National College of Ireland

Configuration Manual

MSc Internship
CyberSecuirty

Raja Sekhar Reddy Modugula
Student ID: x17151911

School of Computing
National College of Ireland

Supervisor: Mr Ross Spelmen

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Raja Sekhar Reddy Modugula
Student ID:	x17151911
Programme:	MSc in Cyber Security
Year:	2020
Module:	MSc Internship
Supervisor:	Mr Ross Spelmen
Submission Due Date:	17/08/2020
Project Title:	Configuration Manual
Word Count:	1320
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	17th August 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Raja Sekhar Reddy Modugual
x17151911

1 Introduction

This configuration manual supplies the brief understanding of technical aspects involved in augmenting the password security using hybrid approach of Argon2i hashing and AES algorithm. This configuration manual includes system setup, software/hardware specifications, installation of artefacts and libraries, evaluation of performance which are presented in detail. To understand the previous studies and functional concepts related to proposed model, kindly refer the research project report.

2 Configuration requirements of the system

2.1 Hardware Requirements

Processor	Intel(R) Core (TM) i7-7500U CPU @2.70GHz 2.90GHz 2cores
Installed Memory (RAM)	16GB
Hard Disk (HDD)	2TB
Graphics Card (GPU)	AMD Radeon TM R7 M445 Graphics with 4GB GDDR5 Graphics Memory
System Type	64-bit Operating System, x64-based processor
Operating System	Windows 10 Home

2.2 Software Requirements

Programming Languages: HTML, CSS, JavaScript, Bootstrap, PHP, AJAX.
Software Tools: Microsoft Visual Studio is an integrated development environment used for writing and editing code for developing web applications. We have used XAMPP, which is a free and open source web server stack package developed by Apache Friends. XAMPP has built-in HTTP server part which allows your personal computer to host locally and act as a web server.

3 Operation of the System

The main idea of research was using a user supplied password for key derivation using Argon2 key derivation function and then use the key for AES encryption and decryption processes. User must register by supplying credentials in the registration form with first name, last name, email, password. If the registration was successful, he can use the same email and password used during the time of registration to successfully authenticate

and login into the web application. There are three main phases involved in proposed methodology namely, key derivation phase, AES encryption phase, and AES decryption phase.

3.1 key derivation phase

In this key derivation phase, we have generated the key using a password based key derivation function called Argon2i in PHP. Argon2i is the evolution of Bcrypt and Scrypt algorithms for supplying security against dangerous attacks like Brute force attacks. In this phase we have predefined parameters like memory cost = 2048, time cost = 4, and threads = 3 to avoid GPU attacks. During registration, the password given by the corresponding user is passed through Argon2i hashing for obtaining the hashed password. code below shows the implementation of Argon2i hashing.

```
$passwordhashed="";
try {
    $currentTime = microtime(true);
    $hashed_password = password_hash($Password, PASSWORD_ARGON2I,['memory_cost' =>
                $passwordhashed=AESEncryption::encrypt($hashed_password);
                error_log("encrypt time interval in milli seconds during registraic
    array_push($error_array, "Something went wrong ! we will get back soon <br>");
    exit();
}
```

3.2 AES encryption phase

In this AES encryption phase, the hashed password obtained from password hashing is encrypted using AES-256-CBC method. In order to generate the cipher data, we give inputs to openssl encrypt like plain-text password, AES-256-CBC, Key (salt), iv (Initialization vector) generated by openssl random pseudo of length 16 bytes and OPENSSL RAW DATA. The resultant cipher data along with iv, hash key (password hashed using Argon2i) are again hashed using HMAC method for generating another keyed hash value. We have used base64 encoder to encode the iv, hash, cipher data for cutting special characters before storing it in data base. The final password stored in data base is 192 bytes. code below shows the password encryption and generated keyed hash value using HMAC method.

```php
public static function encrypt($plaindata)
{
    //string $cryptokey key for encryption (with 256 bit of entropy)
    $cryptokey=self::getCryptKey();
    //string $hashkey key for hashing (with 256 bit of entropy)
    $hashkey=self::getHashKey();
    $method = "AES-256-CBC"; //cipher method
    $key = hash('sha256', $cryptokey, true);
    $iv = openssl_random_pseudo_bytes(16);
    $cipherdata = openssl_encrypt($plaindata, $method, $key, OPENSSL_RAW_DATA, $iv);

    if ($cipherdata === false)
    {
        $cryptokey = "**REMOVED**";
        $hashkey = "**REMOVED**";
        throw new \Exception("Internal error: openssl_encrypt() failed:".openssl_err
    }
    // Generate a keyed hash value using the HMAC method
    $hash = hash_hmac('sha256', $cipherdata.$iv, $hashkey, true);

    if ($hash === false)
    {
        $cryptokey = "**REMOVED**";
       $hashkey = "**REMOVED**";
        throw new \Exception("Internal error: hash_hmac() failed");
    }

    return base64_encode($iv.$hash.$cipherdata);
```

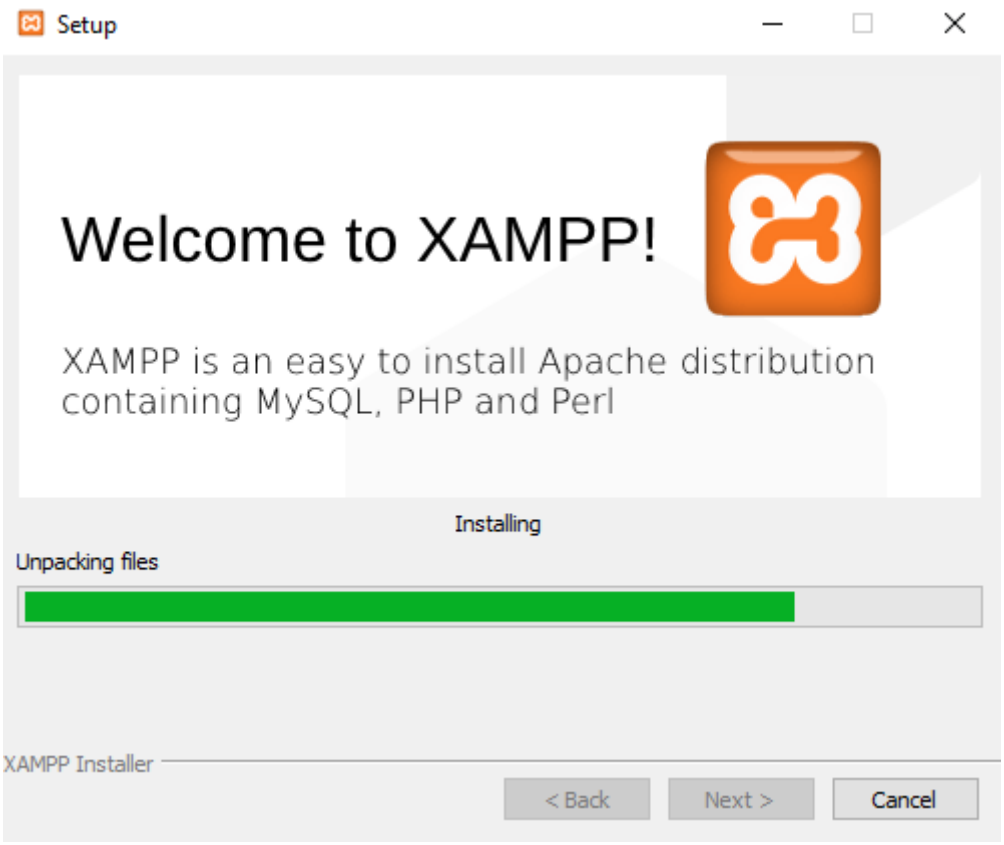
3.3 AES decryption phase

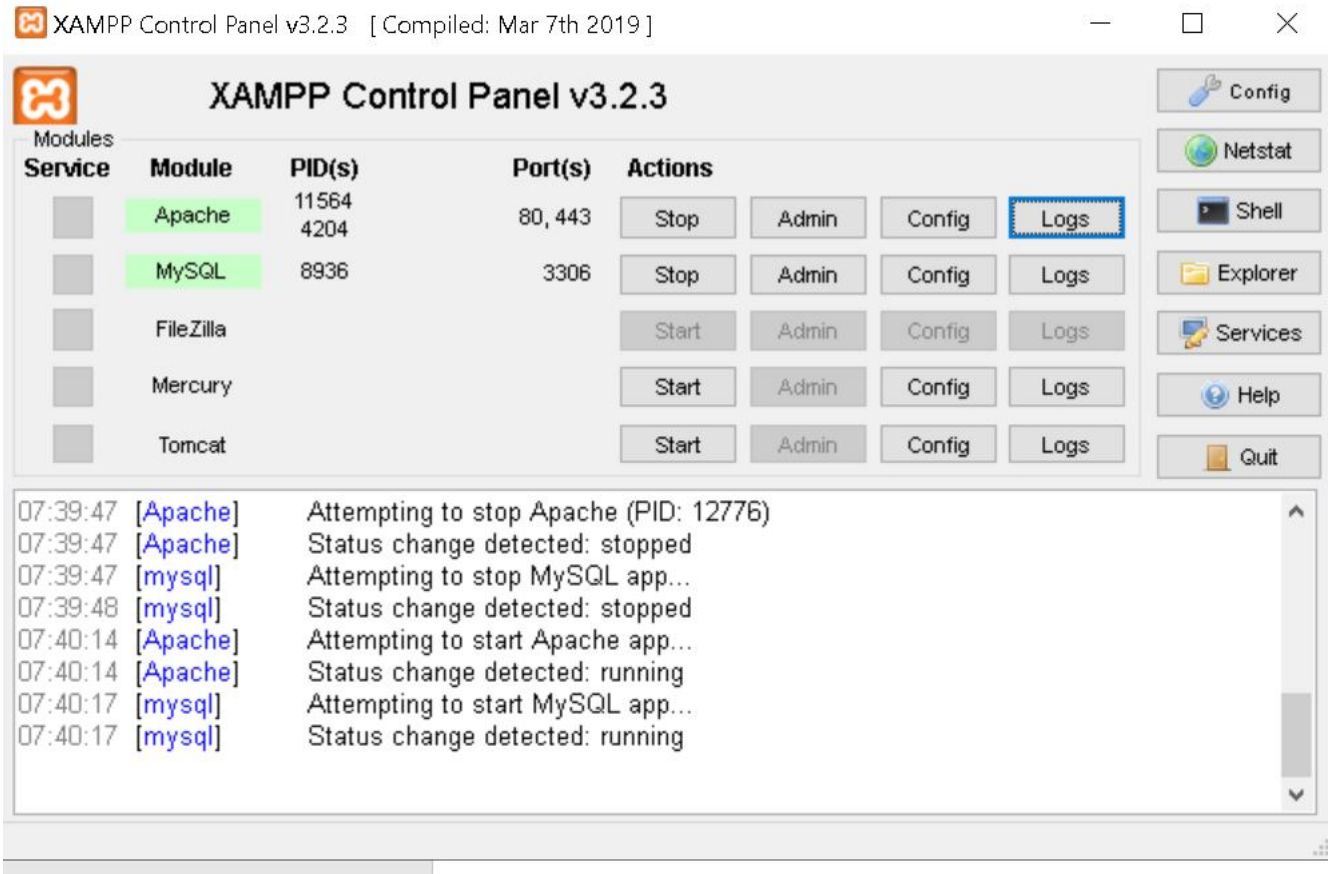
In AES decryption phase, we start by base64 decoding the encrypted data and generate the keyed hash value using HMAC (SHA-256) and confirm with the keyed hash generated in AES encryption phase for validation. After hash verification, we get the plain data using openssl decrypt function with cipher data, iv, key, AES-256-CBC, and OPENSSL RAW DATA for returning plain text. Code below shows the keyed hash validation and password decryption.

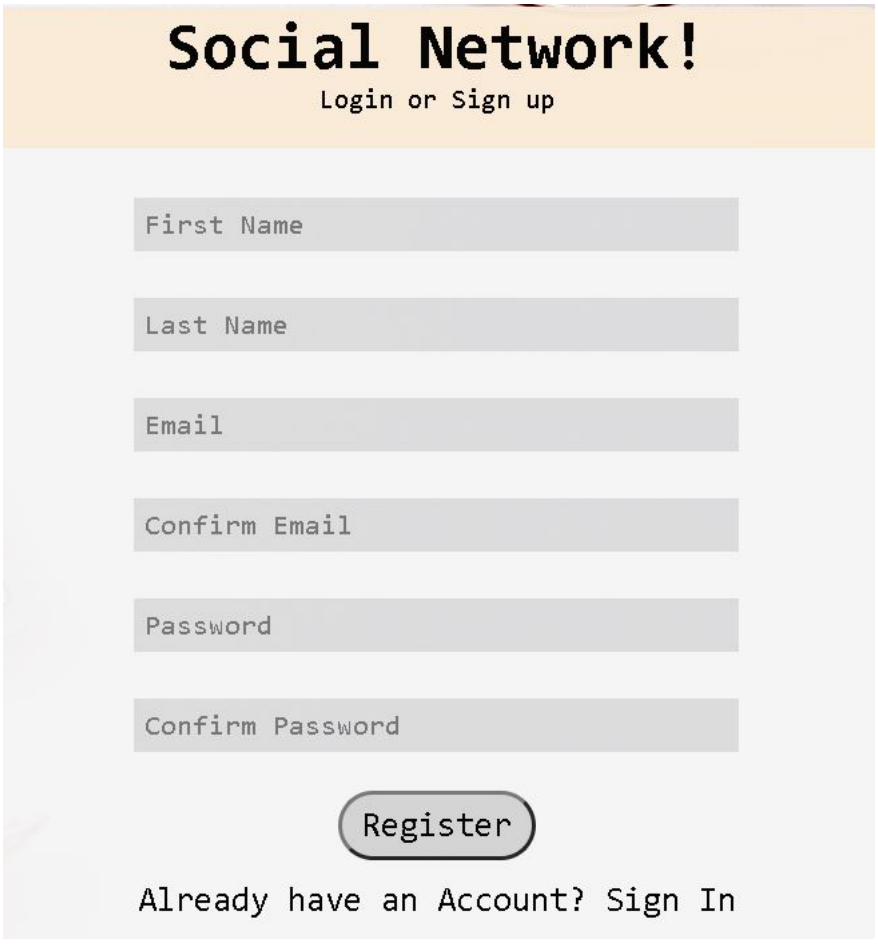
```php
public static function decrypt($encrypteddata)
{
    $encrypteddata=  base64_decode($encrypteddata);
    //string $cryptokey key for encryption (with 256 bit of entropy)
    $cryptokey=self::getCryptKey();
    //string $hashkey key for hashing (with 256 bit of entropy)
    $hashkey=self::getHashKey();
    $method = "AES-256-CBC"; //cipher method
    $key = hash('sha256', $cryptokey, true);
    $iv = substr($encrypteddata, 0, 16);
    $hash = substr($encrypteddata, 16, 32);
    $cipherdata = substr($encrypteddata, 48);
    // Generate a keyed hash value using the HMAC method and check with hash if hash
    if (!hash_equals(hash_hmac('sha256', $cipherdata.$iv, $hashkey, true), $hash))
    {
        $cryptokey = "**REMOVED**";
        $hashkey = "**REMOVED**";
        throw new \Exception("Internal error: Hash verification failed");
    }

    $plaindata = openssl_decrypt($cipherdata, $method, $key, OPENSSL_RAW_DATA, $iv);
```

3.4 Xampp Installation

Xampp is an open source web application tool used for running web applications developed by apache friends to set up PHP development environment . We have downloaded xampp windows 64bit installer and click next. Choose the root directory path to set up the htdocs folder for our application. Click the allow access button to allow the xampp modules from the windows firewall. After the installation process click finish button in the xampp wizard setup. To start the Apache and MySQL, click on the start button on the xampp control panel. In order to run the code, first copy the code to htdocs folder and import the SQL file to the MySQL database. To record the encryption time and decryption time we have used apache error log in logs from xampp control panel. The time recorded in error log was in milli seconds which was converted to seconds and noted down.


Setup
Welcome to XAMPP!
XAMPP is an easy to install Apache distribution containing MySQL, PHP and Perl
Installing
Unpacking files
XAMPP Installer
< Back
Next >
Cancel


XAMPP Control Panel v3.2.3 [Compiled: Mar 7th 2019]
XAMPP Control Panel v3.2.3
Modules
Service
Module
PID(s)
Port(s)
Actions
Apache
11564
4204
80, 443
Stop
Admin
Config
Logs
MySQL
8936
3306
Stop
Admin
Config
Logs
FileZilla
Start
Admin
Config
Logs
Mercury
Start
Admin
Config
Logs
Tomcat
Start
Admin
Config
Logs
Config
Netstat
Shell
Explorer
Services
Help
Quit
07:39:47 [Apache] Attempting to stop Apache (PID: 12776)
07:39:47 [Apache] Status change detected: stopped
07:39:47 [mysql] Attempting to stop MySQL app...
07:39:48 [mysql] Status change detected: stopped
07:40:14 [Apache] Attempting to start Apache app...
07:40:14 [Apache] Status change detected: running
07:40:17 [mysql] Attempting to start MySQL app...
07:40:17 [mysql] Status change detected: running



3.5 Evaluation

To evaluate the performance of the proposed model, we have recorded the performance measures like Encryption time, Decryption time, and throughput.
Encryption time is defined as the time taken by Argon2i hashing and AES scheme together to convert the plain-text password to cipher-text password. Similarly, decryption time is

defined as the time taken by AES scheme to convert the cipher-text password to plain-text password.
Another performance measure throughput can be calculated by dividing the size of plain-text password with the time taken for encryption. Tables 1 and 2 shows the test data generated from registered users and Analysis of encryption and decryption time recorded. Table:1 is the data considered for calculating Encryption time and Decryption time for our case studies. we have implemented our proposed model on three different machines and the results are noted down in Tables 2,3, and 4 respectively.

Email Address	Password as plain-text	Password Size in MB
testcase1@gmail.com	Myfirsttestcase1	0.000016
testcase2@gmail.com	Mysecondtestcase2	0.000017
testcase3@gmail.com	Mythirdtestcase3	0.000016
testcase4@gmail.com	Myfourthtestcase4	0.000017
testcase5@gmail.com	Myfifthtestcase5	0.000016
testcase6@gmail.com	Mysixthtestcase6	0.000016
testcase7@gmail.com	Myseventhtestcase7	0.000018
testcase8@gmail.com	Myeighttestcase8	0.000016
testcase9@gmail.com	Myninthtestcase9	0.000016
testcase10@gmail.com	Mytenthtestcase10	0.000017

Table1: Test Data considered for evaluating this model

3.5.1 Case study: 1

In our case study 1, we have implemented the proposed model on 2.70 GHZ (2cores) Intel i7 processor, with installed memory of 16.00 GB RAM, AMD Radeon TM R7 M445 Graphics with 4GB GDDR5 Graphics Memory, 64-bit Operating system (Windows 10 Home). In our proposed model, we have calculated the encryption and decryption time in Milli seconds and later converted to seconds using Milliseconds to Seconds Conversion calculator. In Table 2 we have noted the results of encryption time, decryption time and further calculated the throughput.

Usernames	Argon2i + AES Encryption (Time in seconds)	AES Decryption (Time in seconds)
Test_case1	0.036	0.037
Test_case2	0.038	0.038
Test_case3	0.036	0.038
Test_case4	0.037	0.036
Test_case5	0.036	0.042
Test_case6	0.039	0.037
Test_case7	0.037	0.037
Test_case8	0.049	0.035
Test_case9	0.036	0.036
Test_case10	0.037	0.036
Average Time	0.381	0.372
Throughput (MB/Seconds)	0.000433071	0.000443548

Table 2: Analysis of Encryption time and decryption time.

3.5.2 Case study: 2

In our case study 2, we have implemented our proposed model on 1.70GHz Intel core i5-3317U with installed memory of 8.00 GB, windows 10 home operating system. Firstly , we have created test cases from 1 to 10 with the same email addresses and passwords mentioned in the Table:1. By using the Apache error log from logs in xampp control panel, we have recorded the encryption time and decryption for the test cases 1 to 10 and tabulated the results in Table: 3. The encryption and decryption time listed is converted from milli seconds to seconds and later listed down in Table: 3.

Usernames	Argon2i + AES Encryption (Time in seconds)	AES Decryption (Time in seconds)
Test_case1	0.2	0.228
Test_case2	0.249	0.195
Test_case3	0.221	0.204
Test_case4	0.197	0.198
Test_case5	0.196	0.193
Test_case6	0.190	0.209
Test_case7	0.192	0.189
Test_case8	0.187	0.202
Test_case9	0.191	0.193
Test_case10	0.214	0.193
Average Time	2.037	2.004
Throughput (MB/Seconds)	0.00007364	0.00008234

Table: 3 Analysis of Encryption time and Decryption time

3.5.3 Case study: 3

In our case study 3, we have implemented our proposed model on 1.80GHz Intel core i5-3337U CPU with installed memory of 4GB, 64-bit operating system. Similar to the above case studies, we have first registered test cases from 1 to 10 with same usernames and passwords as mentioned in Table: 1. To record the encryption time and decryption time, we have used the Apache log from logs in the Xampp control panel. The encryption and decryption time listed is converted from milli seconds to seconds and later listed down in Table: 4.

Usernames	**Argon2i + AES Encryption** (Time in seconds)	**AES Decryption** (Time in seconds)
Test_case1	0.092	0.060
Test_case2	0.065	0.060
Test_case3	0.063	0.063
Test_case4	0.064	0.065
Test_case5	0.065	0.060
Test_case6	0.059	0.059
Test_case7	0.061	0.062
Test_case8	0.061	0.081
Test_case9	0.068	0.065
Test_case10	0.061	0.062
Average Time	0.659	0.637
Throughput (MB/seconds)	0.00025038	0.00025903

Table: 4 Analysis of Encryption time and Decryption time

3.5.4 Comparison of Analysis of Encryption time with existing models

we have compared the encryption time of our proposed model with existing hybrid models like Bcrypt with AES, and Scrypt with AES. The results shows that our model have recorded the fastest encryption time compared to the previous approaches as shown in Table: 5.

Bcrypt + AES Encryption (Time in seconds)	**Scrypt + AES Encryption** (Time in seconds)	**Argon2i + AES Encryption** (Time in seconds)
0.60741	0.47530	0.036
0.27369	0.11730	0.038
0.27692	0.10961	0.036
0.27889	0.12441	0.037
0.27539	0.12398	0.036
0.27881	0.11792	0.039
0.27279	0.12204	0.037
0.27164	0.12123	0.049
0.27629	0.12072	0.036
0.28850	0.12368	0.037

Table: 5 Analysis of Encryption time in seconds

3.5.5 Comparison of Analysis of Throughput with existing models

we have compared the Throughput of our proposed model with existing hybrid models like Bcrypt with AES, and Scrypt with AES. The results shows that our model have recorded the highest Throughput compared to the earlier approaches as shown in Table: 6.

Throughput of Bcrypt + AES (MB/seconds)	Throughput of Scrypt + AES (MB/seconds)	Throughput of Argon2i + AES (MB/seconds)
0.0000164632	0.0000210389	0.00458333
0.0000365368	0.0000852484	0.00434211
0.0000252778	0.0000638612	0.00458333
0.0000430267	0.0000964499	0.00445946
0.0000254176	0.0000564571	0.00458333
0.0000286924	0.0000678377	0.00423077
0.0000329924	0.0000737452	0.00445946
0.0000404937	0.0000907333	0.00336735
0.0000325742	0.0000745485	0.00458333
0.0000311951	0.0000727654	0.00445946

Table: 6 Analysis of Throughput in MB/seconds