Efficient Encryption for Securing Sensitive Data of a Webpage Using Hashing

Abhilash D C

***Abstract*— The issue of security is very paramount for a website of any organization, especially for those organizations who cannot afford to pay for few secure protocols. Therefore we intend to bring in Hashing Algorithm and Encryption Algorithm which encrypts the sensitive data to n-bit code at the client's side. For example, when a user enters a username and password, the password being the sensitive data gets encrypted to n-bit code before leaving the system. By this we mean that the server receives the password as an encrypted data and not just as a plain text. We use multiple Hashing Algorithms which makes it difficult for intruders to decrypt the captured encrypted data. By this project even a small organization can give security to their websites without any cost.**

***Index Terms*— Security, Hashing Algorithms, Encryption**

I. INTRODUCTION

Increased information sharing through social networking and increasing business adoption of the web as a means of doing business and delivering service, websites are often attacked directly.

The issue of security is very paramount for a website of any organization, especially for those organizations who cannot afford to pay for few secure protocols. We are using multiple Hashing Algorithms and Encryption Algorithms to encrypt the data at the client's side which makes it difficult for intruders to decrypt the captured encrypted data.

A hash function is any function that can be used to map dataof arbitrary size to data of fixed size. The values returned by a hash function are called hash values, hashcodes, hash sums, or simply hashes.Hash functions are related to (and often confused with) checksums, check digits, fingerprints, randomization functions, error-correcting codes, and ciphers. Encryption is the process of encoding messages or information in such a way that only authorized parties can read it.The intended communication information or message, referred to as plaintext, is encrypted using an encryption algorithm, generating cipher-text that can only be read if decrypted. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm.

In order to make it more secure, we are using the combination of both hashing algorithms and asymmetric encryption algorithm.

II. PROPOSED SYSTEM

In the proposed system, we have given security to the sensitive data. The sensitive data is encrypted at the client's side using hashing algorithm and encrypting algorithm. By this, the webpage is made secure.

We are making use of two hashing algorithms and one encryption algorithm. They are

Hashing algorithms - MD5 and SHA 2
Encrypting algorithm – RSA

MD5 Algorithm (Message Digest):

- Step1 Append padding bits

The input message is "padded" (extended) so that its length (in bits) equals to 448 mod 512. Padding is always performed, even if the length of the message is already 448 mod 512.
Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448 mod 512. At least one bit and at most 512 bits are appended.

- Step2. Append length

A 64-bit representation of the length of the message is appended to the result of step1. If the length of the message is greater than 2^64, only the low-order 64 bits will be used.

The resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. The input message will have a length that is an exact multiple of 16 (32-bit) words.

- Step3. Initialize MD buffer

A four-word buffer (A, B, C, D) is used to compute the message digest. Each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):
word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10

- Step4. Process message in 16-word blocks

Four functions will be defined such that each function takes an input of three 32-bit words and produces a 32-bit word output.

F (X, Y, Z) = XY or not (X) Z
G (X, Y, Z) = XZ or Y not (Z)
H (X, Y, Z) = X xor Y xor Z
I (X, Y, Z) = Y xor (X or not (Z))

SHA 2 Algorithm (Secure Hash Algorithm):
STEP 1: Create an SHA256 object.
STEP 2: Initialize it with sha256_init().
STEP 3: Read some/all of the data to hash into an array, calculate the size of the data, and add it to the hash with sha256_update().
STEP 4: Repeat the previous step for all the data you want to hash.
Finalize and output the hash with sha256_final().
RSA Algorithm (Rivest Shamir Adleman):

Abhilash D C, Department Of CSE, SVCE Bangalore

1. Key Generation

a. Choose two distinct prime numbers p and q.

b. Find n such that n = p*q.
n will be used as the modulus for both the public and private keys.

c. Find the phi of n, $\phi(n)$

$\phi(n)=(p-1)(q-1)$.

d. Choose an e such that $1 < e < \phi(n)$, and such that e and $\phi(n)$ share no divisors other than 1 (e and $\phi(n)$ are relatively prime).
e is kept as the public key exponent.

e. Determine d (using modular arithmetic) which satisfies the congruence relation

$de \equiv 1 \pmod{\phi(n)}$.

In other words, pick d such that de - 1 can be evenly divided by (p-1)(q-1), $\phi(n)$.
This is often computed using the Extended Euclidean Algorithm, since e and $\phi(n)$ are relatively prime and d is to be the modular multiplicative inverse of e.d is kept as the private key exponent.

The public key has modulus n and the public (or encryption) exponent e. The private key has modulus n and the private (or decryption) exponent d, which is kept secret.

2. Encryption

a. Person A transmits his/her public key (modulus n and exponent e) to Person B, keeping his/her private key secret.

b. When Person B wishes to send the message "M" to Person A, he first converts M to an integer such that $0 < m < n$ by using agreed upon reversible protocol known as a padding scheme.

c. Person B computes, with Person A's public key information, the cipher-text c corresponding to

$c \equiv m^e \pmod{n}$.

d. Person B now sends message "M" in cipher-text, or c, to Person A.

Initially, we give the password as a plain text to MD5 function and a hash value of128-bit is generated. In turn, this hash value is given as a input parameter to SHA 2 function. This generates a hash value of 256-bits.
Now, the hashed value is obtained using multiple hashing algorithm which adds to the security of webpage.The hashed value is later encrypted using RSA algorithm which is an asymmetric algorithm.
Hence, the password gets stored in the database as an encrypted password rather than as a plaintext.

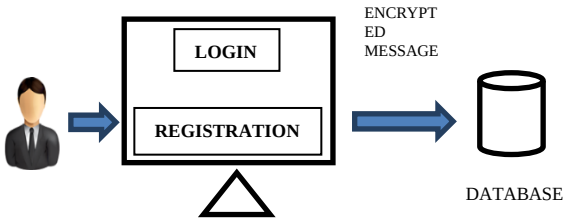

Fig 1: High-Level Design

III. IMPLEMENTATION

We have created a sample webpages which consists of a register and a login page.
REGISTER PAGE:
It consists of username, password and other personal details of the user. Once all the fields are filled, it gets stored in the database. The password gets stored as an encrypted password as explained above.

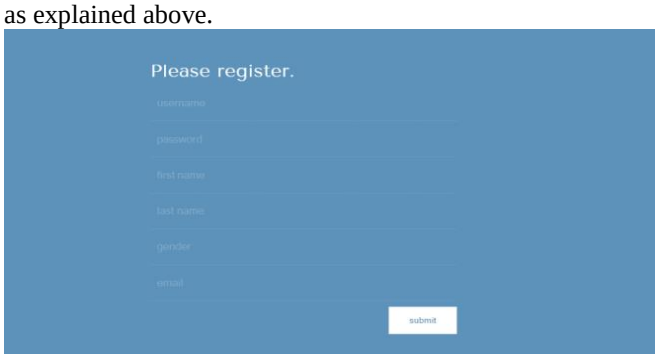

Fig 2: Registration Page

LOGIN PAGE:
It consists of username and password. Upon entering these, the password gets encrypted and is checked with appropriate password in database.

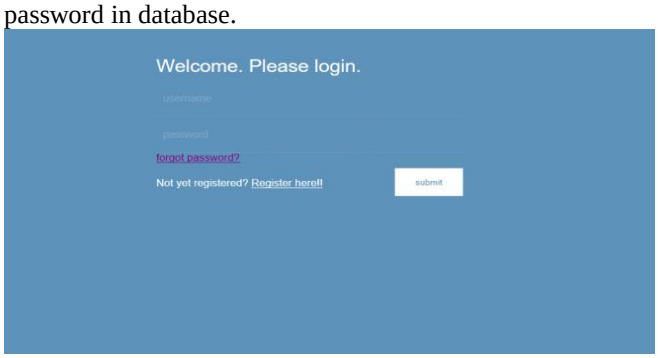

Fig 3: Login Page

DATABASE:
A screenshot of how the data is stored in the database is shown here.

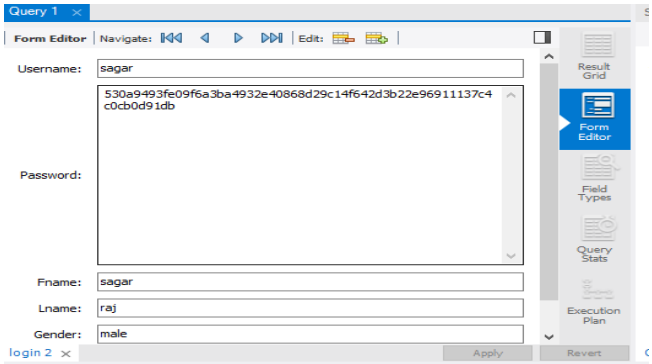

Fig 4: Database

IV. TESTING

We used wireshark, a packet analyzing tool to capture packets from the system. We noticed that the packets from HTTP website, without the implementation of project, were captured and the data was a plaintext as shown in fig 5.

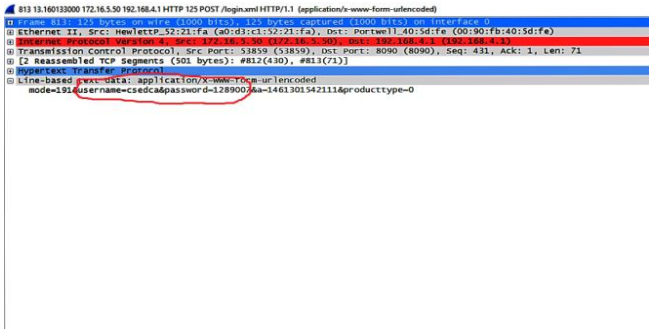

Fig 5: Packet capturing using Wireshark

Whereas, with the implementation of our project to the same website and by capturing packets, we obtained the sensitive as an encrypted text.

V. CONCLUSION

In this project, the sensitive data is secured by encrypting it using multiple hashing algorithms at the client's side. Thereby, preventing the intruders from capturing the data as a plain text and making it difficult to decrypt it.

REFERENCES

[1] Research and implementation of RSA algorithm for encryption and decryption by Xin Zhou ; Dept. of Computer Science & Technology, Harbin University of Science & Technology, Harbin, China ; Xiaofei Tang
[2] Design and optimized implementation of the SHA-2(256, 384, 512) hash algorithmsby Wanzhong Sun ; Institute of Electronic Technology, Information Engineering University, Zhengzhou 450004, China
[3] Implementation of New Modified MD5-512 bitAlgorithm for Cryptography by Priyanka Walia, Vivek Thapar.
[4] Symmetric Algorithm Survey: A comparative analysis By Mansoor Ebrahim
[5] Packet analysis using wireshark by Joseph Gehring
[6] Generalised secure hash algorithm: SHA-X by Chu-Hsing Lin, Chen-Yu Lee.