Implementation of Secured password for Web applications using two server model

Dr.R.Shashikumar C.N.Vijay kumr H.M.Kotresh Vijay kumar K
SJCIT, E&C dept
Chikballapur, Karnataka, India
SJCIT, E&C dept
Chikballapur, Karnataka, India

***Abstract* — The secured password is the most commonly used authentication mechanism in security applications [11]. There may be chances of password hacking from the hackers, so that it is very essential to protect password information while sending request to the servers. Early Web Application used central database as the password authentication scheme. It has been one of the biggest challenges in deploying sharing password authenticated key exchange solutions in practice using multiple servers. Several multi server schemes have been proposed for authentification. This work emphasis on shared secured password for web application using the two server model. Here this system will overcome all the previously proposed single and multi server model authentication systems. This new system is being developed as secured password for web application viz, authentication in VoIP (voice over internet protocol) services, PDA devices ete.**

Keywords- Password system, password verification data (PVD), user authentication, key exchange, offline dictionary attack.

I. INTRODUCTION

Password systems are normally built over the following three types of architectures

- Single-server model.
- Plain multiserver model.
- Gateway augmented multiserver model.

The first type is the single-server model given in Fig. 1, where a single server is involved and it keeps a database of user passwords. Most of the existing password systems follow this single-server model, but the single server results in a single point of vulnerability in terms of offline dictionary attacks against the user password database.

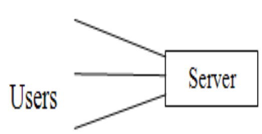

Figure 1.

The second type is the plain multiserver model depicted in Fig.2, in which the server side comprises multiple servers for the purpose of removing the single point of vulnerability; the servers are equally exposed to users and a user has to communicate in parallel with several or all servers for authentication. Clearly, the main problem with the plain multiserver model is the demand on communication bandwidth and the need for synchronization at the user side since a user has to engage in simultaneous communications with multiple servers. This may cause problems to resource-constrained mobile devices such as hand phones and PDAs. The systems in [4], [5], [8] and one of the two protocols in [9] assume this model.

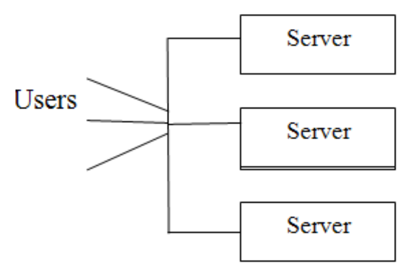

Figure 2.

The third type is the gateway augmented multiserver model shown in Fig. 3, where a gateway is positioned as a relaying point between users and servers and a user only needs to contact the gateway. Apparently, the introduction of the gateway removes the demand of simultaneous communications by a user with multiple servers as in the plain multiserver model. However, the gateway introduces an additional layer in the architecture, which appears "redundant" since the purpose of the gateway is simply to relay messages between users and servers, and it does not in any way involve in service provision, authentication, and other security enforcements. From security perspective, more components generally imply more points of vulnerabilities. Protocols based on the gateway augmented multiserver model include [6] and [9].

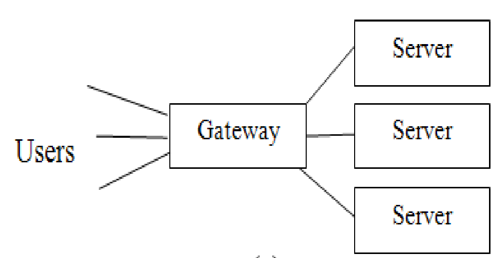

Figure 3.

II. TWO-SERVER MODEL

Two-server model [7] comprises two servers at the server side, one of which is a public server exposing itself to users and the other of which is a back-end server staying behind the scene. u sers contact only the public server, but the two servers work together to authenticate users

Basic two-server model to an architecture where a single control server supporting multiple service servers. In such an architecture, the control server and the service servers are managed in different administrative domains, and the domain where the control server resides enforces more stringent security measurements. The two server's model is as shown in Fig.4

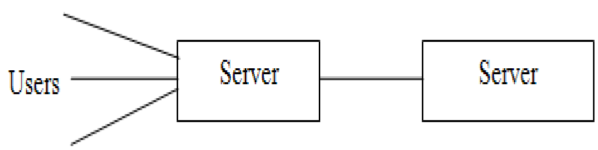

Figure 4.

III. MODEL DESCRIPTION

Three types of entities are involved in this system, i.e., users (U), a service server (SS) that is the public server in the two server model, and a control server (CS) that is the back-end server. In this setting, users only communicate with SS and do not necessarily know CS. For the purpose of user authentication, a user U has a password which is transformed into two long secrets, which are held by SS and CS, respectively. Based on their respective shares, SS and CS together validate users during user login. We assume the following security model: CS is controlled by a passive adversary and SS is controlled by an active adversary in terms of offline dictionary attacks to user passwords, but they do not collude (otherwise, it equates the single- server model). By definition a passive adversary follows honest-but-curious behavior, that is, it honestly executes the protocol according to the protocol specification and does not modify data, but it eavesdrops on communication channels, collects protocol transcripts and tries to derive user passwords from the transcripts; moreover, when an passive adversary controls a server, it knows all internal states of knowledge known to the server, including its private key (if any) and the shares of user passwords. In contrast, an active adversary can act arbitrarily in order to uncover user passwords. Besides, we assume a secret communication channel between SS and CS for this basic protocol. This security model exploits the different levels of trust upon the two servers. This clearly holds with respect to outside attackers. As far as inside attackers are concerned, justifications come from our application and generalization of the system to the architecture of a single control server supporting multiple service servers, where the control server affords and deserves enforcing more stringent security measurements against inside attackers.

IV. MODULES OF TWO SERVER MODEL

It consists of three modules

- User Registration
- User Authentication using two servers
- Session creation using Key exchange

The Symbols used in this paper has summarized in the fallowing table.1

TABLE.1

SS	Service Server name.
CS	Control Server name
π	Hashed encrypted password.
π1	First half of the Hashed encrypted password
π2	Second half of the Hashed encrypted password
R Z_q	Random number in long integer
U	Request message generated by user
b1	Random number generated at SS.
B1	Combination of b1 and encrypted half password of SS (π1).
B2	Combination of b2 and encrypted half password of CS (π2).
B	Combination of both B1 and B2 .
A	Random number generated at User.
Su	Authentication Key of the User.
S1	SS Authentication Key.
S2	CS Authentication Key.
k	Session Key of user.
g1, g2	random numbers.
q	Number of bits used in algorithm.
Ss	Service server session key.
Q,p, q	Three large primes such that Q=2p+1 and p=2q+1
$\in_R$	Belongs to real part of
QR_p	Quadratic residues
g1,g 2	g1,g2 $\in QR_p$ are of order q and discrete logarithms to each other are not known ,where QR_p is the group of quadratic residues of p
g3	g1,g2 $\in QR_q$ is of order q
h(.)	A Cryptographic hash Function

A. *User Registration*

In any password system, to enroll as a legitimate user in a service, a user must beforehand register with the service provider by establishing a shared password with the provider. U needs to register not only to the service provider SS but also to the control server CS. Let us suppose U has already successfully identified himself to SS, e.g., by showing his identification card, U splits his password π into two long random numbers **π1 ЄR Zq** and **π2 ЄR Zq** such that **π 1 + π 2 = π.(mod q),** where q is defined in Table 1. U then registers in a secure manner **π1**and **π2** to SS and CS, respectively. SS stores the account (U, **π1**) to its secret database, and CS stores (U, **π2**) to its secret Database. In case CS supports multiple servers, it stores (U, **π2**, SS) to distinguish users associated with different servers. This completes the user registration phase. One may wonder how U registers π 2 to CS as CS is supposed hidden from U. This actually is not a problem in practice: U can reach CS through out-of-band channels. Figure.5 shows the flowchart for user registration procedure.

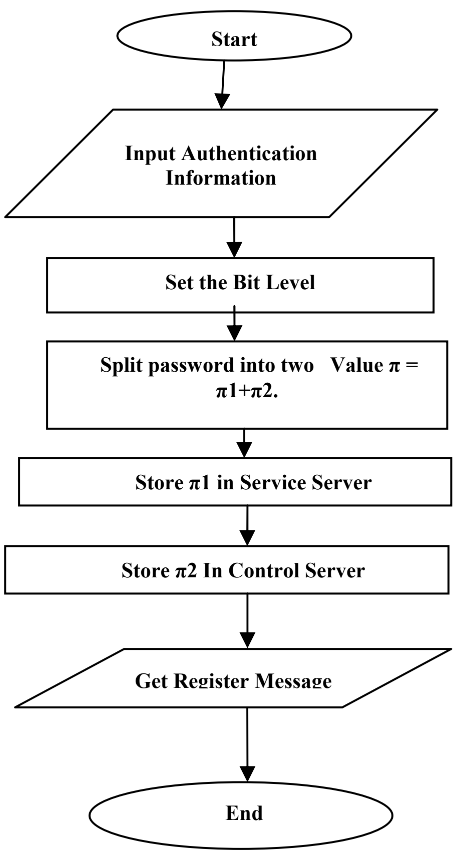

Figure.5

B. *User authentication*

Figure.6 shows the flowchart of procedures in User authentication.

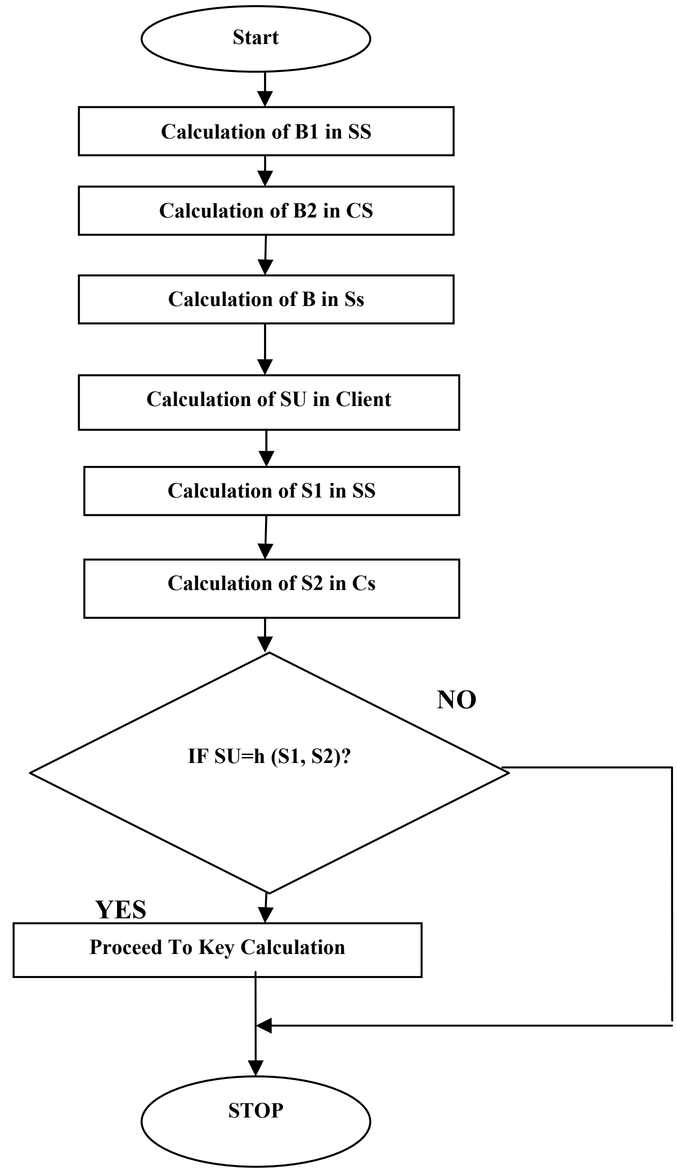

Figure.6

- U sends his identity together with a service request Req to SS.
- SS first relays the request to CS by sending the user ID and then selects a random number $\mathbf{b_1}$ **Є**$_\mathbf{R}$ $\mathbf{Z_q}$ and computes $\mathbf{B_1 = g_1^{b1} g_2^{\pi 1} (mod\ p)}$**,** using his password share $\boldsymbol{\pi_1}$.
- CS chooses a random number $\mathbf{b_2}$ **Є**$_\mathbf{R}$ $\mathbf{Z_q}$ and computes $\mathbf{B_2 = g_1^{b2} g_2^{\pi 2}}$**(mod p**) using his password share $\boldsymbol{\pi_2}$.

- CS then sends $\mathbf{B_2}$ to SS. Upon reception of $\mathbf{B_2}$, SS computes and sends $\mathbf{B = B_1\ B_2\ (mod\ p)}$,
- U selects $\alpha \in_R Z_q$, and computes $\mathbf{A = g_1^{\alpha}\ (mod\ p)}$,
- $S'_u = (B/\ g_2^{\pi}) = g_1^{\alpha\ (b1+b2)}\ (mod\ p)$ and $S_u = h(S'_u)$, respectively.
- U then sends **A** and $\mathbf{S_u}$ to SS. Getting the message, SS computes
- $\mathbf{S_1 = A^{b1}\ (mod\ p)}$ and sends $\mathbf{S_1}$, **A** and $\mathbf{S_u}$ to CS .
- Upon receipt of $\mathbf{S_1}$, CS computes $\mathbf{S_2 = A^{b2}\ (mod\ p)}$ and checks
- Whether Su **(?)= h** $(S_1S_2) = h\ (g_1^{\alpha\ (b1+b2)})$: If it holds, CS is assured of the authenticity of U

C. Key exchange

Figure.7 shows the flowchart of procedures involved in key exchange.

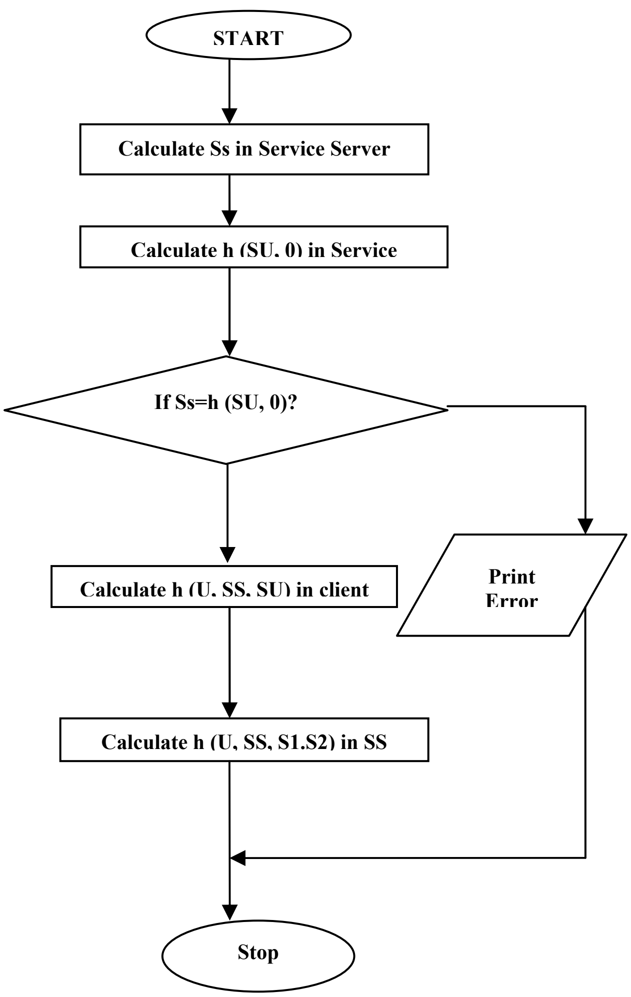

Figure.7

- After authentication Service server SS receives $\mathbf{S_2}$,Which is calculated in authentication phase it checks whether $\mathbf{S_u}$ **(?)= h (S1S2)**. If it holds, SS is convinced of the authenticity of U. At this stage, both servers have authenticated U.
- SS then computes and sends $\mathbf{S_s}$ **= h (0,S1S2)** to U and afterward computes a session key **K = h(U,SS,S1S2)**; otherwise, SS aborts the protocol.
- Upon receiving **K**, U checks if **h (0, S'u) (?)=** $\mathbf{S_s}$. If it holds, U has validated the servers and then computes a session key $\mathbf{K = h\ (U,\ SS,\ S'_u)}$; otherwise, U aborts the protocol

V. ALGORITHMS

A. Algorithm used for user Registration

SHA (Secure Hash Algorithm) refers to a family of NIST-approved cryptographic hash functions. The most commonly used hash function from the SHA family is SHA-1 [9]. It is used in many applications and protocols that require secure and authenticated communications. SHA-1 Algorithm is used for user registration. Most secure hash functions are based on the structure proposed by Merkle as shown in figure.8.

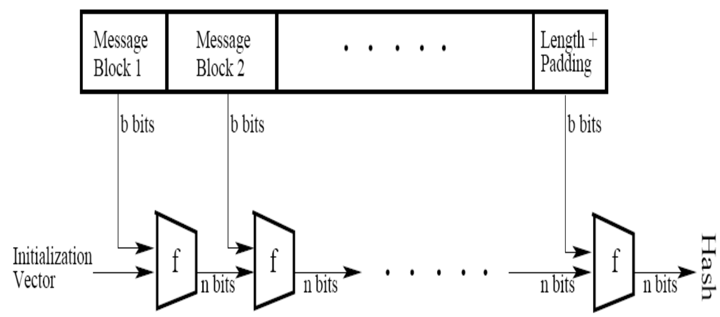

Figure.8

Figure.8 consists of L stages of processing, each stage processing one of the b-bit blocks of the input message. Each stage of the structure in Figure takes two inputs, the b bit block of the input message meant for that stage and the n-bit output of the previous stage. For the n-bit input, the first stage is supplied with a special N-bit pattern called the Initialization Vector (IV). The function f that processes the two inputs, one n bits long and the other b bits long, to produce an n bit output is usually called the compression function. That is because, usually, b > n, so the output of the f function is shorter than the length of the input message segment. The function f itself may involve multiple rounds of processing of the two inputs to produce an output.

B. Algorithm used for Encryption and Decryption

Blowfish algorithm is used for encryption and decryption [6]. Blowfish has a 64-bit block size and a key length of anywhere from 32 bits to 448 bits (32-448 bits in steps of 8 bits; default 128 bits). The computation diagram is as shown in figure.9. It is a 16-round Feistel cipher and uses large keydependent S-boxes. Initialize the P-array and S-boxes XOR P-array with the key bits. For example, P1 XOR (first 32 bits of key), P2 XOR (second 32 bits of key),...

Encrypt the new P1 and P2 with the modified subkeys This new output is now P3 and P4 repeat 521 times in order to calculate new subkeys for the P-array and the four S-boxes

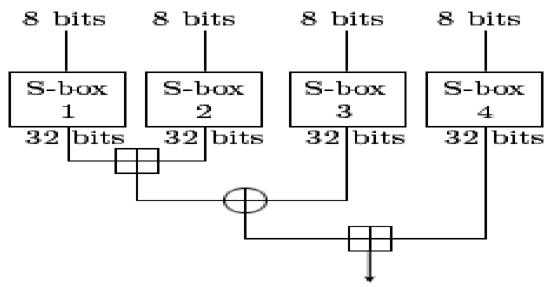

Figure.9

B. Algorithm used for Key exchange

The Diffie-Hellman key agreement protocol (also called exponential key agreement) is used for Key exchange[10].

The protocol has two system parameters *p* and *g*. They are both public and may be used by all the users in a system. Parameter *p* is a prime number and parameter *g* (usually called a generator) is an integer less than *p*, with the following property: for every number *n* between 1 and *p*-1 inclusive, there is a power *k* of *g* such that $n = g^k \bmod p$.

Suppose A and B want to agree on a shared secret key using the Diffie-Hellman key agreement protocol. They proceed as follows: First, A generates a random private value *a* and B generates a random private value *b*. Both *a* and *b* are drawn from the set of integers. Then they derive their public values using parameters *p* and *g* and their private values. A's public value is $g^a \bmod p$ and B's public value is $g^b \bmod p$. They then exchange their public values. Finally, A computes $g^{ab} = (g^b)^a \bmod p$, and B computes $g^{ba} = (g^a)^b \bmod p$. Since $g^{ab} = g^{ba} = k$, A and B now have a shared secret key *k*.

VI. STEPS USED FOR IMPLEMENTATION

Following Are the Important steps in implementing this project and it is shown in figure.10.

A FRONT END(HTML,JSP)

- HTML is used only for designing the static page
- JSP are same as the html but are dynamic and execution is same as servlets but saves the time in calling the separate servlets for each time.
- JavaScript is for client side calculation purpose. For example, calculation of hashed value of the password used before sending it to servlet

B. MIDDLE TIER (SERVLETS) SERVER SIDE SCRIPTING (CONTROLLER)

HTML, JSP pages will call servlets after clicking the submit button

- All the business logic is programmed in servlet programs
- All the responses for which the client will get is from the servlet program
- Servlet may return text message or give response as the HTML page and all database transaction logic is written in the servlet program itself

C. DATABASE USED (BACK END)

- MYSQL 5.5 and MSSQL 2005 SERVER EDITION
- All the values are stored in respective database.

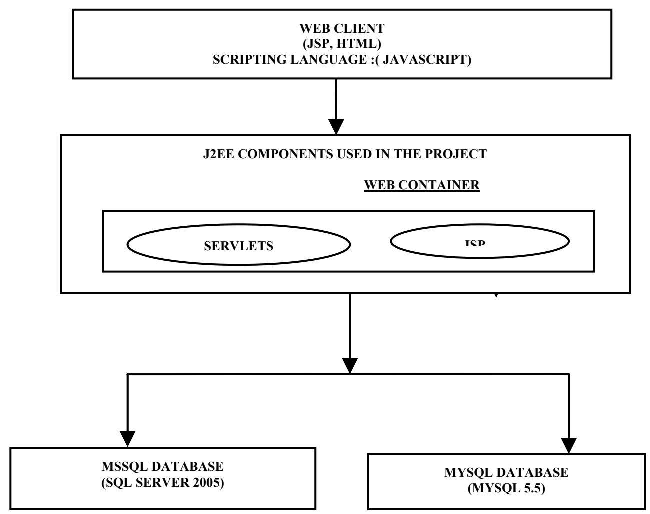

Figure.10

VII. IMPLEMENTATION OF THE ALGORITHMS

In this project Implementation is being done as three modules

A User registration

In this phase user(U) gives user name and password for authentication using these parameters. User side module calculates as shown below
π = calcSHA1 (u + ":" + p);
CalcSHA1 uses function str2blks_SHA1 (str) above function convert a string to a sequence of 16-word blocks, stored as an array. Append padding bits and the length. And function calcSHA1Blks (str2blks_SHA1 (str)) take a string and return the hex representation of its SHA-1. The pseudo code for SHA1 algorithm as follows.

Initialize variables: h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476

h4 = 0xC3D2E1F0

Break message into 512-bit chunks. Each chunk breaks into sixteen 32-bit big-endian words w[i], $0 \le i \le 15$ Extend the sixteen 32-bit words into eighty 32-bit words: for i from 16 to 79 w[i] = (w[i-3] XOR w[i-8] XOR w[i-14] XOR w[i-16]) leftrotate 1.

```
Initialize hash value for this chunk:
   a = h0;
   b = h1;
   c = h2;
   d = h3;
   e = h4;

 Main loop:
   for i from 0 to 79
       if 0 ≤ i ≤ 19 then
          f = (b and c) or ((not b) and d)
          k = 0x5A827999
       else if 20 ≤ i ≤ 39
          f = b xor c xor d
          k = 0x6ED9EBA1
       else if 40 ≤ i ≤ 59
          f = (b and c) or (b and d) or (c and d)
          k = 0x8F1BBCDC
       else if 60 ≤ i ≤ 79
          f = b xor c xor d
          k = 0xCA62C1D6
         temp = (a leftrotate 5) + f + e + k + w[i]
       e = d
       d = c
       c = b leftrotate 30
       b = a
       a = temp
Add this chunk's as shown below
   h0 = h0 + a
   h1 = h1 + b
   h2 = h2 + c
   h3 = h3 + d
   h4 = h4 + e
```

It produces the final hash value hash.

hash = h0 append h1 append h2 append h3 append h4

User module splits π into $\pi1$ and $\pi2$ in such a way that $\pi = \pi1 + \pi2$, $\pi1$ and $\pi2$ are the variables of type Big Integer.

A cryptographically strong random number minimally complies with the statistical random number generator tests specified in FIPS 140-2, Security Requirements for Cryptographic Modules. Additionally, Secure Random must produce non-deterministic output. Therefore any seed material passed to a Secure Random object must be unpredictable, and all Secure Random output sequences must be cryptographically strong, as described in RFC 1750: Randomness Recommendations for Security. Using π and $\pi1$ easy to calculate $\pi2$ as $\boldsymbol{\pi2 = \pi - \pi1}$. Next step is to register splitted passwords in to the two servers databases

B. User Authentication

For authentication splitted password $\pi1$ and $\pi2$ are used. $\pi1$ is send to the Service server and $\pi2$ is send to the Control Server. The Service server selects the random prime numbers g1, g2 and bl, using these variables and stored password $\pi1$, it calculates the value B1

$B1=g_1^{b1} . g_2^{\pi 1}$

Control Server is also calculates B2 using probable Prime functionality available in BigInteger class.

Along with these variables and store password $\pi2$ Server Calculates the B2 value

$B2=g1^{b2} . g2^{\pi 2}$.

After calculated B2, Control Server sends back B2 value to the Service Server. Service Server calculates B in such a way that B=B1B2

After calculated A and Su these values are passed to the Service Server SS, then SS computes

$S1=A^{b1}$ and sends {A, Su, S1} to CS.

Control Server CS also calculates S2=Ab2 and also SS calculates h(S1,S2) the resultant value is compared with Su if both values are same then only user successfully authenticated himself to the server otherwise entire process will be aborted.

C. Key exchange

- **Client side key calculation:**

```
KSs = calcSHA1 (u + "," + temp_s1s2 + "," + temp_Ss);
Var temp_Hc = bigInt2Str (sup);
Var temp_z = String (0);
Hc = calcSHA1 (temp_z + ","+temp_Hc);
Alert ("Hc="+Hc); var temp_HSs = str2BigInt (HSs);
Var temp_Hc = str2BigInt (Hc);
if (temp_HSs.subtract(temp_Hc)==0)
 {
 alert ("Authentication success between SS hash and
Client
 hash");
}
else
 {
 alert ("Authentication Failed"); } var temp_Sup =
         bigInt2Str(sup); var temp_Ss =
HSs; Server side key calculation
```

- **Server side key calculation:**

```
KCs =calcSHA1 (u + "," + temp_Sup + "," +
temp_Ss); alert ("KCs"+KCs);
```

VIII. RESULT

The home page for registration is as shown in figure.11

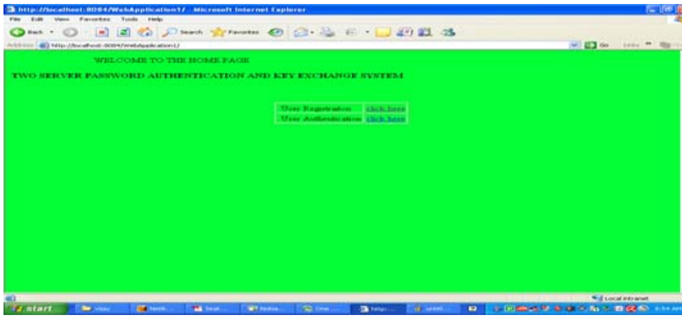

Figure.11

In Home page we get two options to click

- User Registration
- User Authentication page

The registration page is as shown in figure.12

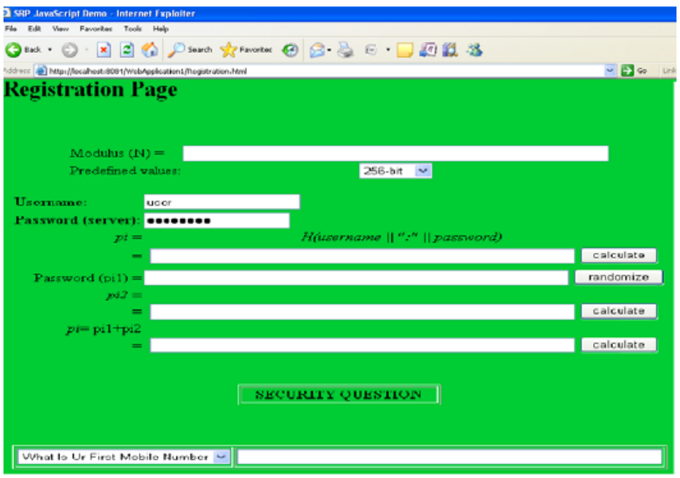

Figure.12

In Registration Page, user has to enter User Name and Password, the private Key and hashing of password are automatically executed by the JavaScript. User going to enter the security question's answer in order to protect users from online and offline dictionary attacks. If the registration is successful, it displays as shown in figure.13 and if it not successful, it displays as shown in figure.14

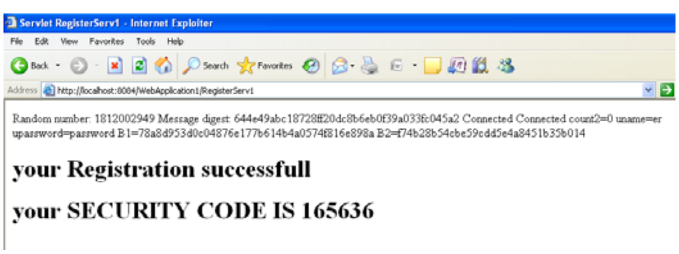

Figure.13

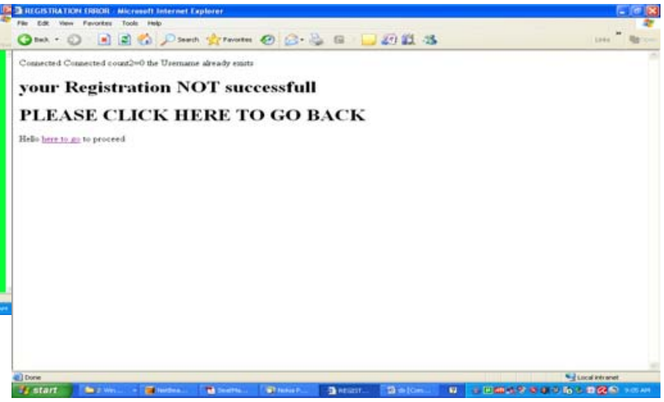

Figure.14

For authentication, user has to enter user name and password as shown in figure.15. All other values are automatically generated.

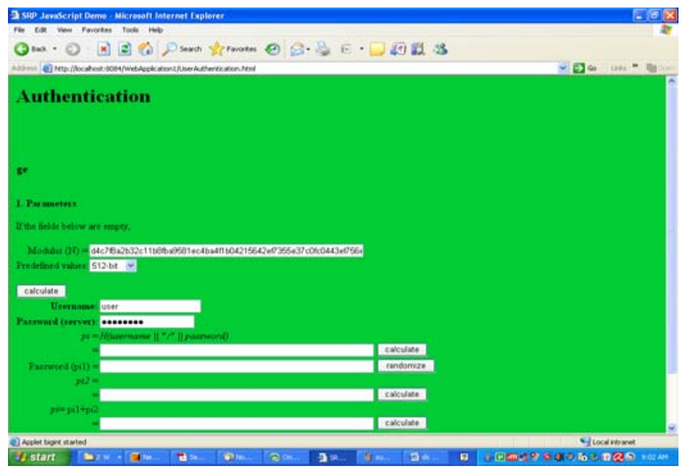

Figure.15

For protection from Brute force or online attack or if the user enters his name more than two times, the security question is raised and he is not going to be authenticated until user answers the security question correctly as shown in figure.16

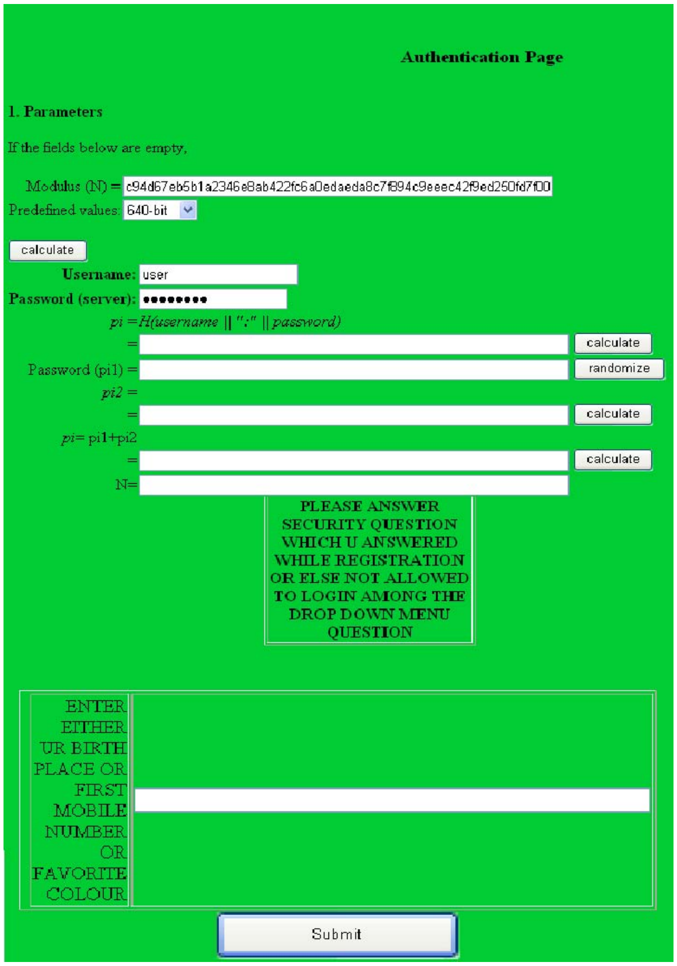

Figure.16

Once the authentication process is over, the below screen will appear for further calculations as shown in figure.17 and calculation is as shown in figure.18

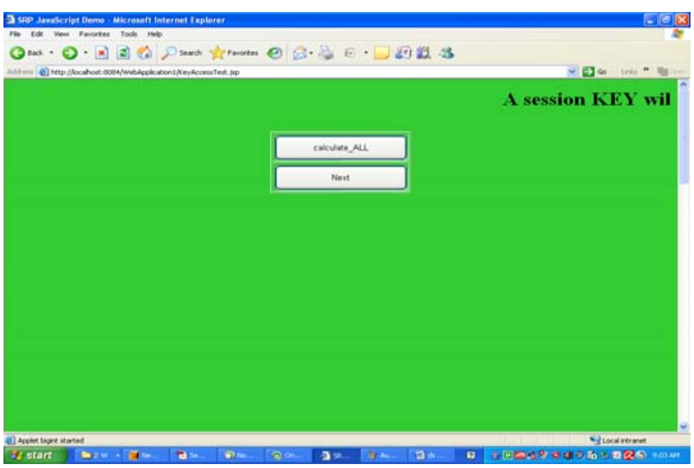

Figure.17

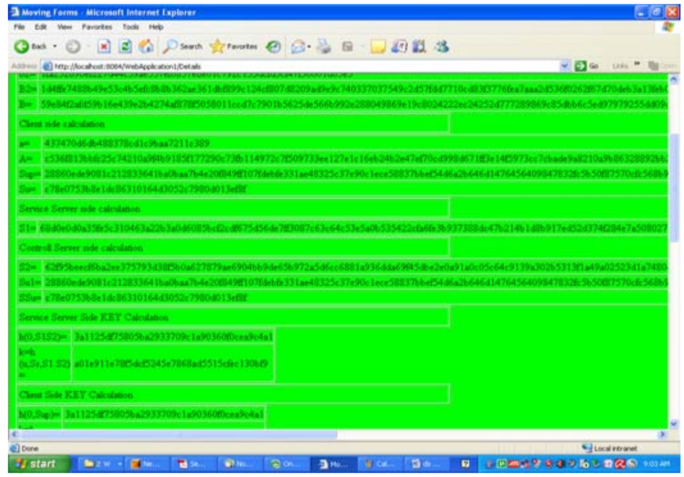

Figure.18

After User authentication, a unique session ID is generated that involves authentication by both the servers as shown in figure.19

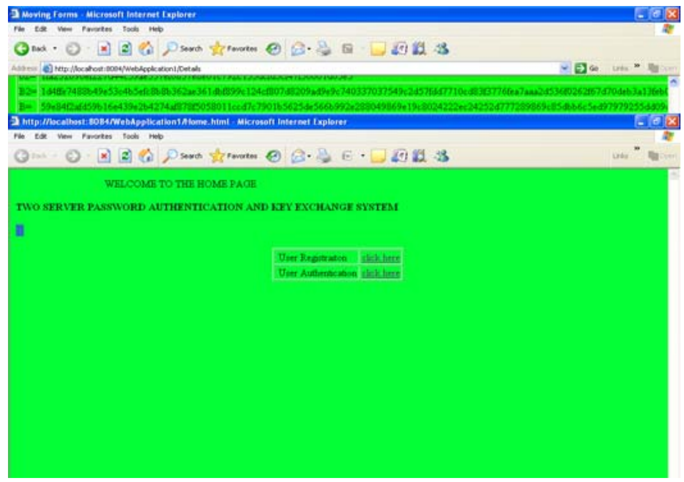

Figure.19

Resultant values for all intermediate variables as shown below.

N=eeaf0ab9adb38dd69c33f80afa8fc5e86072618775ff3c0b9ea2314c9c25576 d674df7496ea81d3383b4813d692c6e0e0d5d8e250b98be48e4 95c1d6089dad15dc7d7b46154d6b6ce8ef4ad69b15d4982559b 297bcf1885c529f566660e57ec68edbc3c05726cc02fd4cbf4976 eaa9afd5138fe8376435b9fc61d2fc0eb06e3

User name = Vijay

Password = abc$123

π = h (U: P) = 3f5eae7cca48f2c05f1c36cbee727637e7af4d86

π1 = c16c951e1b6ec7049c8c8a7b2f67a0fc

π2 = 3f5eae7c08dc5da243ad6fc751e5ebbcb847ac8a

b1=a4ca2457261b1336d43ad5af18807c93d1dca2f9e15ed02d6a530186 e00131d3

b2=7f7d8d1b6d0bbdd18d174db4b594780e6eb06c96649f01beb9bc6bde1 a80638a

g1=49d8b06e48603e8e93073d375c2641ee099111ae7b628f4fdbc5faaa7b0cefc8

g2=5790fa312f5b1ab5627dc6ef5df614a8feaae3549a88c042ce

132bb7c8 f6b2c9
B1=3f6249c383bb53d09e0e25b830dfd7fe09596d2f2c5cbd9c71b62fc8c156e23b7a1650425755f9045fa1016710c7e7c751d27264093a06eecbc77eca0c6348d2616e0a5b911c7451c048d0e4016a499a96d78f892cbec7daaf80dbdb8aa15ba18b64603eeab9ad96bf0369dcb8f7c7c0ee6f70 bc63f0a640167a1f214b88b8bb
B2=3c86bd83c584c74502a5be0cd951de4131232eb79d56c21be23835f095e16ad3c0ea8855e3993c1deb504a19e0f9082156e093daf635d32e4b3f02f8a95c3f432024a7e9c3024c186cbf3f88806530d1789b6c5850dc04eda2ee09070b3aecde7086bc05b91584b5124e9d7bc173282e8ea6a59f980dd25307a984cbc07e0b4f
B=B1*B2(modN)=52208fa4cc7a9ffb91f8f48b42c0391f55064b13da971a72fcf5184de56e78e244667aa4842ca0705f7245f2e32425a9824d6802ec1497771f4884d5435116f031b5c1a573e1c0ade49354335e3646a20e17b410dbc62f6d9a1ec63adf02628aea508b9a6123dea21d9f85fb64ac0c91cfa2c90152359ca47bf28ba908d7c2ee
a = 12cb716fd3a2ff6f1912e6f349a8cc2
A=g1^a=eab26052b3cef945dccd2ac251dfeabb6e950c7d76f7ebfeef8c173e2e7969268a5383f07431483288a3747e97a34e3c3ebac4e2ae7683f5ba90159ab25ddb4db4d64870a840bddde3c3bcd4e06ff90817b5e05b711a8ee051af044e2be5fed8f82d7108a6f109dead4d0d624c794d3136d888aa967a088d22726275e3006433
Su1=g1^(a(b1+b2))=26aefe2d5823f0d29428973a967530b0c1fdfba8c5bfbbd137a74cdb1f47ff2ac838955523f66817c2a02105504d8599b85d48daac553d5aaaddf09f2bb90c3718756d71bbf964063595ab45abc3d326747adc31ab97f569cbeda31001d33f1b08740ba8b13924fa027801cdeba3eb38f0716a1df83ab196fd27149a65ce4026dbf5ab2f54514e0336e4a094922da1695903be012948a1e6781cb6f57f519792c2d625cf7604e22fb2aa698883184275409f71b7db1a3921a722b3e0bb4e4756315b149e9e3db8e4c9bd53d6bb1befe2655c04e4e5ae3fd096a3e794068c3be4c3f355c94af84a206537c07826b01a845c384175e3998dc3a76669117febadbc
Su=h(g1a(b1+b2))=6345623bdedb15004bf079b33d7b67ad1c7862b6
S1=A^b1=38d8c7145d94def0c9e13024b90410fd0e1251db181a0985689654abf2752843332ffadcd170530fae4bbd7df6b70e102be89741b9b803cdd7eafa12ad3694b2656b0925943415e355e6ed40318e9ca357a506daaf4d7f1c55e57515d8788bbe86f1fc15a3aa5ac14da6c3130f327d415747064b70f961f78b3ee3a58d24c91
S2=A^b2=ae3478d5ae568c153d13e51b9f0f90b1775ccd05050e6cbeb3820437f88374e7103ac29624f26568af7cff31cd1ceb727b6af9e26c227d6b77e9ebb3be9306278f46cff9e341e1c1826b3bee19b25a1671b26bc8240ecbc826f333c5aa2ba52ddac1affb5ad7d0a05e864edcd725eb013227f73499b9b406cdfec3046fd6dffc
S1.S2(modN)=26aefe2d5823f0d29428973a967530b0c1fdfba8c5bfbbd137a74cdb1f47ff2ac838955523f66817c2a02105504d8599b85d48daac553d5aaaddf09f2bb90c3718756d71bbf964063595ab45abc3d326747adc31ab97f569cbeda31001d33f1b08740ba8b13924fa027801cdeba3eb38f0716a1df83ab196fd27149a65ce4026dbf5ab2f54514e0336e4a094922da1695903be0129481e6781cb6f57f519792c2d625cf7604e22fb2aa698883184275409f71b7db1a3921a722b3e0bb4e4756315b149e9e3db8e4c9bd53d6bb1befe2655c04e4e5ae3fd096a3e794068c3be4c3f355c94af84a206537c07826b01a845c384175e3998dc3a76669117febadbc
Su=h(S1.S2)
=6345623bdedb15004bf079b33d7b67ad1c7862b6
SS=h(0,S1S2)
=3e3b099368878c7be1249de624294d4f7b8d9293
K=h(U,SS,S1S2)
= e4be9956e091ab007c155fa182ef0d43e579453b
H (0, Su1) = 3e3b099368878c7be1249de624294d4f7b8d9293
K=h(U,SS,Su1)=e4be9956e091ab007c155fa182ef0d43e579453b

IX. CONCLUSION

In contrast to existing multiserver password systems, our system has great potential for practical applications. It can be directly applied to fortify existing standard single-server Password applications, e.g., FTP and Web applications. It can also be applied in the federated enterprise setting, where a single control server supports multiple service servers. also be applied in the federated enterprise setting, where a single control server supports multiple service servers.

REFERENCES

[1] Ivan Bayross, "java Server Programming", 2007 Edition, Shroff publication.

[2] William Stallings, "Cryptography and Network Security", Third edition, Pearson publication.

[3] Tom Negrino and Dori smith, "Java Script for The World Wide Web", second edition ,Pearson publication.

[4] Roger S Pressman, "Software Engineering", McGraw Hill Inc., 3rd Edition ,McGraw Hill Inc publication.

[5] Livion, "SQL Complete Reference", second edition, McGraw Hill

[6] Bruce sheienier,"cryptography algorithms and source codes, The Blowfish Encryption Algorithm" – Tata mcgraw hill Inc

[7] "A pracical password –based two server authentication and key exchange system", Yanjiang Yang, Robert H.Deng and Feng Bao, IEEE transactions on dependable and secure computing, vol3, no.2, 2006.

[8] J.Brainard, A.Juels, B.Kaliski, and M.Szydlo, " A new two server approch for authentication with short secrets", Proc, USENIX Security Symp.2003

[9] Bruce Schneier, "Schneier on Security: Cryptanalysis of SHA-1" , htttp://www.schneier.com/blog/archives/20005/02/cryptanalysis_o_html, Februry 18, 2005.

[10] E. Bresson, O. Chevassut and D. Pointcheval, *A Security Solution for IEEE 802.11's Ad-hoc Mode: Password-Authentication and Group-Diffie-Hellman Key Exchange*, International Journal of Wireless and Mobile Computing. Special Issue on Security of Computer Network and Mobile Systems. Volume 2, Number 1, pages 4-13. © IJWMC, Inderscience, 2007.

[11] A.Allan, reserch note, Gartner reserch, G00124979, dec, 2004

AUTHORS PROFILE

[1] Dr. R. Shashikumar is presently working as a Professor

in E & C dept, SJCIT, Chikballapur, Karnataka, India. He is having 10 years of teaching and 6 years of Industry experience. His areas of interest includes ASIC, FPGA, Network Security, Cryptography.

[2] Prof.C.N.Vijayakumar M.E, MISTE, MIE, MIETE is presently working as a HOD and Assistant Professor in the department of Telecommunication engg , SJCIT, Chikballapur, Karnataka, India. He is having 15 years of teaching experience. His areas of interest are Power Electronics, Low Power VLSI, ASIC and Cryptography.

[3] Mr. H.M.Kotresh M.E, MISTE is working as a Senior Lecturer in the Dept of E & C, SJCIT, Chikballapur, Karnataka, India. He is having 7 years of teaching experience. His areas of interest are VLSI, Network security, Cryptography.

[4] Mr. K.Vijaykumar is M.Tech student in the department of Electronics, SJCIT, Chikballapur. His areas of interest are networking, image Processing and cryptography.