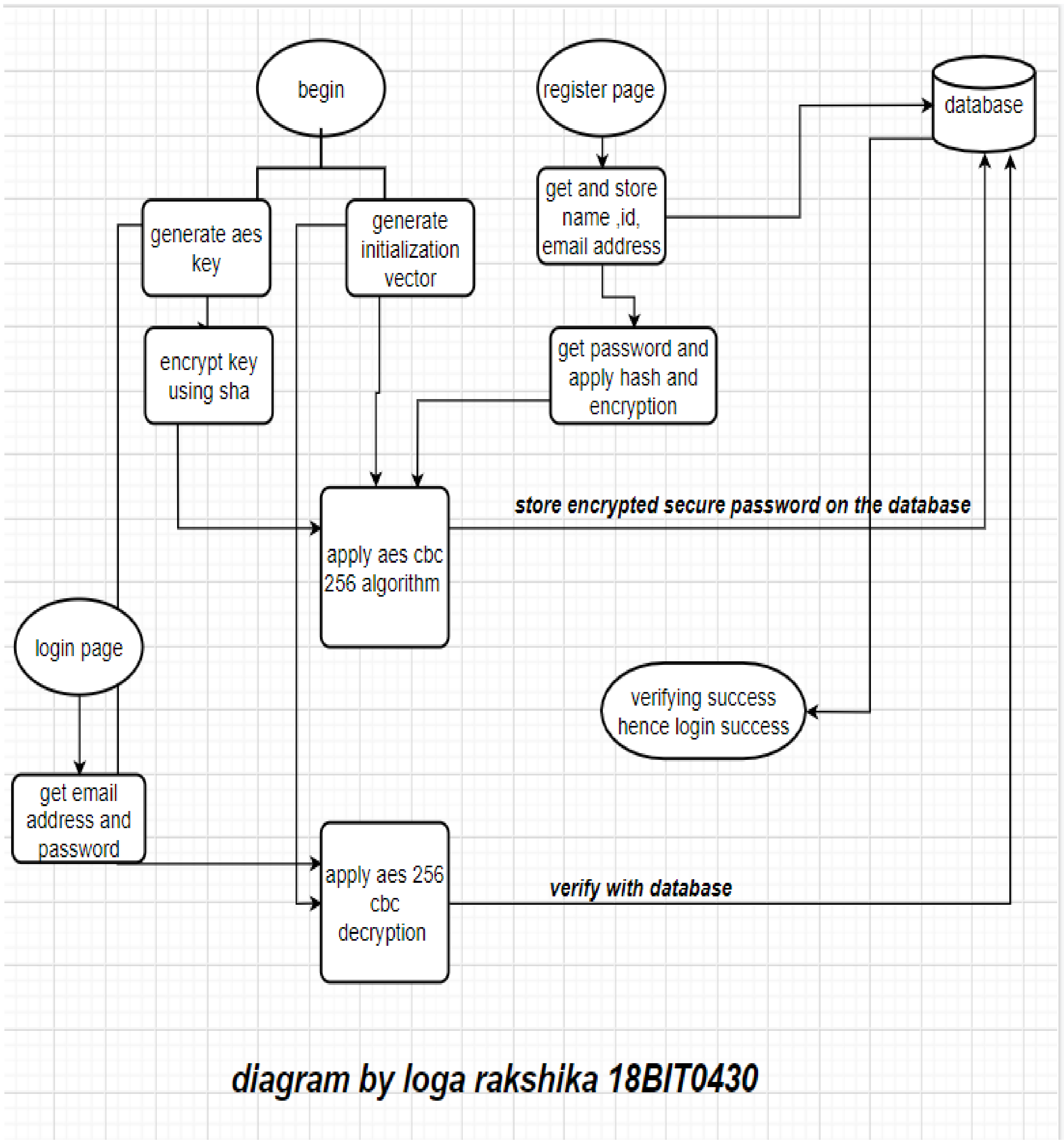
LYDIA CHRISLIN PAUL 18BIT0439

SARADHA DEVI T 18BIT0447

SUBMITTED TO : Dr. SUMAIYA THASEEN I

DONE BY: LOGA RAKSHIKA .B

DESIGN OF THE SYSTEM PROPOSED:


begin
register page
database
generate aes key
generate initialization vector
get and store name ,id, email address
encrypt key using sha
get password and apply hash and encryption
store encrypted secure password on the database
apply aes cbc 256 algorithm
login page
verifying success hence login success
get email address and password
apply aes 256 cbc decryption
verify with database
diagram by loga rakshika 18BIT0430

## DESCRIPTION OF THE SYSTEM:

The proposed system aims to provide secure storage of users password on the database. We implement this goal by applying the concept of hybrid cryptography. In this project I used SHA-256 and AES algorithm to increase the security of the password.

This system I have designed is a registration login system.users can register to the website and once registration is done , they can login. During the registration the user submits their details such as name , email address ,and their password.

On the server side, we get the data from the form elements and store it in the database. When we try to store password without any encryption or hashing , then it is very vulnerable and hackers can easliy login to users account. For this purpose , I incorporated AES and SHA256 to encrypt and hash a password. By doing this , password is safely stored inside the database. And user can use the website without any fear for vulnerability.also the database owner does not have any fear for the hackers.

The reason for choosing AES 256 algorithm is that , it generates 256 bits key and as the reasearch papers say , AES is resistent to brute force attacks than any other algorithm. Security experts maintain that AES is secure when implemented properly. However, AES encryption keys need to be protected. Even the most extensive cryptographic systems can be vulnerable if a hacker gains access to the encryption key.

AES is mainly used to protect data at rest . in this case its the password. As mentioned in above paragraph, AES keys are insecure and researches have identified side channel attacks because of unprotection to the key. So for this purpose , I used SHA256 to encrypt the key generated . this adds more security to the encryption process.

APPLICATION DEVELOPED:

The application we developed is user registration login system.

This consist of registration page, login page and dashborad page.

Registration page running on localhost/project/regrakhika.php

Login page running on localhost/project/login.php

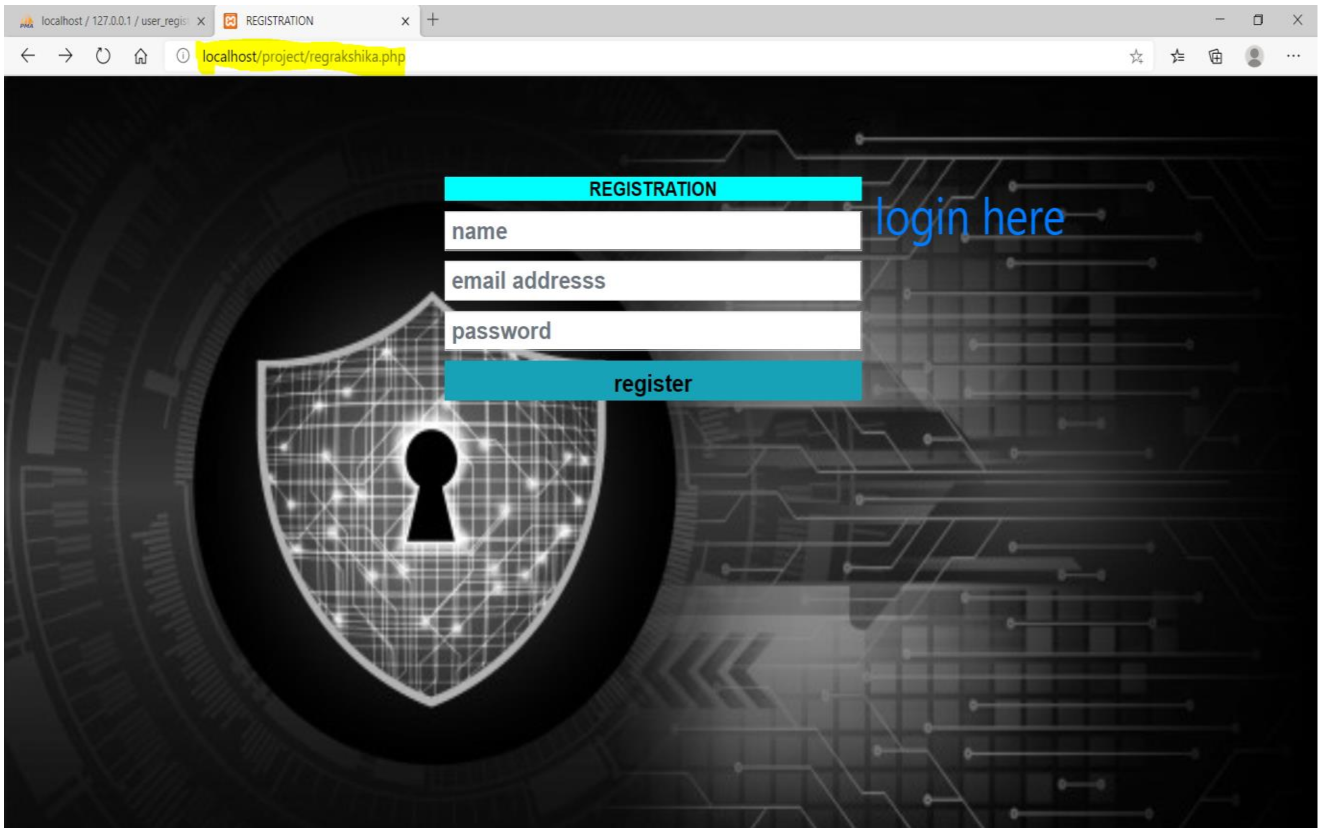
Dashboard page running on localhost/project/dashboard.html

Database name: user_register

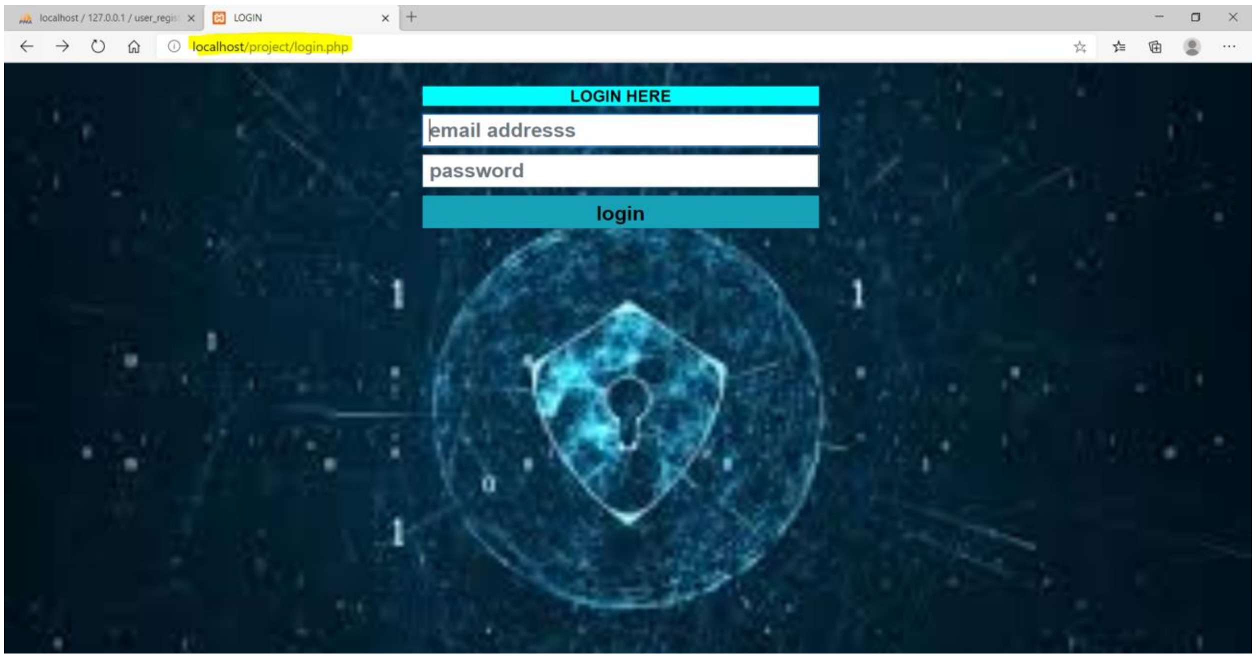
Table name:register

Table fields: id, name, email, password

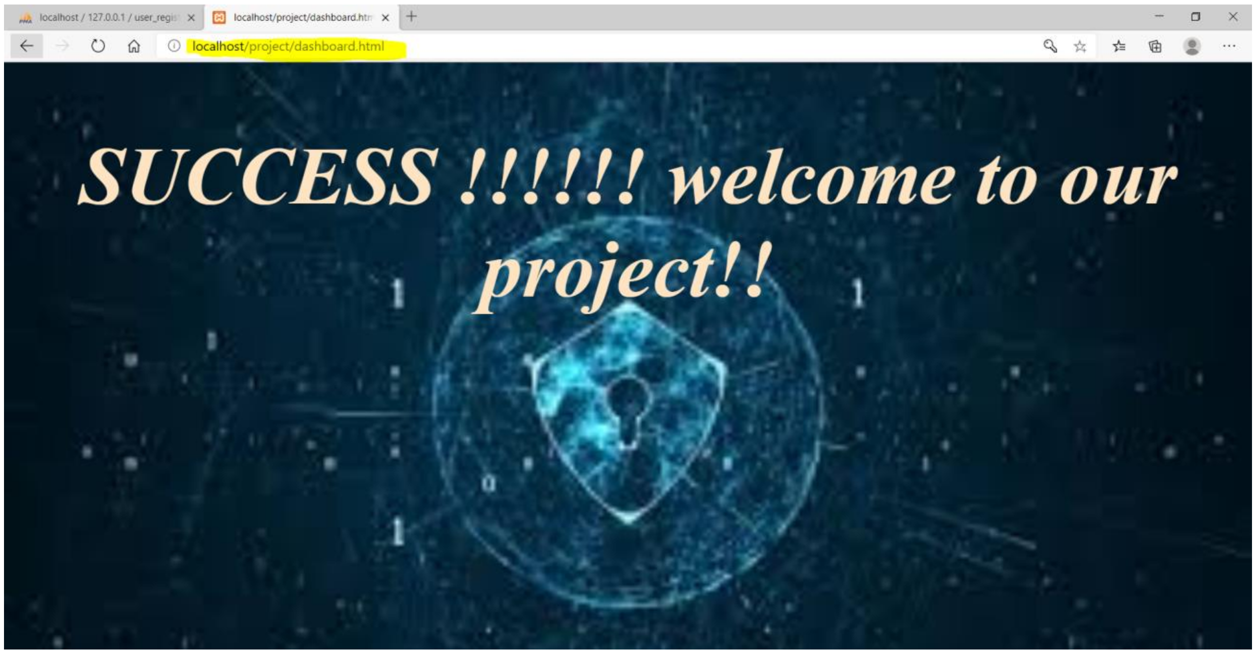
## REGISTRATION PAGE:



## LOGIN PAGE:

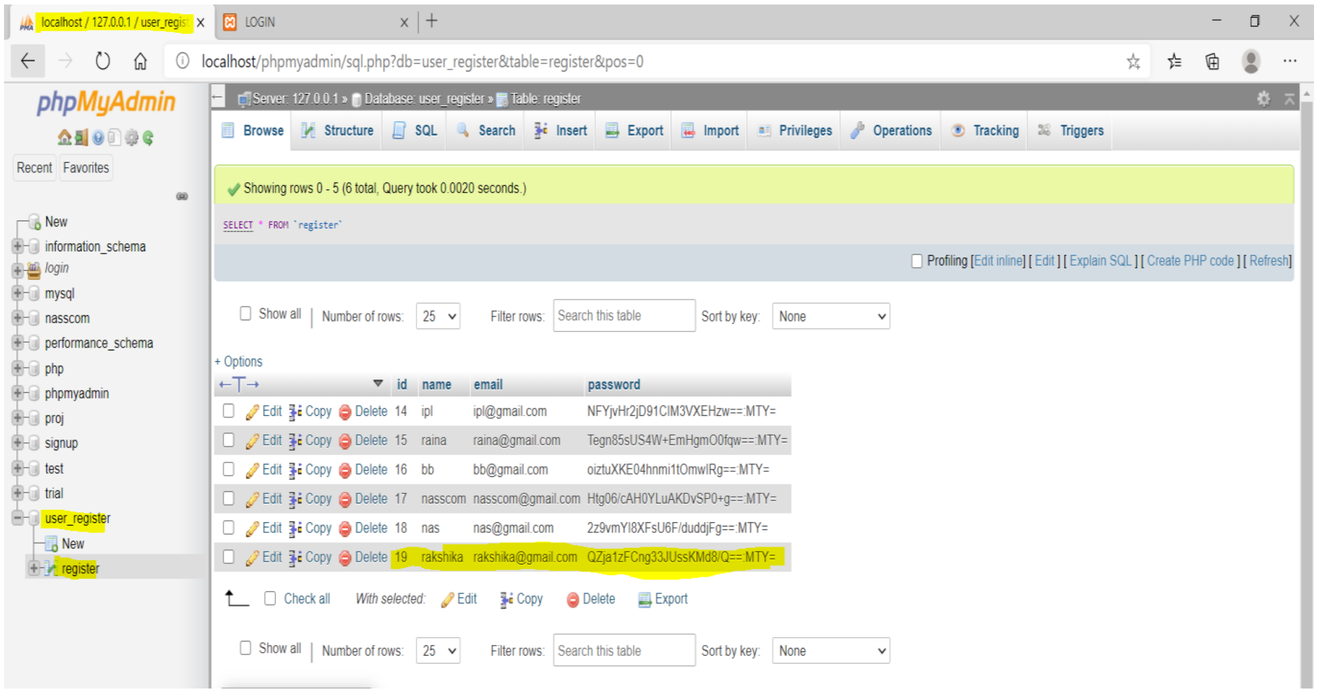

localhost / 127.0.0.1 / user_regis
LOGIN
localhost/project/login.php
LOGIN HERE
email addresss
password
login

DASHBOARD PAGE:


localhost / 127.0.0.1 / user_regis
localhost/project/dashboard.htm
localhost/project/dashboard.html

SUCCESS !!!!!! welcome to our project!!

DATABASE WITH ENCRYPTED PASSWORD:



Showing rows 0 - 5 (6 total, Query took 0.0020 seconds.)

```
SELECT * FROM `register`
```

Profiling [Edit inline] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

| id | name | email | password |
|---|---|---|---|
| 14 | ipl | ipl@gmail.com | NFYjvHr2jD91CIM3VXEHzw==:MTY= |
| 15 | raina | raina@gmail.com | Tegn85sUS4W+EmHgmO0fqw==:MTY= |
| 16 | bb | bb@gmail.com | oiztuXKE04hnmi1tOmwIRg==:MTY= |
| 17 | nasscom | nasscom@gmail.com | Htg06/cAH0YLuAKDvSP0+g==:MTY= |
| 18 | nas | nas@gmail.com | 2z9vmYI8XFsU6F/duddjFg==:MTY= |
| 19 | rakshika | rakshika@gmail.com | QZja1zFCng33JUssKMd8/Q==:MTY= |

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

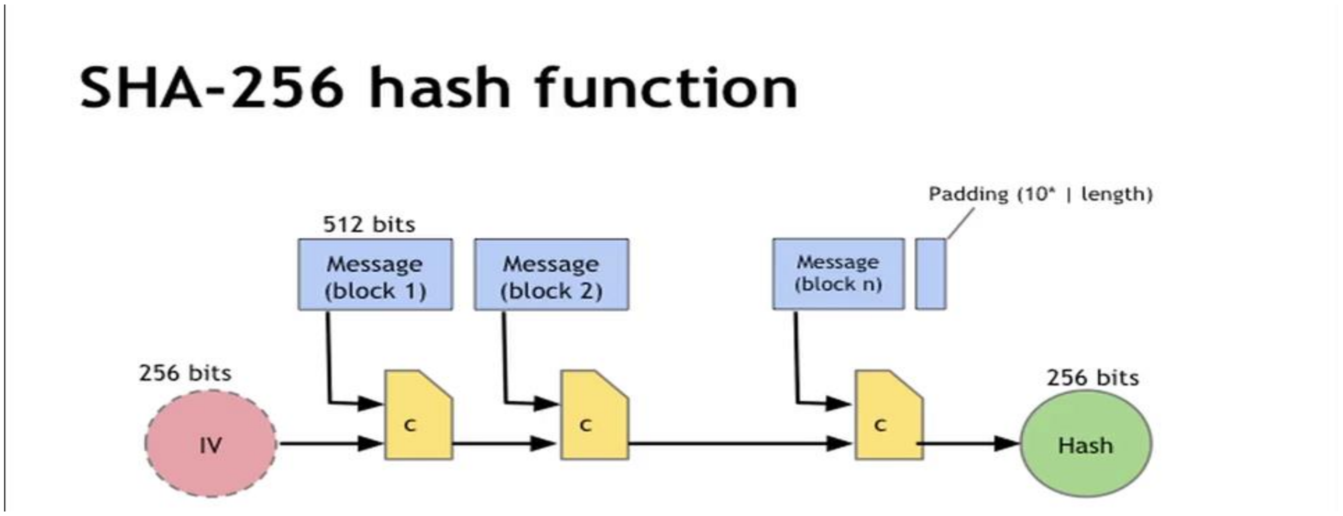
## HASHING ALGORITHM USED: SHA-256

SHA-256 is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2), and is one of the strongest hash functions available. SHA-256 is not much more complex to code than SHA-1, and has not yet been compromised in any way. The 256-bit key makes it a good partner-function for AES.

Here , in this project we use SHA256 to hash the generated AES KEY. Hash functions are not appropriate for storing encrypted passwords, as they are designed to be fast to compute, and hence would be candidates for brute-force attacks.

Key derivation functions such as bcrypt or scrypt are designed to be slow to compute, and are more appropriate for password storage (npm has bcrypt and scrypt libraries, and PHP has a bcrypt implementation with password_hash).

As the name suggest, it generates 256 bits of hashed string. As the number of bits increase, the probability of a collision decreases. PHP offers the built-in function hash(). The first argument to the function is the algorithm name (you can pass algorithm names like sha256, sha512, md5, sha1, and many others). The second argument is the string that will be hashed. The result it returns is the hashed string.



Syntax for sha256 in php:

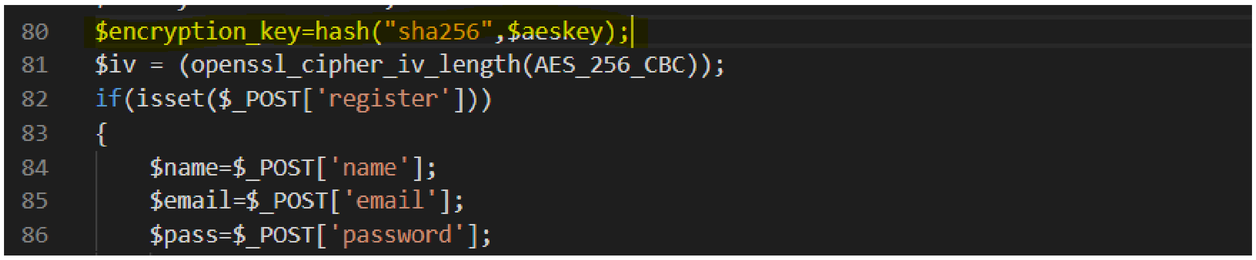
hash ( string $algo , string $data [, bool $raw_output = **FALSE** ] ) : string

in $algo , we put sha256 or md5 or sha1 , as per our choice.

$data is the aes generated key.

$raw_output when true gives binary result.

Example: $hashedkey=hash("sha256",$aeskey);

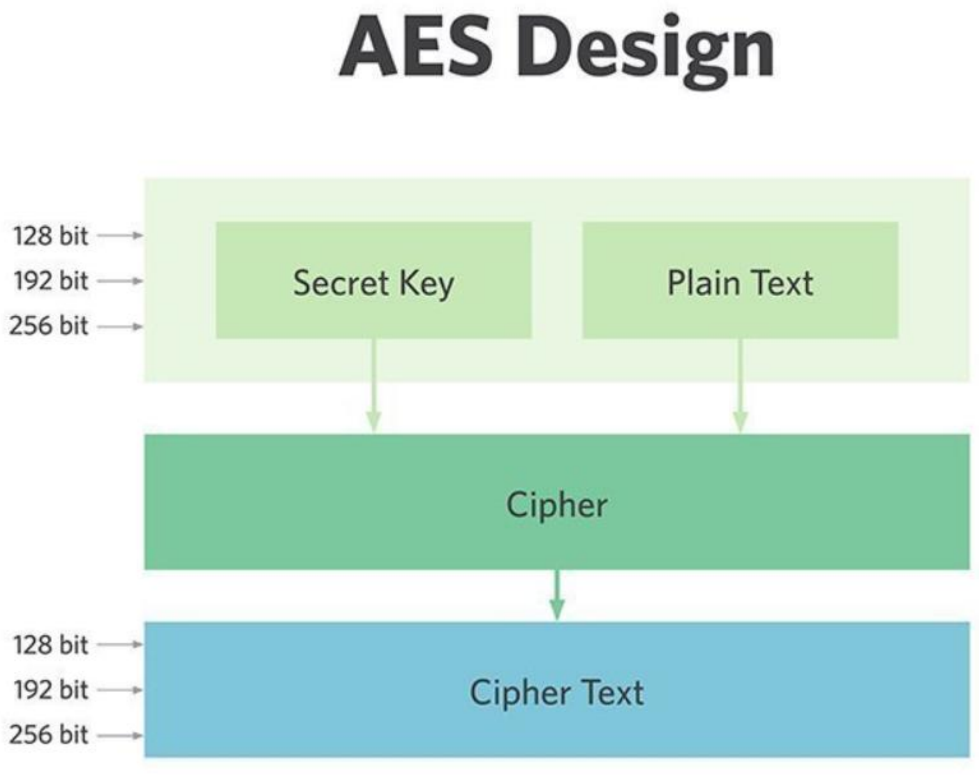


```
$encryption_key=hash("sha256",$aeskey);
$iv = (openssl_cipher_iv_length(AES_256_CBC));
if(isset($_POST['register']))
{
    $name=$_POST['name'];
    $email=$_POST['email'];
    $pass=$_POST['password'];
```

ENCRYPTION AGORITHM:

As I said in description of our system I used AES 256 algorithm. AES 256 is virtually impenetrable using brute-force methods. While a 56-bit DES key can be cracked in less than a day, AES would take billions of years to break using current computing technology. Hackers would be foolish to even attempt this type of attack.

It has 14 rounds for 256-bit keys. A round consists of several processing steps that include substitution, transposition and mixing of the input plaintext to transform it into the final output of ciphertext.



The AES encryption algorithm defines numerous transformations that are to be performed on data stored in an array. The first step of the cipher is to put the data into an array -- after which, the cipher transformations are repeated over multiple encryption rounds.

The first transformation in the AES encryption cipher is substitution of data using a substitution table; the second transformation shifts data rows, and the third mixes columns. The last transformation is performed on each column using a different part of the encryption key. Longer keys need more rounds to complete.

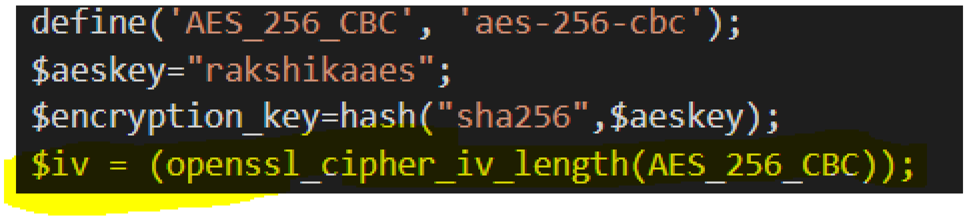
AES can be implemented in php using openssl_encrypt and openssl_decrypt. This helps to encrypt and decrypt the password.

Syntax:

```
openssl_encrypt ( string $data , string $method , string $key ,[, string $iv = "" ] )
: string
```

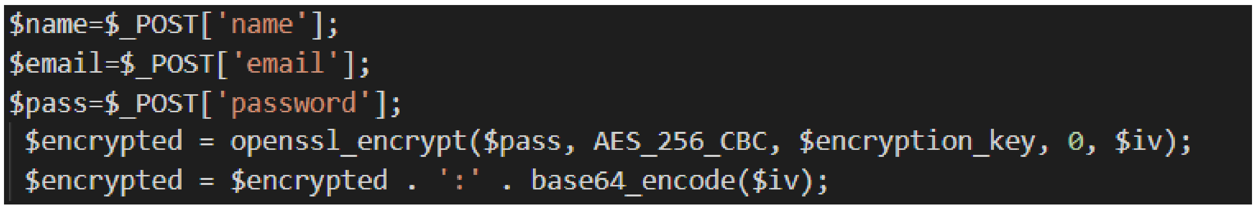
in $method we can specify the algorithm we wanted . in my case , I used *AES-256-CBC* . $iv is the initialisation vector which generated by openssl_randombytes().

The below pic shows generation of initialising vector.



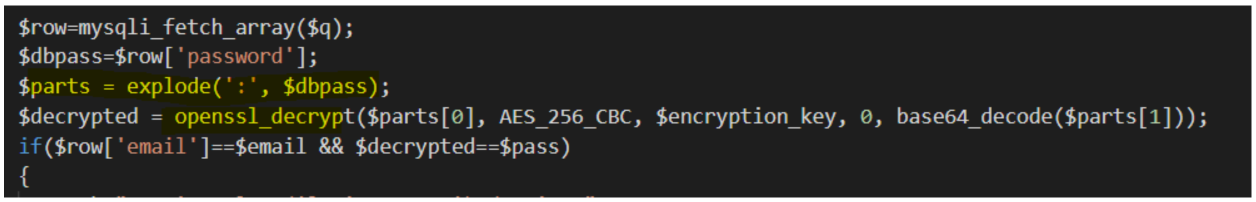
```
define('AES_256_CBC', 'aes-256-cbc');
$aeskey="rakshikaaes";
$encryption_key=hash("sha256",$aeskey);
$iv = (openssl_cipher_iv_length(AES_256_CBC));
```

The below shows the encryption part . after encryption we concatinate the encrypted with the initialising vector. Base64_encode is to give the output in readable format.



```
$name=$_POST['name'];
$email=$_POST['email'];
$pass=$_POST['password'];
 $encrypted = openssl_encrypt($pass, AES_256_CBC, $encryption_key, 0, $iv);
 $encrypted = $encrypted . ':' . base64_encode($iv);
```

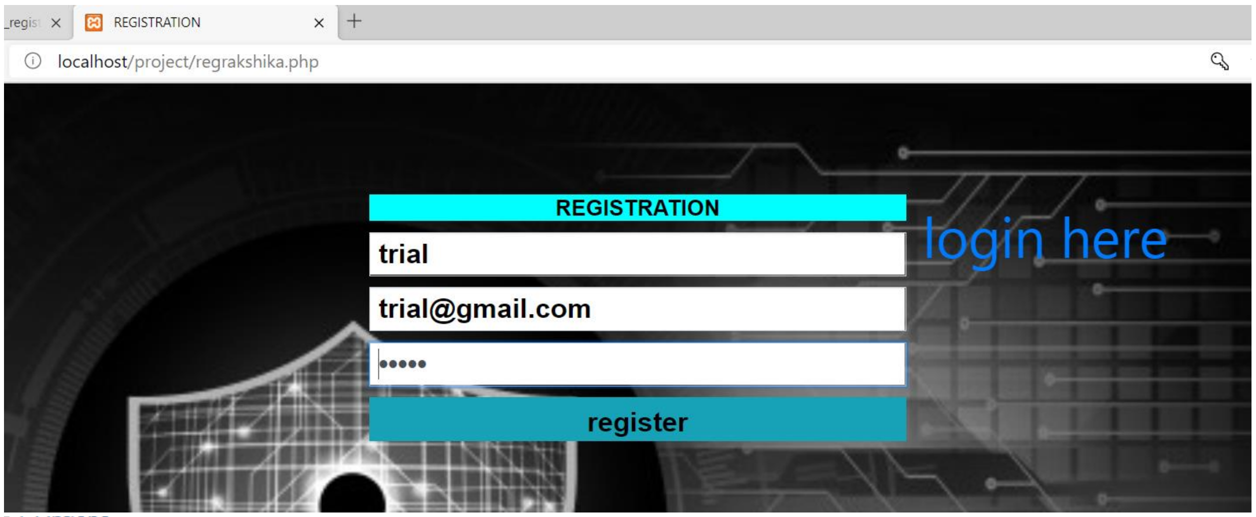
For decryption:
We separate the hashed password stored in database into to .we do this to compare and verify the data entered.the first part is decrypted part and next is the initialization vector part.
Openssl_decrypt() -→ decrypts the password.



```
$row=mysqli_fetch_array($q);
$dbpass=$row['password'];
$parts = explode(':', $dbpass);
$decrypted = openssl_decrypt($parts[0], AES_256_CBC, $encryption_key, 0, base64_decode($parts[1]));
if($row['email']==$email && $decrypted==$pass)
{
```

SAMPLE OUTPUT

Name : trial
Email : trial@gmail.com
Password: trial



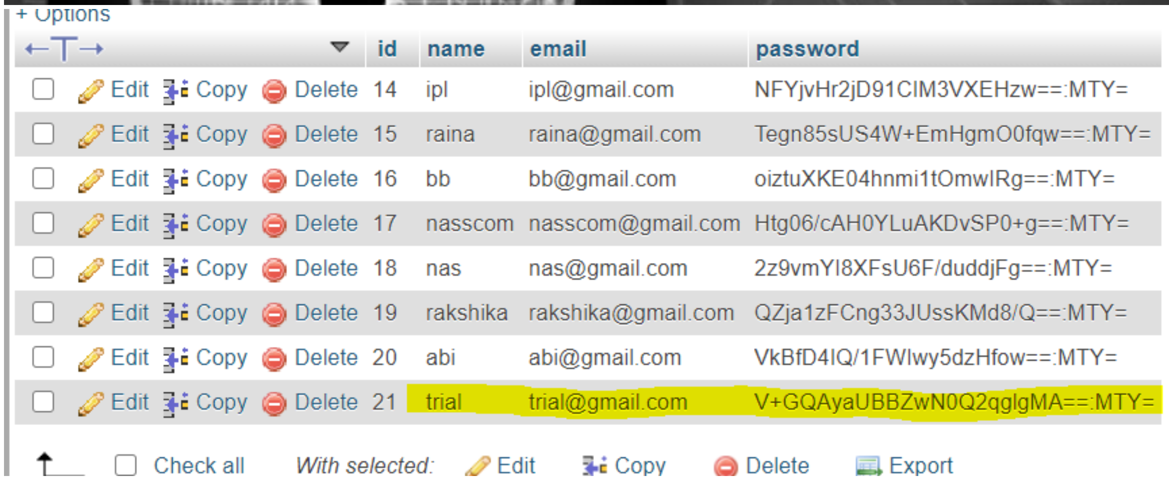
REGISTRATION

trial

trial@gmail.com

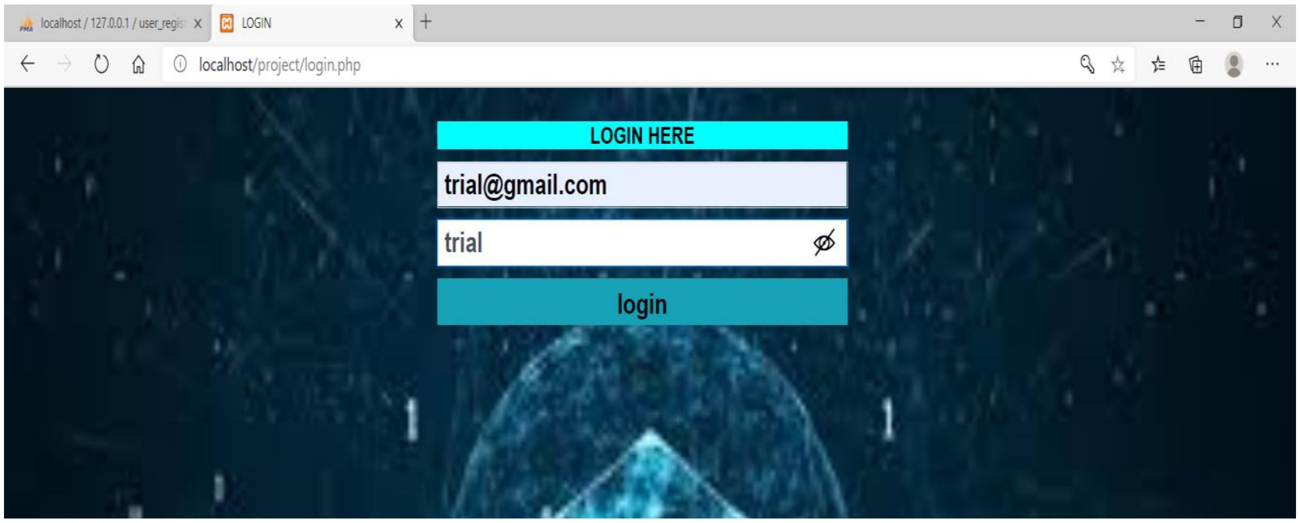
•••••

register



| | | | | id | name | email | password |
|---|---|---|---|---|---|---|---|
| ☐ | Edit | Copy | Delete | 14 | ipl | ipl@gmail.com | NFYjvHr2jD91CIM3VXEHzw==:MTY= |
| ☐ | Edit | Copy | Delete | 15 | raina | raina@gmail.com | Tegn85sUS4W+EmHgmO0fqw==:MTY= |
| ☐ | Edit | Copy | Delete | 16 | bb | bb@gmail.com | oiztuXKE04hnmi1tOmwIRg==:MTY= |
| ☐ | Edit | Copy | Delete | 17 | nasscom | nasscom@gmail.com | Htg06/cAH0YLuAKDvSP0+g==:MTY= |
| ☐ | Edit | Copy | Delete | 18 | nas | nas@gmail.com | 2z9vmYI8XFsU6F/duddjFg==:MTY= |
| ☐ | Edit | Copy | Delete | 19 | rakshika | rakshika@gmail.com | QZja1zFCng33JUssKMd8/Q==:MTY= |
| ☐ | Edit | Copy | Delete | 20 | abi | abi@gmail.com | VkBfD4IQ/1FWlwy5dzHfow==:MTY= |
| ☐ | Edit | Copy | Delete | 21 | trial | trial@gmail.com | V+GQAyaUBBZwN0Q2qglgMA==:MTY= |

Check all
With selected:
Edit
Copy
Delete
Export

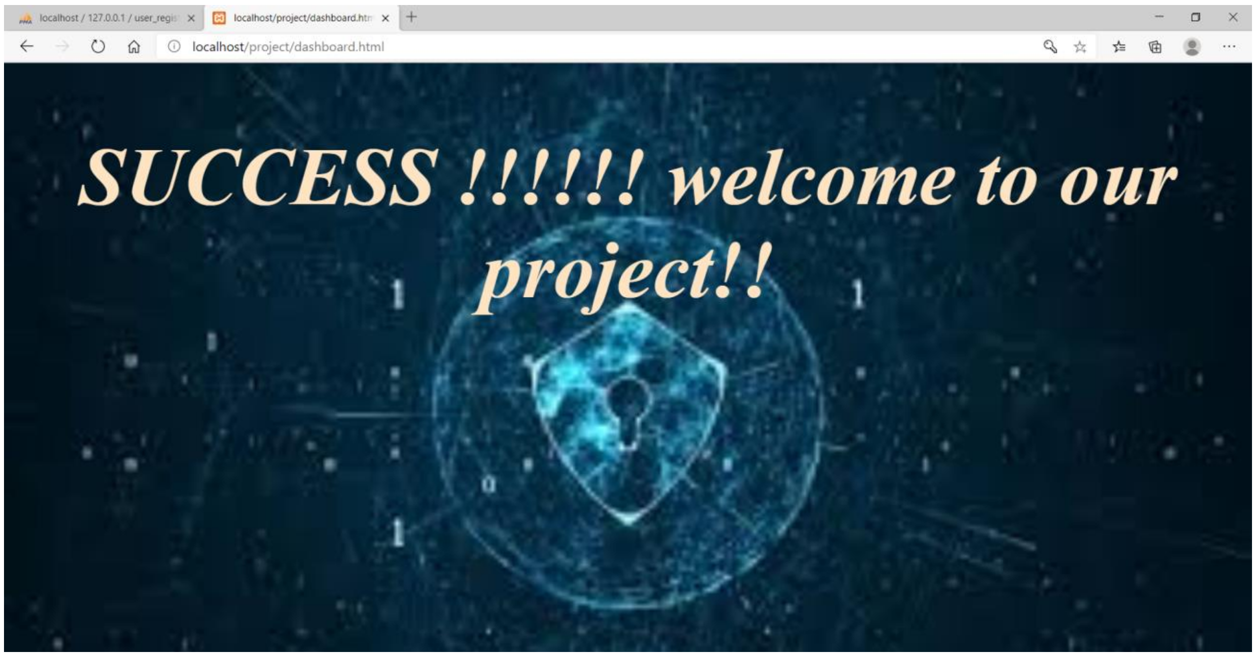

localhost / 127.0.0.1 / user_regis
LOGIN
localhost/project/login.php

LOGIN HERE

trial@gmail.com

trial

login


localhost / 127.0.0.1 / user_regis
localhost/project/dashboard.htm
localhost/project/dashboard.html

SUCCESS !!!!!! welcome to our project!!