

### My Idea of Security:

A completely secure system is a virtual impossibility, so an approach often used in the security profession is one of balancing risk and usability.

Note : We cannot use OpenSSL at the client side for generating a key or encrypting a data since in a real world scenario, we cannot “ask” our client to install openssl and encrypt the data. Hence, we implement some parts of application in javascript and PHP. PHP is a system-side language like C.

### **SECURITY ANALYSIS:**

#### Scenario :

We have two sides which are communicating, client and server. There is a strong finite possibility that the adversary is able to intercept the communication channel between both the parties (and so has the encrypted data). Now, our primary target was to secure that data.

We use a combination of public key and private key cryptography to maximize security. We use the public key encryption protocols to agree upon a secure strong key that will be further used for data exchange using asymmetric cryptography and further use that key to exchange data using the private key cryptography.

We have chosen reasonably strong cryptographic protocols : RSA for private key cryptography and AES for public key cryptography.

For authentication, we use the SHA-3 Hashing. We store the username, password-hash pairs in a database instead of storing the password itself. This improves the security. SHA-3 is a notable improvement over SHA-2.

We try to generate the keys for every “session”. For every such session when the client sent the request, the server generates a pair of public and private keys and sends the public key to the client. The client, then generates a random AES key, encrypts it using the public key and returns back to the server. The server then decrypts the message using its private key and then uses the decrypted key for subsequent communication.

We also design the application to use HTML 5 secure local storage which reduces the chances of compromising the security of client side. Client side may be the most vulnerable side since we cannot predict the security infrastructure of the client browser and/or machine.

Since we cannot force the client to install OpenSSL, we have to rely on the accepted technology standard for all clients for the generation of random AES keys. This may have a security implication, though. However, even if the private keys of a client is compromised, since we are also using the private key infrastructure, we are sure that exactly that client is compromised and no other client or the server is compromised.

From the application security side, since we store content on the hard drive and the passwords in the database, two types of security implications come into picture: Database Security and File-System Security.

The heart of this security protocol is the public key infrastructure. In an event where the private key is compromised, the attacker would be able to read the cipher texts. However, would may not be able to send messages in the name of the client.

We have also used AJAX so that the data transferred is not visible to client. With this, if attacker is client's friend he cannot see the data being sent.

### **POSSIBLE ATTACKS:**

Data eavesdropping :

There may be a possibility that the SHA-3 hashed that are being used to authenticate may be eavesdropped by the attacker. But because of pigeonhole principle, it is not possible to get the password string from the SHA-3 hash (This holds since the password space is substantially larger than the hash-key-space.)

Encrypted Blogs (Symmetric key cryptography): It may be possible that the attacker gets the encrypted blogs. But it is relatively difficult for the attacker to decrypt it as its hardness depends on the Substitution and Permutation boxes.

Encrypted Keys: We Use RSA to agree upon a key that will be used for secure communication exchange. It may be possible for the attacker to get the encrypted version of the key.

It may also be possible for attacker to get the public key. We also use a separate public key for each session, thus increasing the hardness for the attacker.

Data Modification :

Hashed Passwords: It may not be possible or negligibly possible for the attacker to modify the hashed keys since we use SHA-3 hash. However, once the attacker gets the has for a particular user (either if the database is compromised or by manipulating computer networks), the security of that particular user may be compromised and the attacker may modify the contents of that blog.

However, security of other users (and the server itself) will not be affected and in this sense our system is robust.

### Encrypted Content :

Since we assume in the design that for each time we communicate, we use a new pair of keys for agreeing on an AES key and that AES key is random

Moreover, we refresh these keys for each session. Thus, if the attacker gets keys for one session, this in no way helps him to attack in another session.

Also, AES, by the virtue of design is resistant to data modification and would not decrypt if the data is modified in between.

Also, in the design, we have modules that generate a separate key for encrypting a file. Thus, we have two keys for encryption : one generated by the client for transferring data and another generated by the server for storing data. This will improve the hardness for the attacker.

### Data Replay attacks:

Note: Please look at the script and ensure that we have used session Management (in PHP) Because we use a new session each time we log in, it makes no sense for the attacker to store the data and replay it later in another session.

Integrity Detection : We are using Hashes for strong passwords.

**TESTING (Link to the code : [https://github.com/rajatllt/crypto\\_hw\\_project2](https://github.com/rajatllt/crypto_hw_project2)) :**

For testing of my webapp, please walk through the steps. I have did testing and have attached relevant screenshots in place.

**Configuration:**

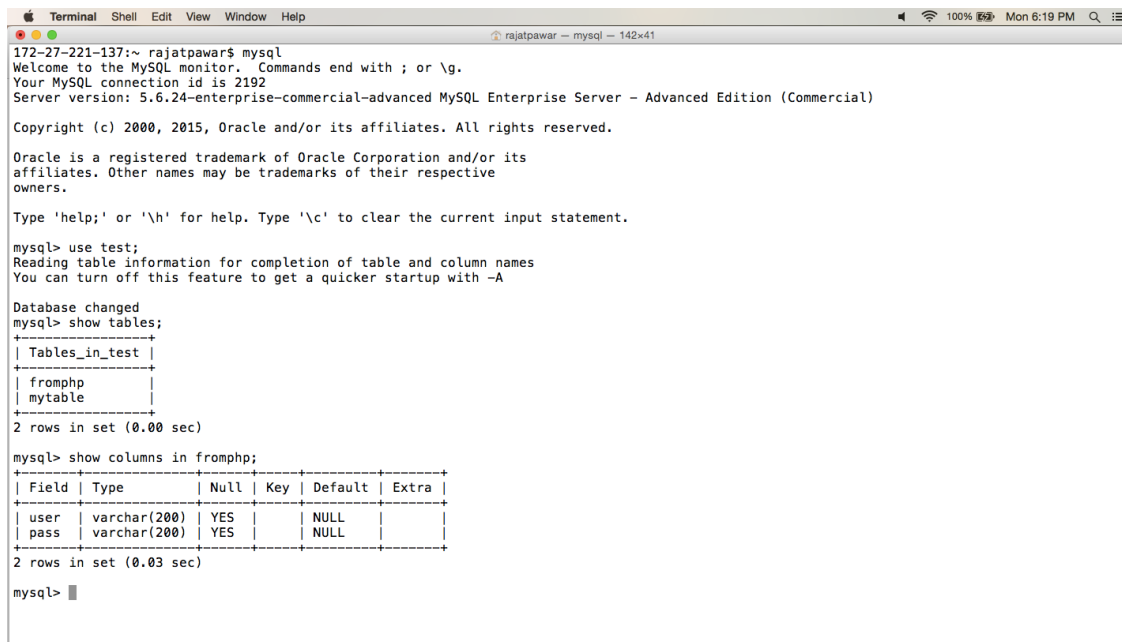
Assuming that the latest version of PHP is installed (module enabled in Apache HTTP Server) and a HTML5 compatible browser preferably Chrome is used for testing. PLEASE KEEP THE CONSOLE OPEN AT ALL TIMES. It display behind-the-scenes-cryptography information.

This application also assumes that a mysql server is running on port number 2023 and has a database by the name test. This database test has a table named fromphp which stores the (login,hash) pairs of the users. Also, the database is logged in using the user “rajat” and the password “” (null) . I request you to create user with this name or modify the code as it is convenient to yourself.

Please place the folder /cryptotest/ in the documents directory of your web-server.

Note : We are running mysql on port 2023. Please start mysql server on that port. There was some issue with my laptop so I had to do use this non-standard port.

Also the app assumes that there is a database by the name test and there is a table by name fromphp in it. This table has two fields : user and pass as shown in the screenshot below:



```

172-27-221-137:~ rajatpawar$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2192
Server version: 5.6.24-enterprise-commercial-advanced MySQL Enterprise Server - Advanced Edition (Commercial)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

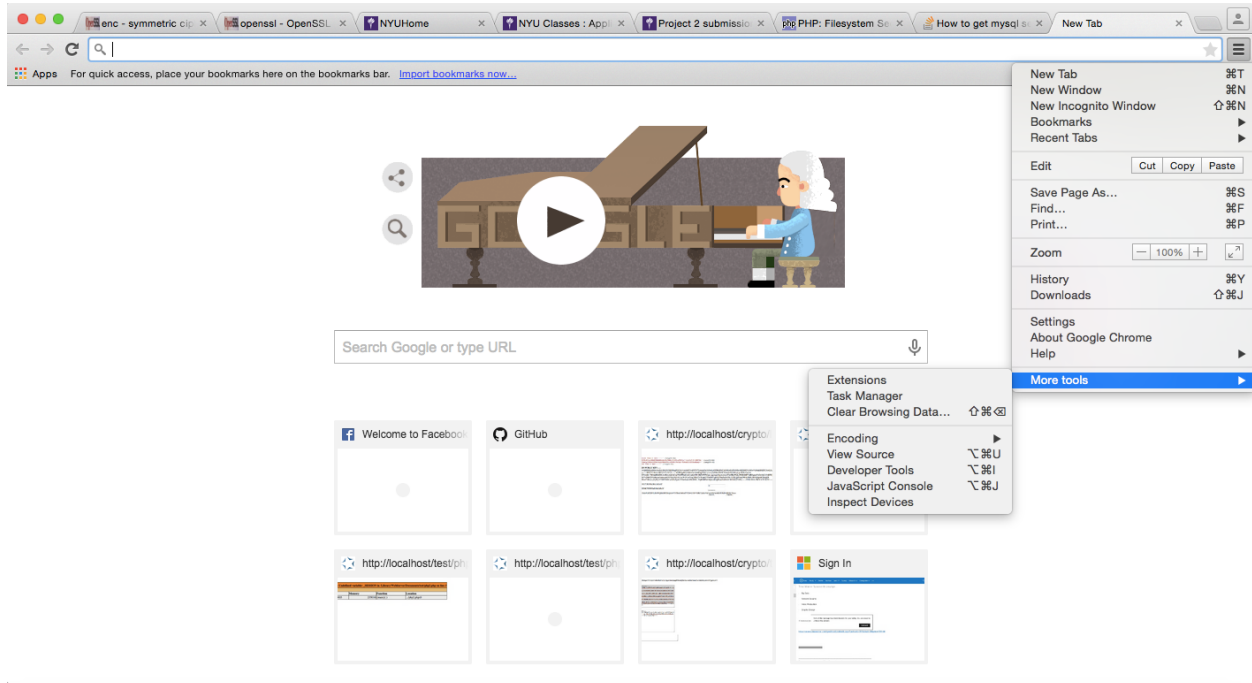
Database changed
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| fromphp        |
| mytable        |
+-----+
2 rows in set (0.00 sec)

mysql> show columns in fromphp;
+----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+----+-----+-----+-----+-----+-----+
| user  | varchar(200)  | YES  |     | NULL    |       |
| pass  | varchar(200)  | YES  |     | NULL    |       |
+----+-----+-----+-----+-----+-----+
2 rows in set (0.03 sec)

mysql>

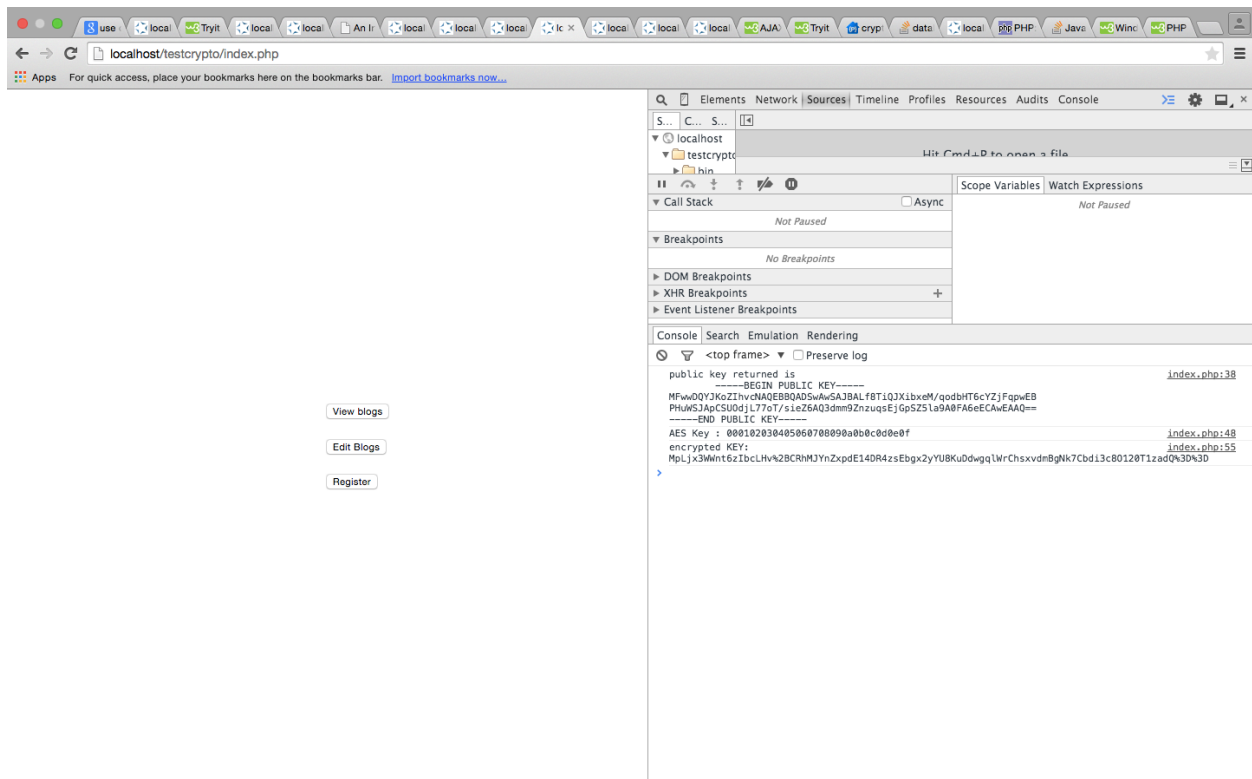
```

Please open the Developer Tools by clicking on the Hamburger Button in the Chrome, then click on “More Tools” and finally on the “Developer Tools”. This shall display the javascript console which will be primarily used for viewing results. As follows:



### Key Generation:

1) First, open the page index.php ; on loading this page, the server generates a public/private key pair and sends the public key to the client. Observe the output of the console to see the public key returned by the server.

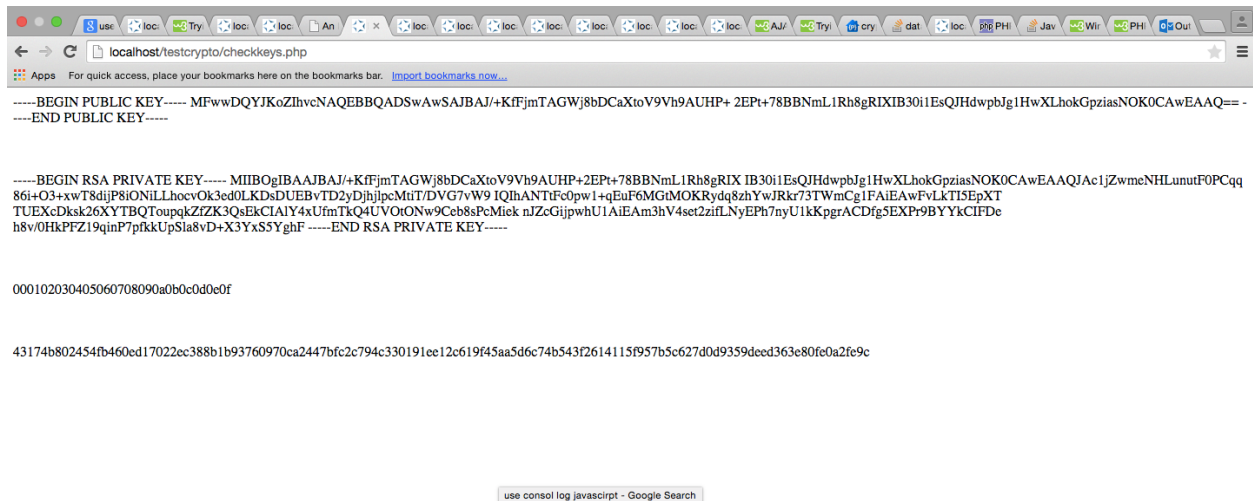


Every time you refresh the page, a new pair is generated which you can check by refreshing the page checkkeys.php .

Note : At any point of time, there exists a unique public and private key. To view these keys, open the page checkkeys.php . This page shows the current public and private key assigned to the client We have developed the app assuming one client here. [in a real world scenario, the server would store the keys by using IP addresses. Ex. for IP 232.34.53.55, the name of variable that stores the key would be 232.34.53.55-public , 232.34.53.55-private and 232.34.53.55-symmetric .]

Note : Observe the scripts keygenerationforclient.php, checkkeys.php and index.php with reference to key generation by the server. All are well commented.

A screenshot of checkkeys.php (The last two entries are AES key and hash key of current user respectively)



```
-----BEGIN PUBLIC KEY----- MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAL/+KfFjmTAGWj8bDCaXtoV9Vh9AUHP+ 2EPt+78BBNmL1Rh8gRiXIB30i1EsQJHdwbpJg1HwXLhokGpziasNOK0CAwEAAQ== -
-----END PUBLIC KEY-----

-----BEGIN RSA PRIVATE KEY----- MIIBOgIBAAJBAL/+KfFjmTAGWj8bDCaXtoV9Vh9AUHP+2EPt+78BBNmL1Rh8gRiX IB30i1EsQJHdwbpJg1HwXLhokGpziasNOK0CAwEAAQAc1JzwmeNHLunutF0PCqq
86i+O3+xwT8dijP8iONiLLhocvOk3ed0LKDsDUEBvTD2yDjhjpcMtiT/DVG7vW9 IQlhANTIfc0pw1+qEuF6MGtMOKRydq8zhYwJRkr73TWmCg1FAiEawFvLkTt5EpXT
TUEXcDksk26XYTBQToupgkZfZK3QsEkCIAIY4xUfmTkQ4UVOtONw9Ceb8sPcMiek nJZcGijpwhU1AiEAm3hV4set2zifLnyEPPh7nyU1kKpgrACDfg5EXPr9BYkCIFDe
h8v/0HkPFZ19qinP7pfkkUpSla8vD+X3YxS5YghF -----END RSA PRIVATE KEY-----

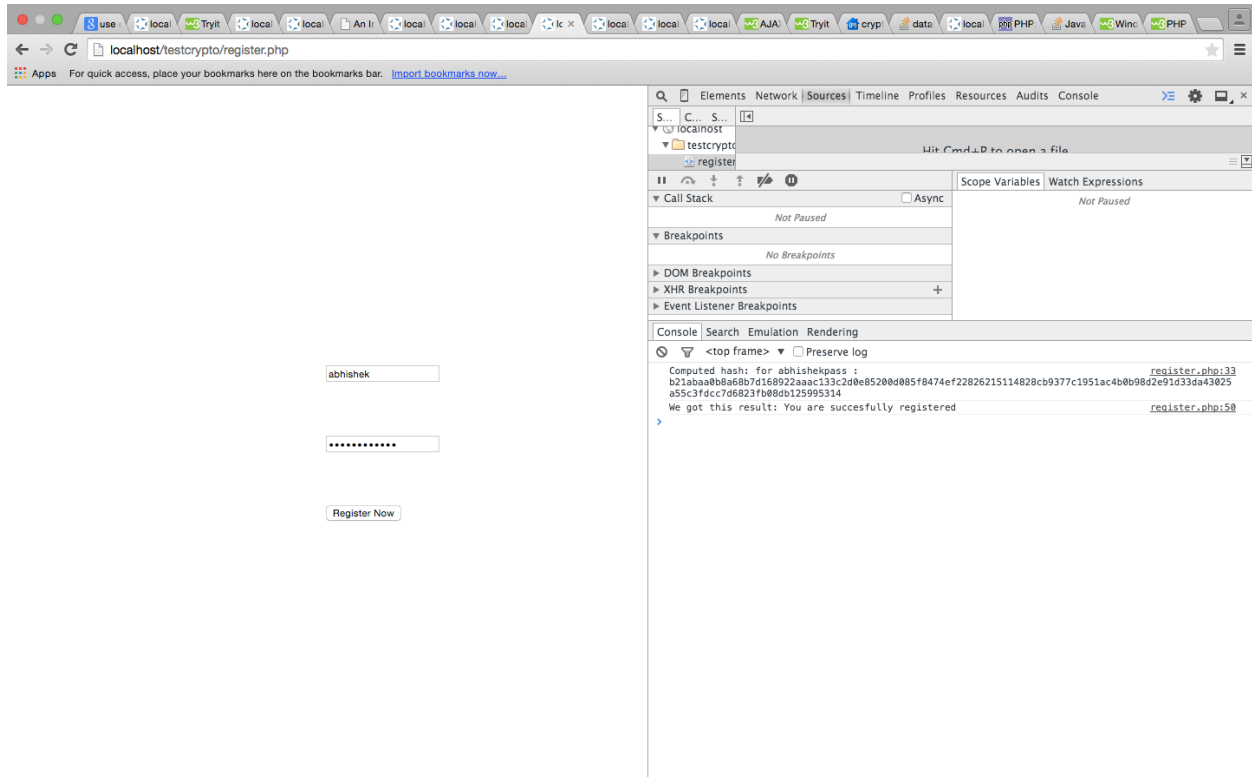
000102030405060708090a0b0c0d0e0f

43174b802454fb460ed17022ec388b1b93760970ca2447bfc2c794c330191ee12c619f45aa5d6c74b543f2614115f957b5c627d0d9359deed363e80fe0a2fe9c

use console log javascript - Google Search
```

### Registration of a new User:

Open the index.php page, and then click on register. A new page will open where you will be prompted to enter Username and password. After entering the password here, a SHA-3 hash of your password will be generated and subsequently stored in the database with the username. A file would also be created which will be used for storing the data. The database can now be queried for authentication.



Also, you can manually check the database table to see that the password hash is stored :

```

172-27-221-137:crypto_hw_project2 rajatpawar$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2199
Server version: 5.6.24-enterprise-commercial-advanced MySQL Enterprise Server - Advanced Edition (Commercial)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from fromphp;
+-----+-----+
| user | pass |
+-----+-----+
| shail | 028d13cd63fb5aecba4ee69c184241b3350c97e83cc6bf0b9c299bf7b1355da48457db94fd1e2f76b685b1e61138e9de04c77c3c894e6565318f394ec662dfed |
| shailesh | 43174b802454fb460ed17022ec388b1b93760970ca2447bfc2c794c330191ee12c619f45aa5d6c74b543f2614115f957b5c627d0d9359deed363e80fe0a2fe9c |
| abhishek | b21abaa0b8a68b7d168922aac133c2d0e85200d085f8474ef22826215114828cb9377c1951ac4b0b98d2e91d33da43025a55c3fdcc7d6823fb08db125995314 |
| jay | 80ad80061fbcd28f66240649821fabf6fc34a8953333b677587b037ff57cd1e89f6089998f49f0e35ca7d7386c5dd3e2e5c807041edef2b1a7faf36633c11b5 |
+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

Note : Look at the files createuser.php and index.php ; they both are well commented.

Also, we note that the php files parsed by the apache HTTP server are first parsed, actions taken as per the php code and then served to the user. We note here that the PHP code is NOT visible to the user. Thus, using PHP is a sensible decision. Also, PHP is designed to be more secure and robust than C itself. (A claim by one of its founders.)



## LOGGING IN USING A USER ID AND UPDATING THE BLOG

### Logging in :

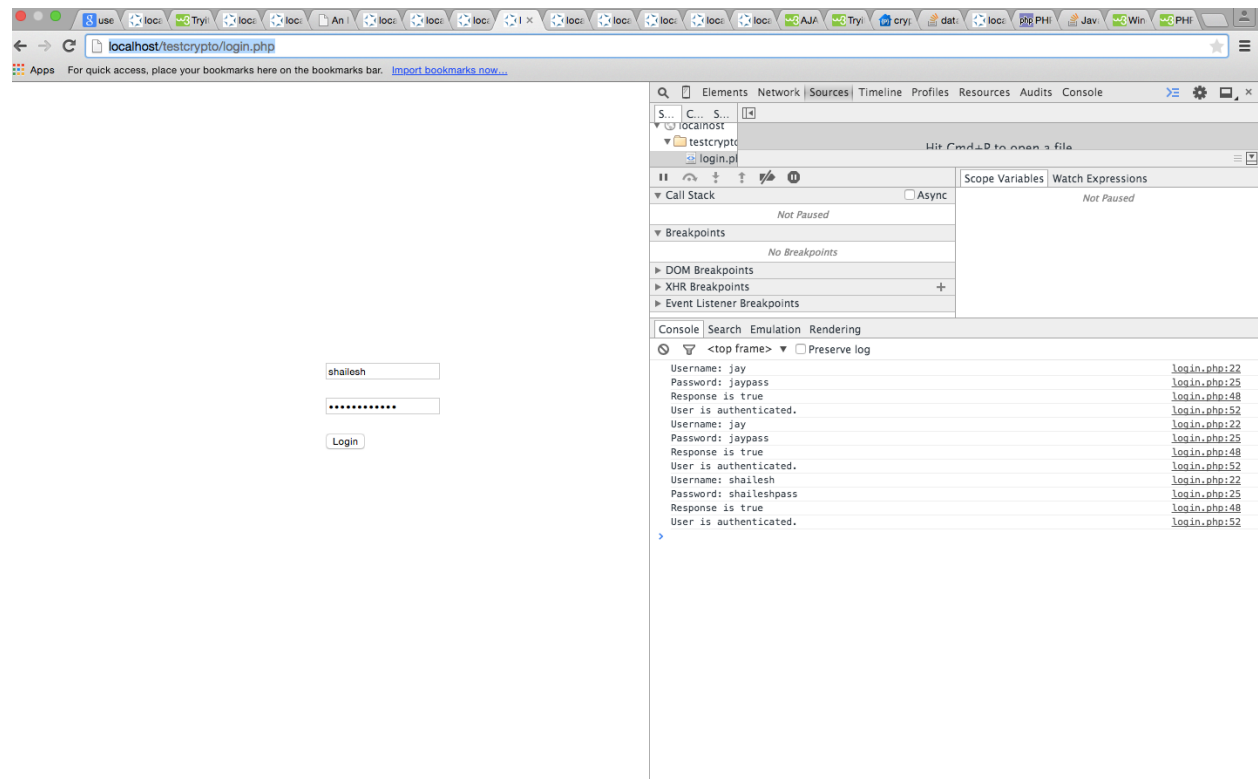
2) Then, click on the button login and a new login page will be opened. (Note: Disable popups in your browsers for this to work!) Enter the ID and the password here that you used for registration and click on login. If you are authenticated, a new window containing your blog contents will be opened.

Observe the output in the javascript console (of page login.php) : You will see the computing of the hash and its checking in the database server for authentication. This is shown in a screenshot below:

The code that allows a user to log in : login.php

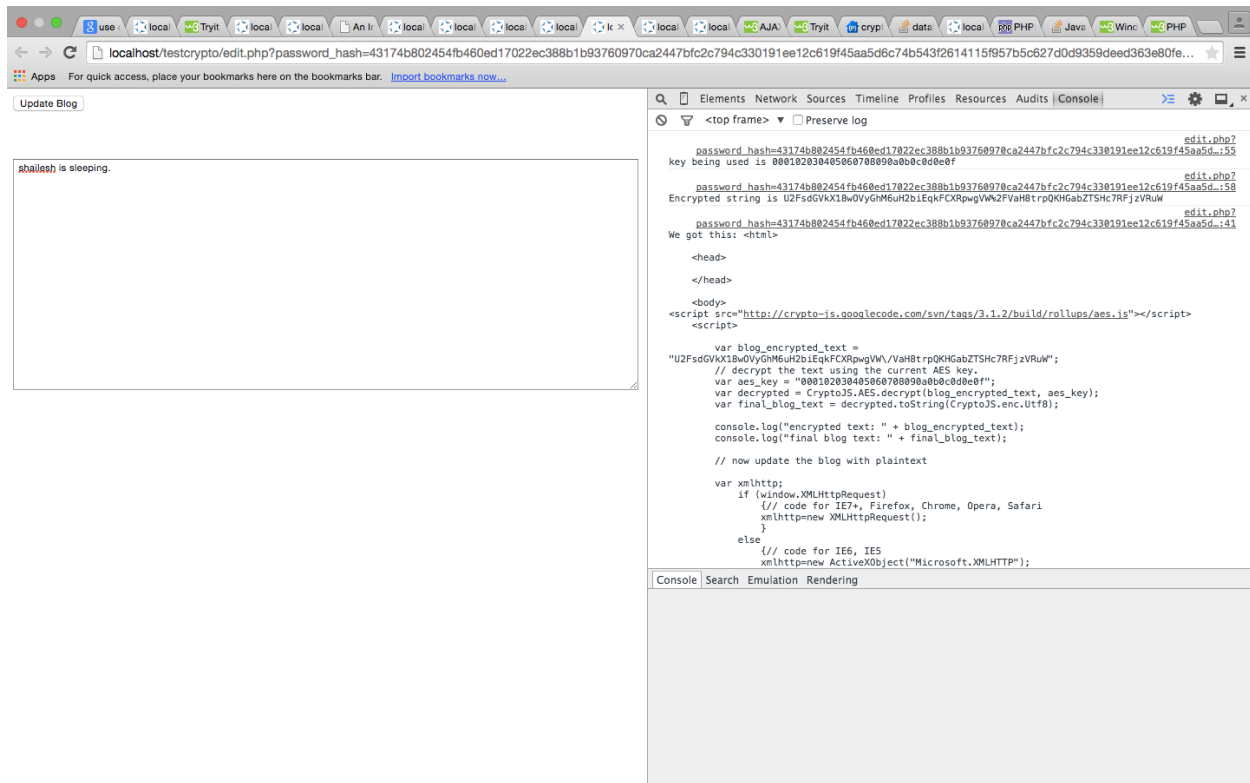
The code that authenticates a user: validatelogin.php

Please have a look at both the codes; they are well commented.



### Editing the blog :

3) Now you will see a new window opened (by clicking the login button) [at edit.php page] . The contents of the user's blog are automatically displayed here. If its a new user, empty blog is displayed. For the already registered case of shailesh,



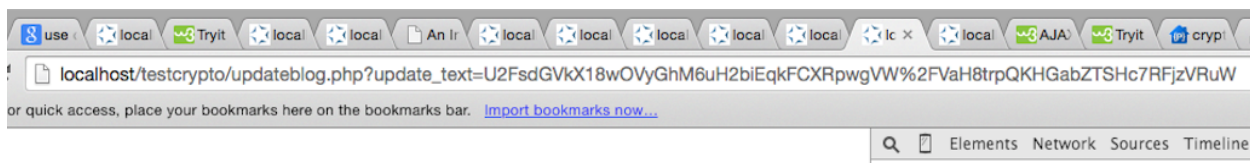
Notice the parameter “Encrypted String on the javascript console in the right.”

Here is a problem of mine. For my poor proficiency in programming web applications, I was unable to link the page of encryption and decryption. Hence I request you to test by copying the encrypted output (the content of encrypted string) of one page and supply to another page as a URL query parameter.

The page, in particular is ,

<path to my project>/testcrypto/updateblog.php?update\_text=<supply the encrypted text here>

An Example of the URL:





**ADDED LEVEL OF SECURITY :**

Note that we also implemented a C++ module that can be used to encrypt the data that we store in the server. As per the design, we plan to generate separate keys on the server and encrypt the data using that. This is an added level of security, since even if the key of client and the server-side encrypted data is compromised, the attacker may not be able to recover the plaintext. Some screenshots follow :

```

Project 2 — pawar-patel-cs69 — 107x53
172-16-188-23:project 2 Jay$ ./pawar-patel-cs6903s15project2.out
1. SHA256 calculate
2. AES encryption
3. RSA encryption
4. Exit

Enter choice to perform the action : 3
Generating RSA (2048 bits) keypair...
-----BEGIN RSA PRIVATE KEY-----
MIIEEwIBAAKCAQEArb2vvf9BpAfKqLb0ZImmVcox8R0ApVlUcahRYKnLqHZ3Iu50
RmC0iPg+wNq4hlLXsMKQyeD8dvHCgMOBKYYfEzNeId5nUFYZekzoKzLaXGG5yA61
3f8RtZrKu1sTs3RWsioTTkyqgODHmRZGgWwufTg7oF4XDxYZ6//RGkn41gBHSn
c6fnAlutWGQUpdL5BjIH3CTjCmUy+KgVpNKLWbIDi3arU3UfMEVfjQUikLHMZhbx
Lp44hgv9iUxpXz+gC7ahDYbUrP8dbZqWab2oa9dDz5g2PyRGQa7QTud29LMYN0F5
Jq03KBDZdfTH0yrzHfKYTOpCHRviDPTA4NyxKQIBAwKCAQBz08p+qivCr9xxj02Y
W8Q5MXagt6sY5jhlxYuVxofFpE9snt7Zlc2wpX7l5yWu4eUggbXb61L59oGrLQDQ
Vj9iIj80lETg0WZrIjRhdzw9lnvatHk+qgv0ZzHSPLfM+DnMHAze3ccAlpdmDtmr
nWsmo0CfAProJ0uWadVTYRhqXSWUocvJeJeIXZQszfJ93AwQBccRcW0uXYJ5r4Dz
riJAVp6tuJDCNRIj+a9r4jeRqLThkj3pQ5HdIzZ4j0VnskNkd0JcJlDas6xFjWTg
ZvDpnDvFdzP/4lqhgiMNUK06VXKZB2JzrJ7hQ8HD9CgaXC49u4SQFNs0ikBz/IJq
uBarAoGBA0ebc73JEEg2Yt5wSpjqGM7eLF5VhXhE3wTr8FR0Xl5DVHGaxX0oExmb
45kg04rTQTfgiuCHImYX05+lhPDmBqlyfvcjos45PQg1EAX1mW1BFQu5EC/Knkbc
UlqWIScbmFSxu+LaF1wrkL10yaC5KrEU/USGy2Pcr66nNdWoy5XpAoGBAMAKDjfi
YRt/bHCK5I/+9pgPlP7sdVU/+epWMH/C7vm126PsHF6IEmpeba9KXkRkk4UzMJnJ
lnejG4wu9CBdrZGciFzfQo0rtJLQnNkjidHmV95uVTh0sndzGUEkytWbMpedmoKf
EuyqnlbP0padfysAXzt+S9xnNK3xCeX/PlBAoGBAJpn06XYUV5lz71hxCcEInp
ud7jrLat6gNH9Y2i6ZQs4vZnLk0at2Zn63DAjQc3gM/rB0BaFu66J7/DrfXurxuQ
VKTCbl7f9v4tVl0ZkjWdgfQtxUxvtnoNuc0wMS9EDh2fUhmuj1yYH5N28B7cctj
U4MEh5fodR8aI+PF3Q6bAoGBAIAgtCVBlhJU8vXDQwqp+bbq1Df9ITjVVUbkIFUs
n1Ej58KdaD8FYZw+88oxLC2YYljMyxExDvpsvQgfTWrpHmETBY7qcbNyeGHJFDC
W+Ey5T700NBnzE+iENYYhz1WZw++ZwG/YfMcZvuSKJw8Tqhy6Aieph+hEzclLW+6
qKYrAoGBANwyIOlybtYuy56pc2Ta8F1v1dDuXIdEaxS4H36fUUVawmuc0HwWw2ws
TqlXjPEbgKNw0tNgndfWJ6zAss860Mrexr8MqiTSFe2mnGmA5CH0V28d3GCEc4Je
spKW3oDIcBEahVHdrcn1w45eW7ZKrrVdHYkZxyjeUe2pt5X3CB9x
-----END RSA PRIVATE KEY-----

-----BEGIN RSA PUBLIC KEY-----
MIIBCAKCAQEArb2vvf9BpAfKqLb0ZImmVcox8R0ApVlUcahRYKnLqHZ3Iu50RmC0
iPg+wNq4hlLXsMKQyeD8dvHCgMOBKYYfEzNeId5nUFYZekzoKzLaXGG5yA613f8R
tZrKu1sTs3RWsioTTkyqgODHmRZGgWwufTg7oF4XDxYZ6//RGkn41gBHSnc6fn
AlutWGQUpdL5BjIH3CTjCmUy+KgVpNKLWbIDi3arU3UfMEVfjQUikLHMZhbxLp44
hgv9iUxpXz+gC7ahDYbUrP8dbZqWab2oa9dDz5g2PyRGQa7QTud29LMYN0F5Jq03
KBDZdfTH0yrzHfKYTOpCHRviDPTA4NyxKQIBAw==
-----END RSA PUBLIC KEY-----

Message to encrypt: jay
Encrypted message written to file.
Reading back encrypted message and attempting decryption...
Decrypted message: jay
1. SHA256 calculate
2. AES encryption
3. RSA encryption
4. Exit

```

```

Project 2 — bash — 107x53
uBarAoGBA0ebc73jEeg2Yt5wSpjqGM7eLF5VhXhE3wTr8FR0XL5DVHGaxX0oExmb
4SkG04rTQTfgiuCHImYX05+lhPDmBqLYfvcjos45PQg1EAX1mW1BFQu5EC/Knkbc
ULqWIScbmFSxu+LaF1wrkL10yaC5KrEU/USGy2Pcr66nNdWoy5XpAoGBAMAKDjfi
YRt/bHCK5I/+9pgPLP7sdVU/+epWMH/C7vm126PsHF6IEmpeba9KXkRkk4UzMJnJ
lnejG4wu9CBdrZGciFZfqo0rtJLQnNkjidHmV95uVTh0sndzGUEkytwBmpedmoKf
EuyqnmnlbP0padfysXAZt+S9xnNK3xCeX/PlBAoGBAJpnon6XYUV5lz71hxCcEInp
uD7jrLAt6gNH9Y2i6ZQs4vZnLk0at2Zn63DAjQc3gM/rB0BaFu66J7/DrfXurxuQ
VKTcbIl7fgV4tvL0ZkjWDgfQtXUxvtnoNuc0wMS9EDh2fUHmuj1yYH5N28B7cctj
U4MEh5fodR8aI+PF3Q6bAoGBAIAgtCV8lhJU8vXDQwqp+bq1Df9ITjjVUUbkiFUs
n1Ej58KdaD8FYZw+88oxLC2YyljMyxExDvpsvQgftWrpHmETBY7qcbNyeGHgJFDC
W+Ey5T700NBnzE+iENYYhz1WZw++ZwG/YfMcZvuSKJw8Tqhy6Aieph+hEzclLW+6
qKYrAoGBANwyIOLybtYuy56pc2Ta8F1v1dDuXIdEaxS4H36fuUVawmuc0HwWw2ws
TqlXjPEbgKNw0tNgndfWJ6zAss860Mrexr8MqiTSFe2mnGmA5CH0V28d3GCEc4Je
spKw3oD1cBEahVHdrcn1w45eW7ZKrrVdHYKZxyjeUe2pt5X3CB9x
-----END RSA PRIVATE KEY-----

-----BEGIN RSA PUBLIC KEY-----
MIIBCAKCAQEArb2vvf9BpAfKqlb0ZImmVcox8R0ApVlUcahRYKnLqHZ3Iu50RmC0
iPg+WNq4hlLXsMKQyeD8dvHCgM0BKYYfEzNel5nUFYZekzoKzLaXGG5yA613f8R
tZrKu1sTs3RWsioTTkyqg0DHmRZGgWwgufTg7oF4XDrxYZ6//RGkn41gBHSnc6fn
AlutWGQUpdl5BjIH3CTjCmUy+KgVpNKLWbIdi3arU3UfMEVfjQUikLHMZhbXLP44
hgv9iUxpXz+gC7ahDYbUrP8dbZqWab2oa9dDz5g2PyRGQa7QTud29LMYNOf5Jq03
KBDZdfTH0yrzHfKYTOPcHRviDPTA4NyxKQIBAw==
-----END RSA PUBLIC KEY-----

Message to encrypt: jay
Encrypted message written to file.
Reading back encrypted message and attempting decryption...
Decrypted message: jay
1. SHA256 calculate
2. AES encryption
3. RSA encryption
4. Exit

Enter choice to perform the action : 1
Enter a message to find the SHA256 hash : jay
bfef4adc39f01b33fe749bb5f28f1b581fef319d34445d21a7bc63fe732fa3
1. SHA256 calculate
2. AES encryption
3. RSA encryption
4. Exit

Enter choice to perform the action : 2
Enter the file name to encrypt : JP.jpg
Encrypted file generated : JP.jpg_enc
File decrypted : JP.jpg_dec
1. SHA256 calculate
2. AES encryption
3. RSA encryption
4. Exit

Enter choice to perform the action : 4
172-16-188-23:project 2 Jay$

```