Incorporating Privacy and Security Features In an Open Source Search Engine

Advisor: Dr. Chris Pollett

Committee Members: Dr. Sami Khuri and Dr. Chris Tseng

Presented By

Akash Patel

Agenda

- Yioop
- Perfect Forward Secrecy
- Tor Network
- Hash CAPTCHA
- Zero Knowledge Authentication

Yioop

- Yioop, an open source PHP search engine based on GPLv3 license, is designed and developed by Dr. Chris Pollett
- It allows user to index a website or a collection of websites
- It can index page ranges up to tens or hundreds of millions
- It is designed to work on PC, smartphone and tablet

Perfect Forward Secrecy

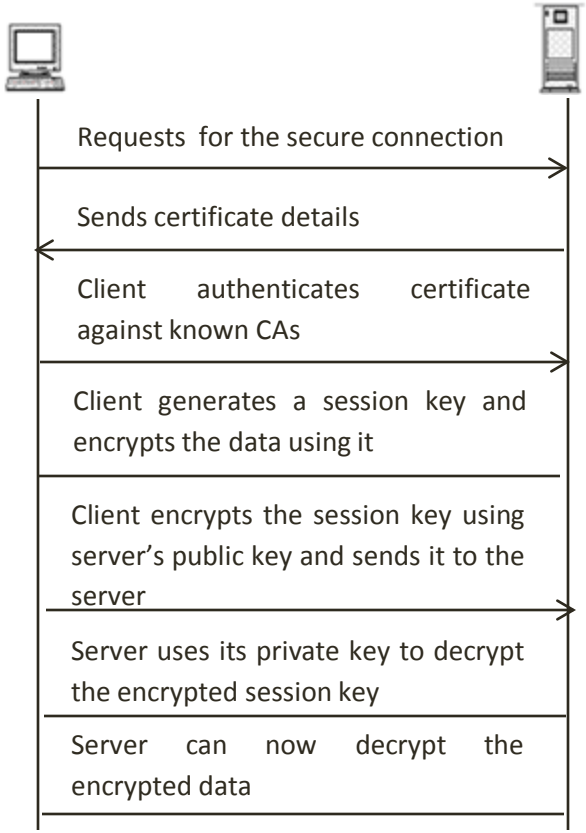
What is Perfect Forward Secrecy?

- Perfect Forward Secrecy (PFS) is a property of a key-agreement protocol which guarantees that a session key used to encrypt data will never be compromised even though a private key is compromised in the future

- The idea is not to use single key (e.g. private key) to generate all the session keys

Motivation behind PFS

- Let's assume a user accesses a website which uses HTTPS protocol for secure communication



SSL Handshaking

How PFS Works?

- In PFS, a client (browser) periodically creates a new session key
- Both client and server contribute to key generation
- Both use a random value for each key. The client and server forget this random value after the key creation
- Even if an attacker manages to obtain a private key, he would not know the random value generated by the client and the server so could not reconstruct the session key

How to Configure PFS?

- A webserver needs to be configured to enable PFS
- Usually the SSL configuration file on the web server has information about a list of cipher suites which are supported by the server
- The order of cipher suites is also important
- In order to enable PFS, DHE and ECDHE cipher suites should be given higher order in the configuration file

PFS Checker Script

- PFS checker is a PHP script written by us to verify whether a website supports PFS or not
- The input to the script is a URL of the website
- The script tries to make the connection with the website using different cipher suites and verifies whether the website supports PFS or not

Snapshot of PFS Checker Script's Output

Testing SSL server:https://www.yioop.com/

Cipher: ADH-AES256-SHA - Not Supported
Cipher: DHE-RSA-AES256-SHA - Supported
Cipher: DHE-DSS-AES256-SHA - Not Supported
Cipher: AES256-SHA - Supported
Cipher: ADH-AES128-SHA - Not Supported
Cipher: DHE-RSA-AES128-SHA - Supported
Cipher: DHE-DSS-AES128-SHA - Not Supported
Cipher: ECDHE-RSA-AES256-SHA - Not Supported
Cipher: AES128-SHA - Supported
Cipher: ADH-DES-CBC3-SHA - Not Supported
Cipher: ADH-DES-CBC-SHA - Not Supported
Cipher: EXP-ADH-DES-CBC-SHA - Not Supported
Cipher: ADH-RC4-MD5 - Not Supported
Cipher: EXP-ADH-RC4-MD5 - Not Supported
Cipher: EDH-RSA-DES-CBC3-SHA - Supported
Cipher: EDH-RSA-DES-CBC-SHA - Not Supported
Cipher: EXP-EDH-RSA-DES-CBC-SHA - Not Supported
Cipher: EDH-DSS-DES-CBC3-SHA - Not Supported
Cipher: EDH-DSS-DES-CBC-SHA - Not Supported

PFS Support:Yes

Request Header

array(26) { ["url"]=> string(22) "https://www.yioop.com/" ["content_type"]=> string(9) "text/html" ["http_code"]=> int(200) ["header_size"]=> int(547) ["request_size"]=> int(159) ["filetime"]=> int(-1) ["ssl_verify_result"]=> int(19) ["redirect_count"]=> int(0) ["total_time"]=> float(0.093) ["namelookup_time"]=> float(0) ["connect_time"]=> float(0) ["pretransfer_time"]=> float(0) ["size_upload"]=> float(0) ["size_download"]=> float(3954) ["speed_download"]=> float(42516) ["speed_upload"]=> float(0) ["download_content_length"]=> float(3954) ["upload_content_length"]=> float(0) ["starttransfer_time"]=> float(0.093) ["redirect_time"]=> float(0) ["redirect_url"]=> string(0) "" ["primary_ip"]=> string(13) "173.13.143.74" ["certinfo"]=> array(0) { } ["primary_port"]=> int(443) ["local_ip"]=> string(11) "192.168.1.4" ["local_port"]=> int(55871) }

Response Header

Array ([0] => HTTP/1.1 200 OK [1] => Date: Tue, 03 Dec 2013 01:22:39 GMT [2] => Server: Apache/2.2.24 (Unix) DAV/2 PHP/5.4.17 mod_ssl/2.2.24 OpenSSL/0.9.8y [3] => X-Powered-By: PHP/5.4.17 [4] => X-FRAME-OPTIONS: DENY [5] => Set-Cookie: yioopbiscuit=0vopseuhbnferj5ao4kq2j1shl432qpqmrfesom7u9persti2123u7l9cj0kg421bfn0t4mc1m3tpea32abvfch2ae5t5qb1jdlj0t3; path=/ [6] => Expires: Thu, 19 Nov 1981 08:52:00 GMT [7] => Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0 [8] => Pragma: no-cache [9] => MS-Author-Via: DAV [10] => Vary: Accept-Encoding [11] => Content-Length: 3954 [12] => Connection: close [13] => Content-Type: text/html)

PFS Check on Various Search Engines

Website Name	Supports PFS
Yioop.com	Yes
Google.com	Yes
Yahoo.com	Yes
Bing.com	No
Duckduckgo.com	Yes
AOL.com	No

Crawling TOR Network

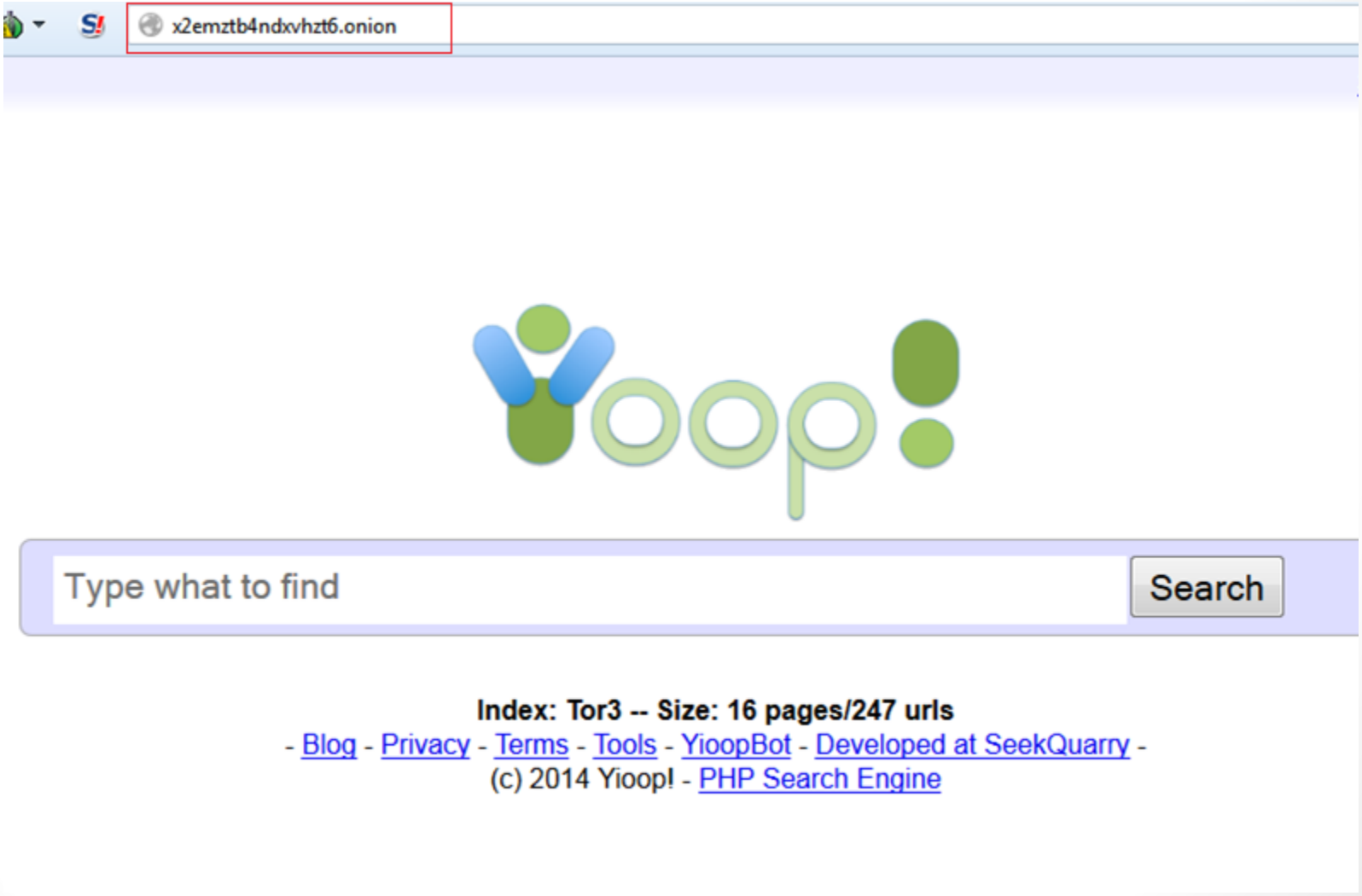
What is Tor Network?

- The Onion Router (Tor) is an open source software used by journalists, the military, corporates, and many others for anonymous internet surfing
- Tor network prevents the “Traffic analysis” attack
- Tor uses an Onion routing system
- It maintains a list of the available machines (a.k.a nodes) and uses them to direct traffic over the Internet
- Tor distributes the transaction among different nodes so there is no single point connection between the sender and receiver

Tor Hidden Service

- Tor hidden service allows publishers to publish the service without revealing their identity (IP address)
- Users can connect to a hidden service without knowing publisher of the service using the rendezvous point
- This type of anonymity provides protection against distributed DoS attacks
- We have published Yioop running on local host as a hidden service

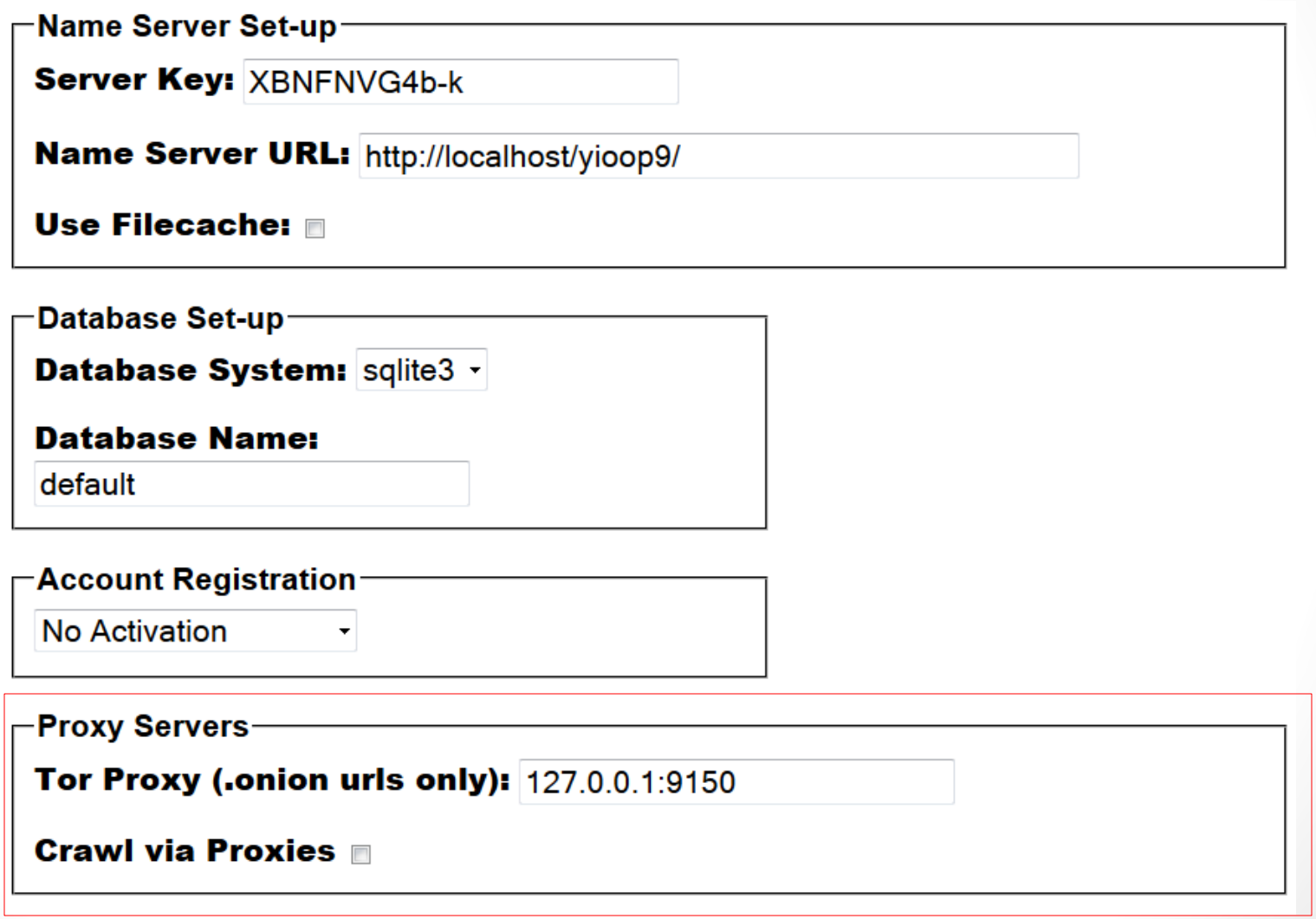
Snapshot of Yioop's Hidden Service



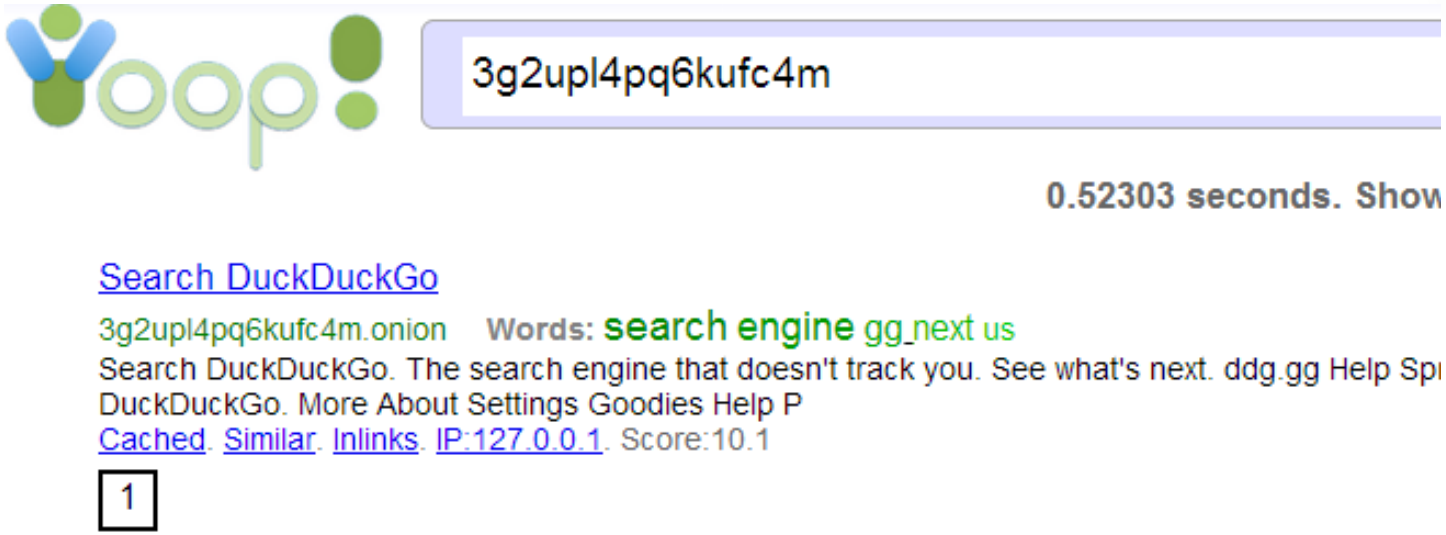
Crawling Tor Network

- Our goal was to implement the code in Yioop so it can crawl the Tor network and can index the Tor hidden services
- We have added code in the fetch URL module
- To verify whether Yioop can actually crawl the Onion URL, we did crawl on the onion URL
- After the crawl, we had set up the crawl data as search index and performed the search query

Snapshot of Tor Proxy Option in Yioop



Snapshot of Yioop Crawling an Onion URL



Tor Proxy Link

- In the client-server communication, a proxy server acts as an intermediate machine
- Ixquick is a popular search engine which provides proxy links for each query result, so the users can surf websites anonymously
- We have decided to implement a similar kind of feature in Yioop
- Yioop generates a proxy link for each search result
- Yioop server acts as a forward proxy server

Tor Proxy Link Feature

- We have implemented code in “search_view.php” file to create a dynamic proxy link

- we have also added class “ProxyURL” to receive a URL from the user and to make a request to the website through the Tor network

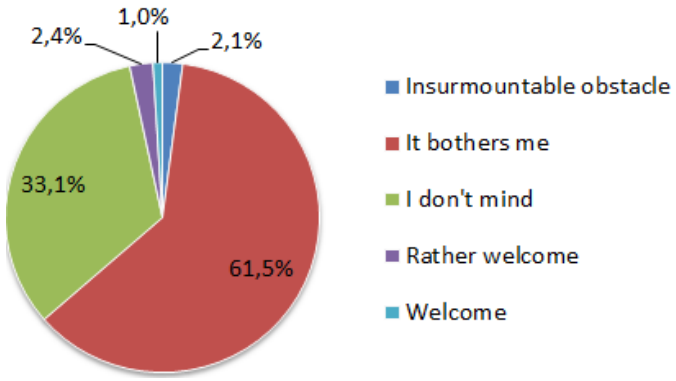
HASH CAPTCHA

CAPTCHA

- CAPTCHA stands for Completely Automated Public Turing test to tell Computers and Humans Apart
- It is a type of a challenge response test to find whether a user is human or machine
- CAPTCHA prevents spam submission by automated software thus improving the quality of the software
- It is mainly used in registration functionality, forgot password functionality, and in the comment field for a blog post

Motivation for Hash CAPTCHA

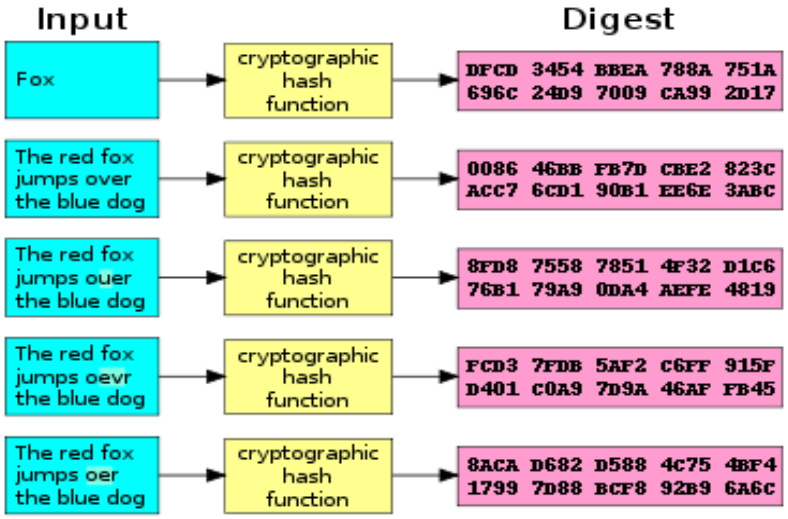
- Most of the users bother by the CAPTCHA
- We were looking for some technique that is user-friendly and at the same time does not compromise security



User's Response on CAPTCHA

Cryptographic Hash Function

- Hash CAPTCHA uses the concept of a cryptographic hash function
- A hash function takes an arbitrary length string as input and returns a fixed-size string as output
- Any minor change in the input should result in a completely different output string



Example of cryptographic hash function

Hash Functions

- SHA-1 and SHA-256 are very popular hash functions
- They were designed by United States National Security Agency
- SHA-1 generates 160-bit (20-byte) hash value and SHA-256 generates 256-bit (32-byte) hash value
- We had two choices, either use SHA1 or SHA256 to generate a hash value for the input string
- We have written a code to compare the execution time of SHA1 and SHA256 functions

Execution Time Comparison [50 iteration each]

#	Random String	SHA1 Time (ms)	SHA256 Time (ms)
1	041006c58b8168eb4096e3f34d0a16e4	11	8
2	1503130d07c2933db297e5fec1cb016b	7	3
3	1cea132c84b7311467e9d47f187af16c	2	7
4	c74b803684c192bb487137c2739cc0c7	3	3
5	e62e08814a6242ee7081c52c6f18a500	2	1
6	6f3174ec55e7499c676dfbd21dff31a1	3	3
7	295e64d28a763357da183c00421f0ec8	1	3
8	6e482115606c2377abbcc4545be62556	2	4
9	99621044dd037e730b9f96c344949413	1	2
10	d01a5d7696818fa79e0a7101c0b76c48	3	2
11	dcc73ab38e6ef10083de0617edd9c425	3	2
12	2b4babddd3511e2d4fcf5158d9805a14	1	5
13	a81664ba0b59b14cadad4f9bffebf77a	4	2
14	40eb5e28864484711466dfc0604f3037	3	2
15	a5ce82caf8066931746e49c4ec28b313	4	1

Proof of Work System

- A proof of work is a piece of data which is difficult to produce as it must satisfy certain requirements
- A client needs to perform some operations on the data at client side
- This setup prevents spam submission and denial attack by the client
- At the server side, it should be very easy to check whether the data sent by the client satisfies certain requirements or not

Hash CAPTCHA Design – 1/3

- In our implementation, the server sends a random string to the client
- JavaScript attached to the requested page calculates SHA1 on this random string
- The script takes the random string sent by the server as an input and concatenates the integer value known as nonce at the end of the input string
- The script then calculates SHA1 on the resultant string
- If the resultant hash begins with ‘00’ then script returns nonce value; otherwise, nonce is incremented by 1 and the entire process starting from string concatenation is repeated

Hash CAPTCHA Design – 2/3

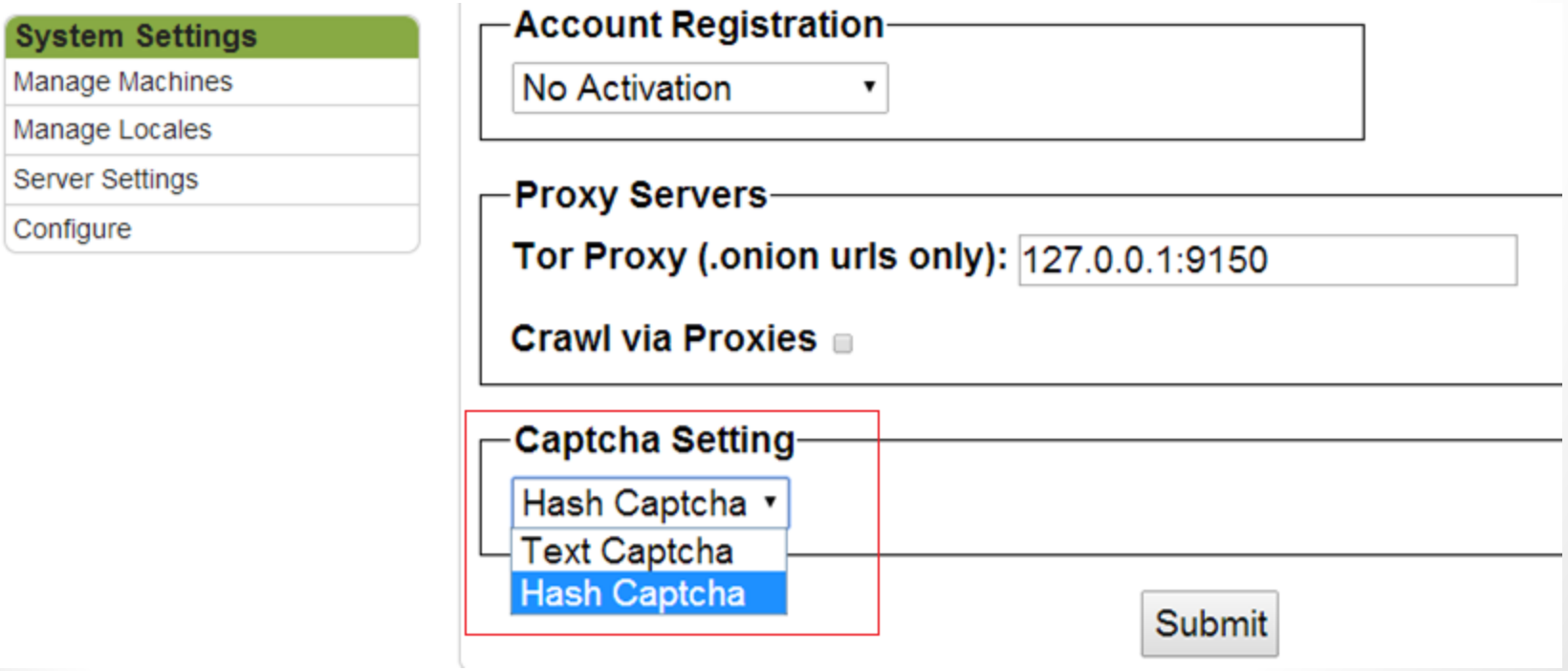
Input String	SHA1 Hash
Test:0	0b96f625c18f94dcebf01ab6bf84413743952f10
Test:1	6e8125d1846308969fb7cd4694b096333f310c4b
Test:2	b2da2df3450dab8c0797d9e7da357d1bde454c07
Test:3	0deb0e1be1b9889bddd4778f71eaa1c8c84845dd
Test:133	7985064947063366a79195bc11a423a491ff542f
Test:134	8575c8682f9fbb79034370fcfd0b17d5253a9ee2
Test:135	0022bd9826082dd8af48c412dde45beba7a1c3d3

Hash CAPTCHA Design – 3/3

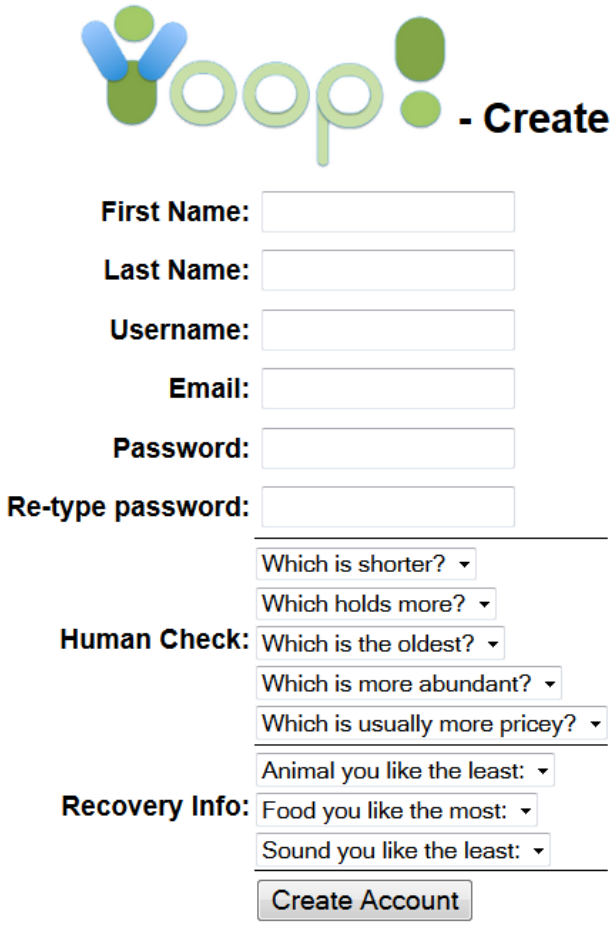
- At the server side, it is very easy to check whether the concatenation of the input string and nonce produces the desired number of leading zeroes or not
- In modern computers, calculation of SHA1 function does not take much time
- So this calculation would not affect the genuine client, but for the spammers it would be a highly CPU-intensive task.
- One advantage of this system is that it is very simple to increase the client side calculation by simply varying the desired number of leading zeroes of the generated hash value
- By simply changing this integer variable from 2 to 34, the client side calculation time will increase from approximately 1 second to 4 hours

Hash CAPTCHA in Yioop

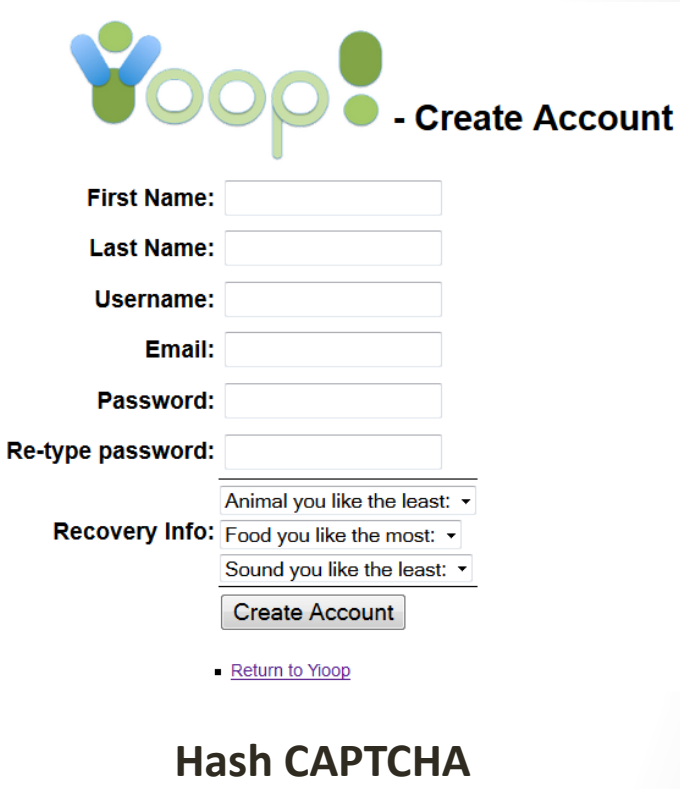
- Yioop is using text CAPTCHA for “create an account” and “suggest URL” functionality
- In the admin panel, we have added an option to toggle between text CAPTCHA and hash CAPTCHA



Hash CAPTCHA in Yioop



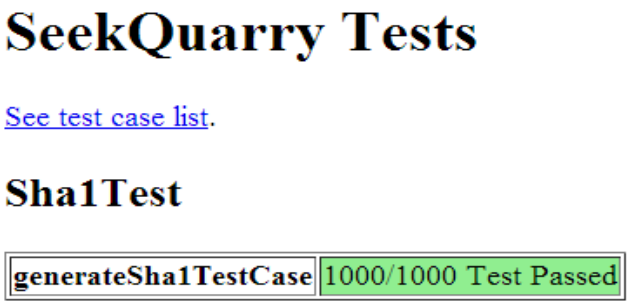
Text CAPTCHA



Hash CAPTCHA

SHA1 Test Case

- In hash CAPTCHA, JavaScript on the client side calculates SHA1 hash on the input string
- It was important to verify whether JavaScript calculates correct SHA1 or not
- We have decided to write the test cases for SHA1



Zero Knowledge Authentication System

Zero Knowledge Authentication System

- Authentication is a process of verifying a user's identity
- Users can provide their identity by providing a username and password, using a biometric card, or swiping the card
- The zero knowledge system allows user to prove that user knows the secret (i.e. password) without revealing the actual secret
- In this system, a user does not need to share the actual password over a network. The server also does not store the password

Fiat-Shamir Protocol

- The idea of the zero knowledge authentication system is based on an authentication scheme developed by Fiat and Shamir
- Let's say Alice wants to convince Bob that she knows the password without revealing the actual password to him
- It sounds impossible, but there is a probabilistic process by which Bob can verify that Alice knows the password
- Fiat-Shamir has developed a protocol for the authentication that relies on the fact that finding a square root modulo N is comparable to the difficulty of factoring

Fiat-Shamir Protocol

One time Set up:

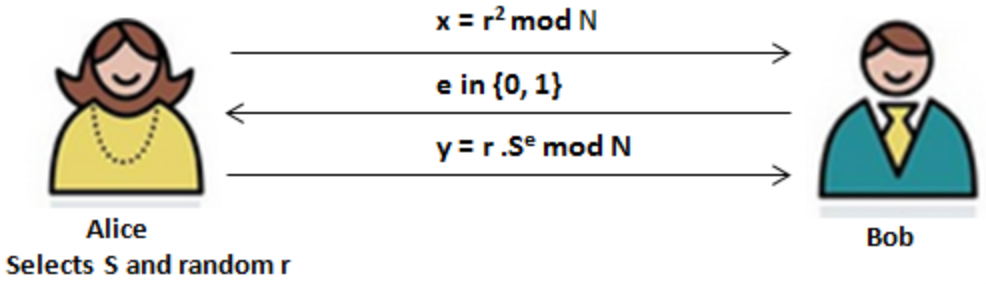
1. In this protocol, Trusted center selects RSA like modulus N = p.q , where p and q are secret prime number and N is public.
2. Alice select secret S such that S is coprime to N and 1<=S<=N-1, computes $V = S^2 \bmod N$. S is private and V is public.

Protocol:

1. Alice select random number r and sends $x = r^2 \bmod N$ to Bob.
2. Bob sends either 0 or 1 to Alice.
3. Alice sends $y = r \cdot S^e \bmod N$ to Bob.

Verification:

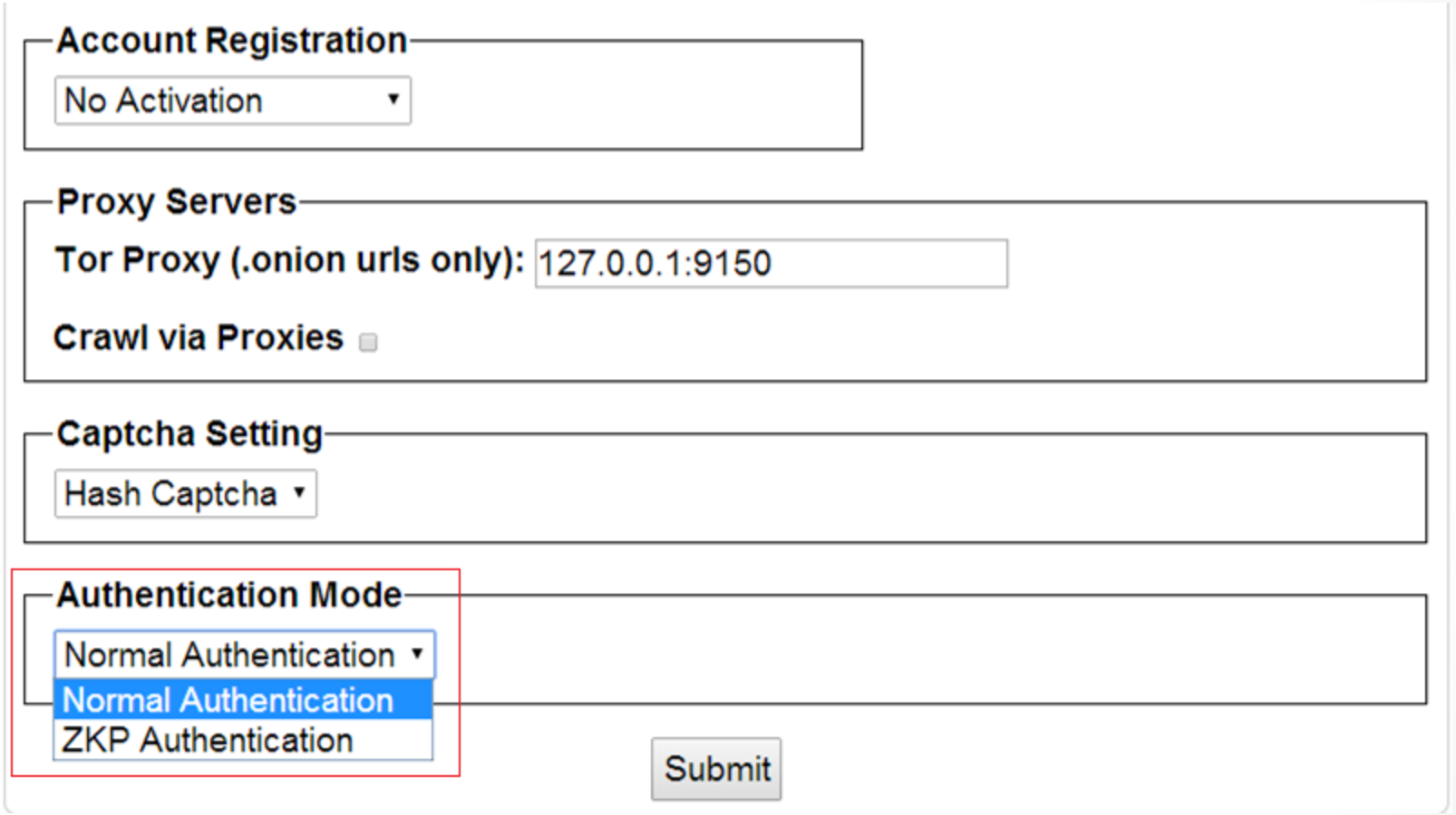
1. Bob verifies it with $y^2 = x \cdot V^e \bmod N$.



Zero Knowledge Authentication in Yioop

- Yioop has a functionality for creating user accounts
- We have decided to implement zero knowledge authentication for the login module
- We also want to provide an option of toggling between normal authentication and ZKP authentication
- The admin user has an option to select the authentication mode

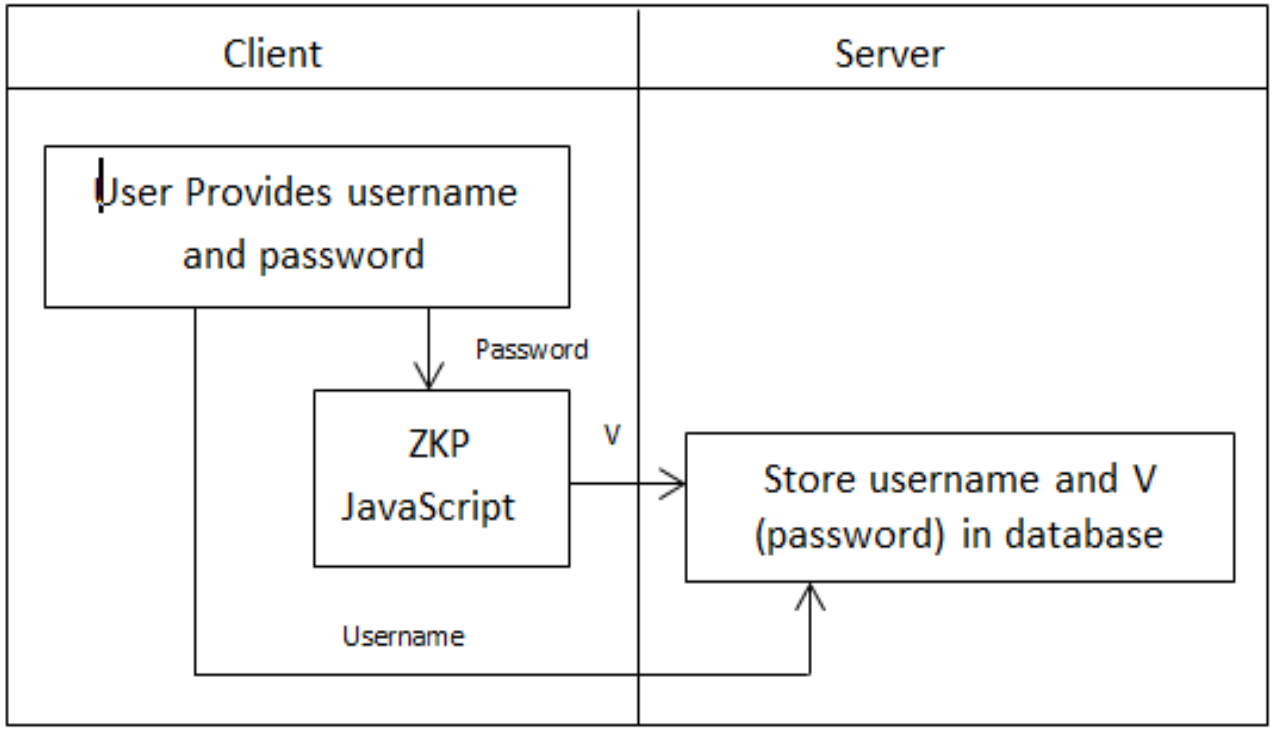
Snapshot of Authentication Mode Option In Yioop



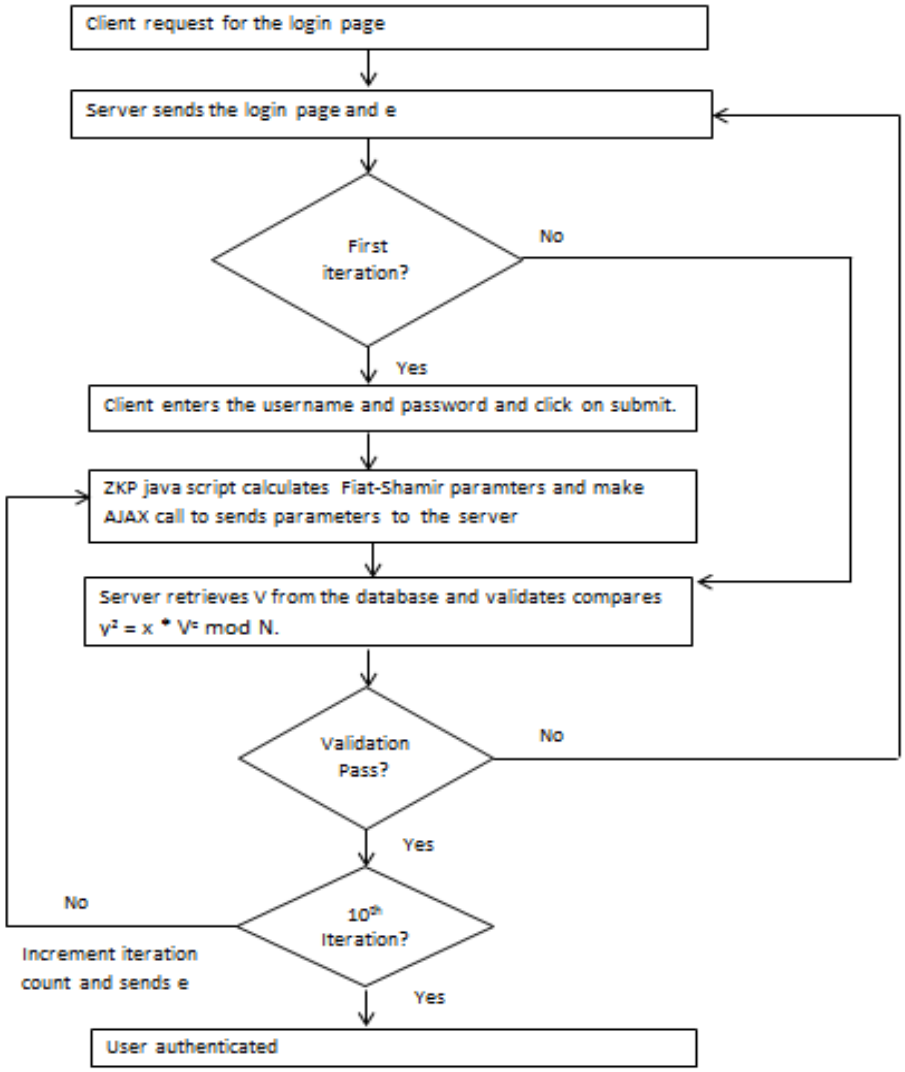
Zero Knowledge Authentication in Yioop

- One of the limitations of the zero knowledge system is that it can only work with numbers
- we have decided to use the SHA1 function to convert the user's password to a numeric value
- The biggest challenge in the implementation was to perform the mathematical operations like multiplication, power and modulo on very large number at the client's side
- The zero knowledge authentication system was difficult to break only when we used very large numbers like 100 digit number
- We have written our own big number library

Registration Module in Yioop



Login Module in Yioop



Conclusion

Feature	What We have done?	Why it is useful?
Perfect Forward Secrecy	- Code to configure PFS - PFS checker script	protects previous communications from retrospective decryption
Crawling Tor Network	- Code to crawl Tor - Tor proxy link	Allow crawling Tor network and indexing Tor hidden services
Hash CAPTCHA	- Design of hash CAPTCHA - SHA1 JavaScript - HashCaptcha JavaScript - SHA1 test case	Easy to implement and user friendly
ZKP Authentication System	- Implementation of ZKP - BigInt JavaScript - ZKP JavaScript	Unique way to authenticate a user

Thank You