A Unique-ID based Usable Multi-Factor Authentication Scheme for e-Services

Mohammed Misbahuddin, Roshni VS, Anna Thomas, Uttam Kumar

Centre for Development of Advanced Computing (C-DAC)
Electronics City, Bangalore - India
Email: mdmisbahuddin@gmail.com (Corresponding Author)

Abstract: *The current day web offers a wide range of e-Governance, e-commerce and other online services that require strong authentication mechanisms to safeguard user's account. In addition, these services require that a user be verified during registration to prevent duplication of accounts in cases where a fraudulent user creates multiple accounts with different credentials to avail the welfare services. Therefore, the challenge is to protect the e-services using secure multi-factor authentication methods with one account per user without compromising the usability. This paper discusses a multi-factor authentication (MFA) scheme which uses password, mobile token and image as multiple factors for authentication. The scheme uses a unique identity for verification of user accounts. The scheme leverages the identity verification system of Unique Identification Authority of India (UIDAI) System for ensuring the issuance of a unique and verified user identity to prevent duplicate and fraudulent accounts. The scheme does not maintain a verifier table at server to prevent stolen verifier attack. In addition, to achieve high level of usability, the scheme proposes to use Image Passwords and Mobile Tokens. The paper also discusses the security analysis of the proposed scheme against common authentication related attacks and the formal verification of the scheme using Scyther.*

Keywords: Two-factor Authentication, Mobile Tokens, e-Services, e-Governance, Graphical Passwords, Image Passwords, Stolen-verifier Attack, Formal Verification, Scyther.

1. Introduction

The advent of web 2.0 and Mobile Technology has led to enormous growth in web based services. The current statistics of Internet users shows that over 3 billion users are connected. In other words, today, over 40% of the World population has an Internet connection. [13]. The Web-based applications and services have changed the landscape of information delivery and exchange in today's corporate, government, and educational arenas. The common services that are offered to users include e-Governance, e-Banking, e-Shopping, financial services, e-scholarships, e-welfare schemes etc. The access to these services is protected mostly by Single Factor Authentication (SFA) methods. The most common example of SFA is Password based authentication. However, offering services using only SFA has various security challenges such as security against Password Guessing attack, Dictionary attack, Brute Force attack, Stolen-Verifier attack etc.

The security strength of authentication can be increased by deploying Two Factor Authentication (TFA) which requires the user to provide an additional factor for identity verification from separate categories of user credentials such as a password (first factor) and a physical token (as additional factor). There are various two factor authentication schemes that use Biometrics, Cryptographic Tokens and Smart Cards as additional factors. But due to the simplicity, cost efficiency and high security that the Smart Card based schemes offer, many researchers focused their attention on designing Smart Card based schemes without verification table at server [1-9]. However, the wide acceptance of Smart Card based schemes requires every user to have a card reader attached to their PCs.

Most recently, using mobile as a second factor for authentication gained significance as the penetration rate of mobile in many of the countries across the globe has crossed 65% of population [14]. Among the given percentage of mobile phone users, atleast 50% owns smart phones and over 83% of them uses mobile Internet. [15].

The mobile phones have come a long way in a very short time with the rapid advancements in mobile technology. With the increase in Smartphone use, it is evident that the usage of mobile as a second factor will surely reap security and usability benefits.

To avail an e-service, a user need to register with each service provider separately leading to creation of multiple user accounts for a single user with different credentials on different services. There are also cases wherein the fraudulent users register for a single welfare or scholarship service multiple times with different credentials to avail the benefits multiple times illegally. Hence, there is a need to verify the identity of a user that is unique and issued by a trusted third party. This identity can be used as a common verified identity of a particular user for all the web accounts the user has registered with.

It is therefore evident that, offering an e-Service securely requires an authentication scheme which shall address the following challenges: 1) Issuance of a verified identity to a user to prevent misuse. 2) Prevention of duplicate and fraudulent accounts 3) Strong authentication using TFA 4) Usable password methods for ease of password remembrance 5) No verifier table at the server to prevent Stolen-verifier attack.

This paper discusses a TFA scheme which uses an image as the first factor and a Mobile Token (MT) as the second factor. The Mobile Token is a mobile application which contains user's personalized data and runs on the user's mobile during authentication phase. During registration, the scheme leverages on the services of UIDAI's CIDR [16] to verify the validity of a user before issuing a registration ID to access e-Services. The proposed scheme can be integrated with the SSO component using SAML [17] protocol for seamless access of multiple services by the registered users.

The proposed scheme has the following features:

- Issues a verified user identity
- Employs a mobile based authentication scheme with multiple factors
- Prevents duplicate account creation
- Presents a unique graphical password method
- Offers Multi-factor Authentication (MFA) Security for SSO services
- Resistant to various attacks such as Guessing, Stolen-verifier, Replay etc.
- Secure mutual authentication between mobile token and server

The paper also discusses the security analysis of the scheme besides presenting the results of automated formal verification using Scyther tool [11-12].

The rest of the paper is organized as follows; Section II presents the proposed scheme, Section III presents the security analysis of the proposed scheme, section IV presents the formal verification of the proposed scheme using Scyther and finally section V presents the conclusion

2. The Proposed Scheme

This section presents the Architecture and Protocol of the proposed scheme. The scheme consists of four phases namely, Registration, Authentication, Password Change and Forget Password phase.

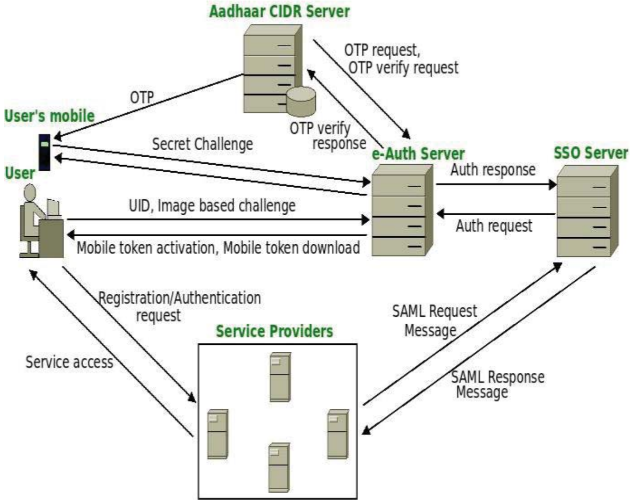


Fig 1: Architecture Diagram

The architecture of the proposed scheme is shown in fig-1 and comprises of the following components in the proposed architecture. These components are described in section 2.1

1) Central Identity Repository (CIDR),
2) User's Mobile Token (MT),
3) Service Provider (SP),
4) e-Authentication Server (AS),
5) SSO Server (SSS).

The proposed scheme uses UIDAI's Aadhaar Authentication Ecosystem (Fig 2) for registration of users of e-services. Aadhaar Authentication is the process wherein an Aadhaar (a unique residents ID), along with other attributes (demographic and/or biometrics and/or OTP) are sent to UIDAI's Central Identities Data Repository (CIDR) for verification; the CIDR verifies whether the data submitted matches the data available in CIDR and responds with a "Yes/No". The personal identity information is never returned as part of the response [16]. The UIDAI provides open APIs to be integrated with e-Services for verification of user's identity. The Aadhaar Authentication eco system is shown in fig 2. This facility helps in ensuring a single verified login identity for each user and also helps in preventing the creation of fraudulent and duplicate accounts on the e-Auth server.

The scheme proposes a unique feature wherein the users' secret credentials are not stored on any of the components given in the architecture except user's mobile token.

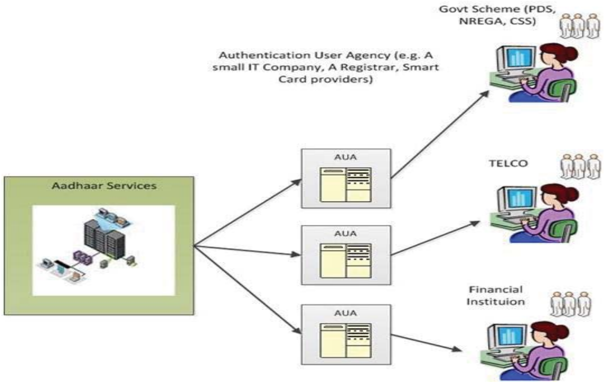

Govt Scheme (PDS, NREGA, CSS)
Authentication User Agency (e.g. A small IT Company, A Registrar, Smart Card providers)
Aadhaar Services
AUA
AUA
AUA
TELCO
Financial Instituion

Fig 2: Aadhaar Authentication Eco System

2.1 Architectural Components

Central Identity Repository (CIDR): The CIDR is the central data repository, and functions as a managed service provider. It implements the core services around the UIDAI – it stores resident records, issue unique identification numbers, and verifies, authenticates and amends resident data. The CIDR holds the minimum information required to identify the resident and ensure no duplicates. This includes:

- Unique Identity (UID) Number
- Unique ID agencies
- Demographic data and biometrics

UIDAI collects the user's data such as mobile number, Demographic data and Biometric and issues a Unique Identity (UID) also called as 'Aadhaar' after verification. While certain types of information such as birth date and gender will remain unchanged, other demographic details may undergo changes with time. The agency monitors these changes periodically with the assistance of a network of registrars who oversee the initial enrollment process for the issuance of UIDs and subsequent change requests. The CIDR adheres to the national and international security standards to ensure the security and privacy of users' data.

While CIDR can be used to verify the validity of Indian residents only; most of the developed and developing nations may make use of the Citizen ID or National ID databases for online identity verification of their users during registration time to issue verified online identity for web accounts. [18]

Service Provider (SP): The SPs offers the services in electronic mode through web based services such as e-Governance, e-Shopping, e-Banking etc. The service providers do not register the users directly and instead delegate the registration to e-Auth server (AS). The SPs do not maintain any authentication related data of their users but keeps only the profile data of the users. They may agree for a SSO server environment.

e-Authentication Server (AS): The AS is the most important component of this architecture. It runs the Registration, Authentication engine and provides these services to the users on behalf of SPs. The registration of a valid user account is accomplished with the help of CIDR. The authentication is carried out using MFA scheme which does not maintain password table at the server. The AS communicates with all the entities in the authentication architecture with or without SSO. The AS does not maintain any secret credentials of the users except the profile data.

Single Sign-on Server (SSS): The SSS runs the SAML protocol for providing the SSO services to the user who wishes to access multiple services in a single login session. It is responsible for creating and managing the SSO session, SSO token and SSO communication with all SPs. SSS does not maintain any secret credential of the user.

Mobile Token (MT): MT is a mobile application which is issued to the user by AS after successful registration. The MT contains the personalized registration and authentication related secrets of the user. The user may need to protect it with a PIN for better security. When the user wishes to login she has to activate the MT by keying the PIN.

2.2 Multi-factor Authentication (MFA) Scheme

2.2.1 Registration Phase

If a user 'Ui' wishes to register with AS, she submits the registration request to AS. In case, she submits the registration request to SP, the request is forwarded to AS. The user then proceeds for registration as follows:

Step R1: Ui submits her UID to AS. The AS forwards it to CIDR.
Step R2: CIDR sends OTP to users' registered mobile.
Step R3: Ui submits the OTP which is verified by CIDR
Step R4: Upon successful verification of OTP, CIDR sends the Users' profile data to AS.
Step R5: AS presents the registration form with the profile data populated on it and an image grid to User for choosing an Image (I) as his secret.
Step R6: Ui chooses an 'I' and sets a text password 'p' and submits it to AS.
Step R7: AS computes $h(Pwi) = h(h(I) \oplus p)$; $a = h(UID)$
$Ni = h(Pwi) \oplus h(x \oplus TIDi)$; where x is server's master secret key.
Step R8: AS personalizes an MT with the parameters Ni, h(I), h(Pwi), a, y_i. where y_i is the server's secret key shared with each MT whereas TIDi is the Token ID (unique for each MT) and is stored at the server with a mapping of UID.
Step R4: AS sends email to Ui with downloadable link of MT.

2.2.2 Authentication Phase

A registered user 'Ui' wishes to login to access resources at the server 'SP' will proceed as follows:
Step L1: Ui requests for login to SP for availing services. SP redirects the request to AS.
Step L2: AS asks Ui to enter UID.
Step L3: Ui keys in her UID to AS.
Step L4: AS retrieves TIDi and generates the image grid consisting of random images and the Ui's chosen Image.
Step L5: AS generates random codes 'r' displayed on each image of the grid.
Step L6: AS presents the image grid with 'r' embossed on images as challenge to Ui.
Step L7: Ui activates her MT and enters her text password 'p'. She also identifies her chosen Image from the challenge image grid and keys in 'r' in MT.
Step L8: Ui submits 'p' and 'r' to MT.
Step L9: MT computes h'(Pwi) and checks if
$h'(Pwi) = = h(Pwi)$. If it holds, MT proceeds to compute
$Bi = h'(Pwi) \oplus Ni$; $Ci = h(Bi \oplus r)$; and $E = E_{yi}(Ci, a)$
Step L10: MT sends E to AS where E is encryption of message using y_i. Upon receipt of E, AS proceeds as follows:
Step L11: AS computes $D_{yi}(Ci,a)$ using $y_{i.}$ where D is decryption. It then retrieves TIDi using 'a',
Step L12: AS computes $Bi' = h(x \oplus TIDi)$; $Ci' = h(Bi' \oplus r)$
Step L13: AS checks if Ci == Ci', if it holds then AS transfers the control to SSO module with authentication success message. SSO provides access to SPi to the user.
The Registration Phase and Authentication Phase are depicted in figures 4 and 5 on last page.

2.2.3 Password Change Phase

A Registered user can change her password by sending password change request to the AS. The change password request is allowed only after successful login.
Step C1: Ui requests for Change Password

Step C2: AS presents image grid to Ui for choosing secret image (I)
Step C3: Ui chooses a ‘I’ and submits to AS
Step C4: AS sends $E_{yi}(h(I))$ and a request to change text password at MT to Ui
Step C5: Ui activates MT and enters new password ‘p’
Step C6: MT computes $Ni_n = h_n(h(I) \oplus p)$ and replaces Ni with the new Ni_n, h(I), h(Pwi).
Step C7: MT sends a confirmation message to AS

2.2.4 Forgot Password

If a user Ui forgets her password (the MT is not lost), she submits the reset password request to AS and proceed as follows:
Step F1: Ui submits request for reset password
Step F2: AS asks Ui to submit her UID
Step F3: Ui submits her UID.
Step F4: AS forwards the UID to CIDR
Step F5: CIDR sends OTP to Ui mobile
Step F6: Ui submits OTP to AS
Step F7: AS forwards the OTP to CIDR for verification
Step F8: Upon successful verification, CIDR sends a positive acknowledgement to AS
Step F9: AS presents a secret question to Ui as challenge or sends email to user with a password reset link
Step F10: When Ui clicks on the link, AS presents a new image grid for choosing a secret image (I).
Step F11: Ui chooses ‘I’ and submits it to AS
Step F12: AS sends $E_{yi}(h(I))$ and a request to change text password at MT to Ui
Step F13: Ui activates MT and enters new password ‘p’
Step F14: MT computes $Ni_n = h_n(h(I) \oplus p)$ and replaces Ni with the new Ni_n, h(I), h(Pwi).
Step F15: MT sends a confirmation message to AS.

2.2.5 Loss of Mobile Token

If a registered user Ui loses her mobile token or accidently deletes it, she has to get a new MT from server by following the steps described in the registration process.

3. Security Analysis

This section presents the security analysis of proposed scheme against various attacks given below.

3.1 Replay Attack

In replay Attack, the adversary intercepts the message transmitted between two parties and then sends it at a later time to gain access to the resources. In the proposed scheme, if the message $E_{yi}(Ci, a)$, transmitted in step L10 is intercepted and replayed by adversary, he cannot gain access to the resources because both the client and server checks for the freshness of nonce every time they receive a message. Hence the attack will fail. Here ‘r’ is a random nonce which is displayed fresh at every login request.

3.2 Insider Attack

The insider attack is usually performed by an insider of the organization who has access to the sensitive resources by revealing the user secret information to others. In the proposed scheme, the user’s password is never stored on any component in architecture. Moreover, none of the parameters required in protocol computation are stored in server in plain text form, instead, to avoid such attack, all the user credentials are stored in the database as message digest which is irreversible.

3.3 Stolen Verifier Attack

To perform this attack, an adversary who has access to the database server steals the password verifier and later uses it for offline guessing attack. But since, the proposed scheme does not maintain a verifier table the attack cannot be performed.

3.4 Server Spoofing Attack

It refers to the situation where in the attacker pretends to be a legitimate server and communicates with client to gain knowledge of the user’s secret credentials.
To perform this attack on proposed scheme, the adversary has to fool the user by creating a page which looks similar to a valid server page and then redirect the user’s login request in Step L1 to malicious server and subsequently generate the image grid as per step L4. Even if the malicious server generates the image grid (which is highly difficult), it will not be able to generate the Image selected by the user during registration.

3.5 Fraudulently copying the Mobile Token

If an adversary gets access to the user’s MT and wants to fraudulently copy the MT (.apk file) which stores the parameters Ni, h(I), h(Pwi), a, y_i, he can neither retrieve the user’s secret Pwi nor the server’s master secret ‘x’ from the available parameters as the Pwi, x are stored as a message digest.

3.6 Denial of Service Attack

Suppose, if the adversary who has control over the server, modifies any secret information of the user stored in the database server by replacing it with a newly created digest, then the user will not be able to login even with his valid credentials. This is called as denial of service attack. In the proposed scheme, since there is no secret information stored on the server, the denial of service attack will not work.

3.7 Storage of Registration Data

In the proposed scheme a user profile is maintained, which stores secret questions; answers to secret questions in message digest form so that even if the attacker gets access to the database he cannot figure out the answers of the secret questions the user has set.

4. Formal Verification using Scyther

This section will present a brief description of the automated formal verification tool called ‘Scyther’ and the specifications of Security Protocol Description Language (SPDL) with .spdl extension for the schemes / protocols to verify the security claims.

Scyther is an automated formal verification tool implemented based on formal and mathematical logics.

Scyther is considered to have many novel features compared to other open source counterparts.

The major objective of Scyther is to guarantee the security of protocol. For this a mathematical model of the protocol and the network is to be created with the assumption that the network is under full control of adversary, meaning that the adversary can intercept, modify, fabricate etc. Since, modeling all the protocol primitives and the cryptographic mechanisms used in the protocol makes a model complex; the cryptographic mechanisms are abstracted with few assumptions, firstly that these mechanisms provides perfect security when the key used to encrypt is known only to the communicating parties. Secondly, it is assumed that the adversary can either decrypt every message or he cannot decrypt any. And finally, that the adversary has full control of the network.

The main objective of formal semantics of Scyther is to clearly distinguish the protocol descriptions from their behavior and the attacker model. Every security protocol has a number of distinct behaviors which are called as roles. For example, in the proposed scheme, the client and server are two 'roles'. A system consists of number of communicating 'agents', where each agent performs one or more role. Therefore, it can be said that, the system does not execute the protocol; instead, it executes the roles performed by agents. Each role performed by an agent is called as 'run'. While agents try to perform roles to achieve security, the attacker tries to oppose them by breaching their security. Each security protocol model can have the following components:

Scyther's specification language is called as Security Protocol Description Language and takes the file extension of .spdl. There are number of basic terms used in .spdl. These include:

Var: Variables that are used to store received messages.

Const: the fresh constants which are generated at each instantiation of role such as nonce, keys etc.

Role: The roles that an agent performs

Func: Represents the function names.

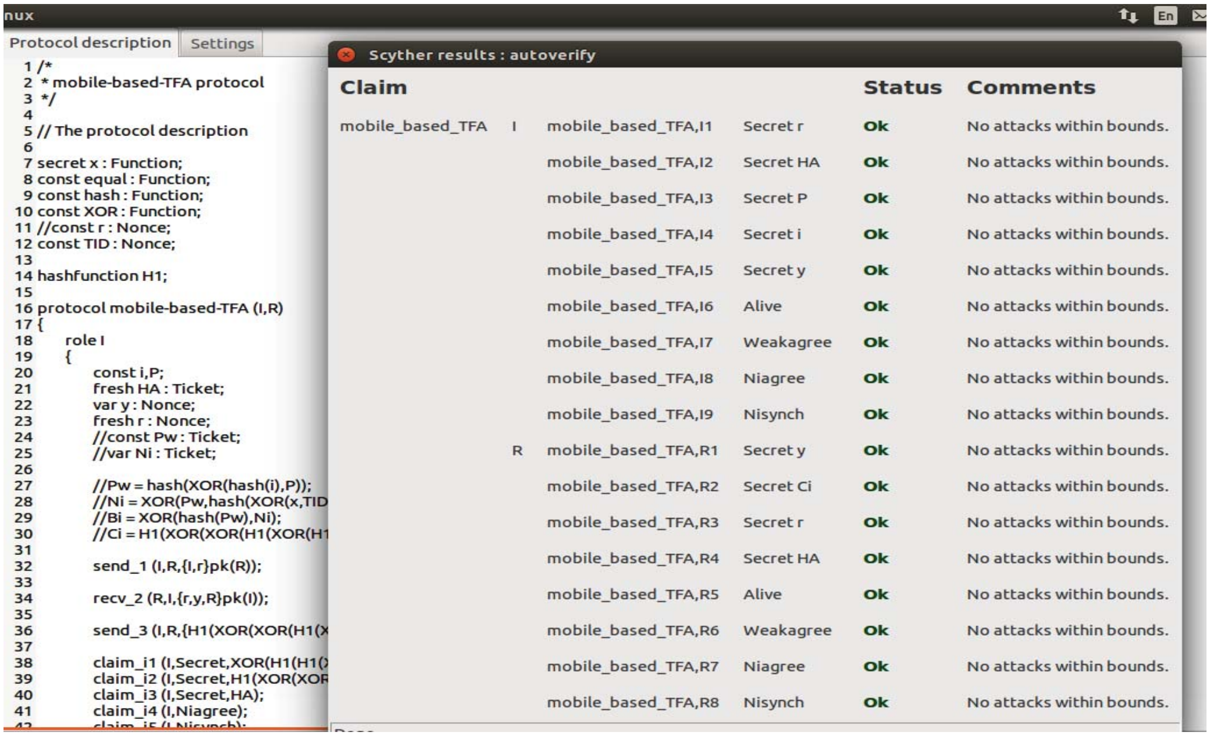
Scyther provides GUI to write the specification in addition to CUI interface. It also assists the protocol analysis by providing classes of attacks as compared to single attack provided by other tools. The tool can be used in three modes i,e. to verify the user defined claims, to verify the automatic claims generated by scyther and to analyze the performance of the protocol in terms of traces by characterization.

Description of Proposed Scheme: As per the specification the Agent model in the proposed scheme has two agents i,e. 'C' – Client and 'S' – Server. Each agent performs the roles, therefore this scheme has two roles that are named after the agents i,e. 'role I' and 'role S'. The adversary model is also designed considering that the adversary has complete control of the network. Therefore, Eve is considered as adversary. Since Scyther checks for the freshness and synchronization by default, those attributes have also been claimed.

Once the tool is run for automated verification, all the claims are analyzed against the automated attacks that the tool has. If any one of the claim is failed to resist the attack, the result of verification will be "Fail"; conversely, if the scheme resists all the attacks then the result will be 'OK' for all claims. For the proposed scheme, the parameters that are being transmitted over the channel were provided as claims. These include, h(UIDi), E_{yi}(Ci, a). The result of the automated verification of proposed scheme is found to be successful as depicted in figure 3. The .spdl is given below.

```
/*
 * mobile-based-TFA protocol
 */
// The protocol description
secret x : Function;
const equal : Function;
const hash : Function;
const XOR : Function;
//const r : Nonce;
const TID : Nonce;
hashfunction H1;
protocol mobile-based-TFA (I,R)
{
        role I
        {
                const i,P;
                fresh HA : Ticket;
                var y : Nonce;
                fresh r : Nonce;

                send_1 (I,R,{I,r}pk(R));
                recv_2 (R,I,{r,y,R}pk(I));
                send_3
(I,R,{H1(XOR(XOR(H1(XOR(H1(i),P)),XOR(H1(XOR(H
1(i),P)),H1(XOR(x,TID)))),r)),y}pk(R),{H1(HA,y)}pk(R));
        claim_i1
(I,Secret,XOR(H1(H1(XOR(H1(i),P))),XOR(H1(XOR(H1(
i),P)),H1(XOR(x,TID)))));
                claim_i2
(I,Secret,H1(XOR(XOR(H1(XOR(H1(i),P)),XOR(H1(XO
R(H1(i),P)),H1(XOR(x,TID)))),r)));
                claim_i3 (I,Secret,HA);
                claim_i4 (I,Niagree);
                claim_i5 (I,Nisynch);
        }
        role R
        {
                var HA : Ticket;
                var r : Nonce;
                fresh y : Nonce;
                var Ci : Ticket;
                recv_1 (I,R,{I,r}pk(R));
                send_2 (R,I,{r,y,R}pk(I));
                recv_3
(I,R,{Ci,y}pk(R),{H1(HA,y)}pk(R));
                claim_r1 (R,Secret,H1(XOR(x,TID)));
                claim_r2 (R,Secret,Ci);
                claim_r3
(R,Secret,H1(XOR(H1(XOR(x,TID)),r)));
                claim_r4 (R,Secret,HA);
                claim_r5 (R,Niagree);
claim_r6 (R,Nisynch);                                    } }
```



```
/*
 * mobile-based-TFA protocol
 */

// The protocol description

secret x : Function;
const equal : Function;
const hash : Function;
const XOR : Function;
//const r : Nonce;
const TID : Nonce;

hashfunction H1;

protocol mobile-based-TFA (I,R)
{
    role I
    {
        const i,P;
        fresh HA : Ticket;
        var y : Nonce;
        fresh r : Nonce;
        //const Pw : Ticket;
        //var Ni : Ticket;

        //Pw = hash(XOR(hash(i),P));
        //Ni = XOR(Pw,hash(XOR(x,TID
        //Bi = XOR(hash(Pw),Ni);
        //Ci = H1(XOR(XOR(H1(XOR(H1

        send_1 (I,R,{I,r}pk(R));

        recv_2 (R,I,{r,y,R}pk(I));

        send_3 (I,R,{H1(XOR(XOR(H1(X

        claim_i1 (I,Secret,XOR(H1(H1(X
        claim_i2 (I,Secret,H1(XOR(XOR
        claim_i3 (I,Secret,HA);
        claim_i4 (I,Niagree);
        claim_i5 (I,Nisynch);
```

Claim				Status	Comments
mobile_based_TFA	I	mobile_based_TFA,I1	Secret r	Ok	No attacks within bounds.
		mobile_based_TFA,I2	Secret HA	Ok	No attacks within bounds.
		mobile_based_TFA,I3	Secret P	Ok	No attacks within bounds.
		mobile_based_TFA,I4	Secret i	Ok	No attacks within bounds.
		mobile_based_TFA,I5	Secret y	Ok	No attacks within bounds.
		mobile_based_TFA,I6	Alive	Ok	No attacks within bounds.
		mobile_based_TFA,I7	Weakagree	Ok	No attacks within bounds.
		mobile_based_TFA,I8	Niagree	Ok	No attacks within bounds.
		mobile_based_TFA,I9	Nisynch	Ok	No attacks within bounds.
	R	mobile_based_TFA,R1	Secret y	Ok	No attacks within bounds.
		mobile_based_TFA,R2	Secret Ci	Ok	No attacks within bounds.
		mobile_based_TFA,R3	Secret r	Ok	No attacks within bounds.
		mobile_based_TFA,R4	Secret HA	Ok	No attacks within bounds.
		mobile_based_TFA,R5	Alive	Ok	No attacks within bounds.
		mobile_based_TFA,R6	Weakagree	Ok	No attacks within bounds.
		mobile_based_TFA,R7	Niagree	Ok	No attacks within bounds.
		mobile_based_TFA,R8	Nisynch	Ok	No attacks within bounds.

Fig 3: Result of Automated verification of proposed Scheme

5. Conclusion

This paper presented a Multi-Factor Authentication (MFA) Scheme for e-Services which does not require verifier table at the server. The paper discussed the security analysis of the proposed scheme against common attacks and the automated attacks using theoretical and formal analysis respectively. The authentication scheme is usable and secure as it considers image as the first factor, mobile token as second and password as the third factor. The Multi-factor security is required as the scheme is used with SSO for accessing multiple services in a single login session. The registration phase creates a verified identity of the user and prevents the fraudulent creation of duplicate accounts. This is accomplished using CIDR verification of Indian residents. However, for large scale deployment of this scheme, if the Citizen ID and National ID databases of developed and developing nations provide APIs to verify the validity of their residents then the issuance of verified identity would be a reality.

6. References

[1] Shunmuganathan, S. ; Saravanan, R.D. ; Palanichamy, Y., “Secure and Efficient Smart-Card-Based Remote UserAuthentication Scheme for Multiserver Environment” Electrical and Computer Engineering, Canadian Journal of Volume: 38 , Issue: 1, 2015 , Page(s): 20 - 30

[2] Das M. L., Saxena A. and Gulati V. P., “A dynamic ID based remote user authentication scheme”, IEEE Trans. Consumer Electronics, May, vol.50, No. 2, 2004, Pg. 629 - 631

[3] Jenq-Shiou Leu ; Wen-Bin Hsieh, “Efficient and secure dynamic ID-based remote user authentication scheme for distributedsystems using smartcards, Information Security, IET Volume: 8 , Issue: 2, 2014 , Page(s): 104 - 113H.

[4] Haw Lee ; Wei-Chih Hong ; Chia-Hung Kao ; Chen-Mou Cheng, "A User-Friendly Authentication Solution Using NFC Card Emulation on Android”, IEEE 7th International Conference on Service-Oriented Computing and Applications (SOCA), 2014, Page(s): 271 – 278

[5] Binu, Sumitra ; Misbahuddin, Mohammed ; Raj, Pethuru, "A Single Sign on based secure remote user authentication scheme for Multi-Server Environments”, International Conference on Computer and Communications Technologies (ICCCT),2014, Page(s): 1 – 6

[6] Kumari, S. ; Om, H.,” Remote login password authentication scheme usingtangent theorem on circle in a multi-server environment” First International Conference on Networks & Soft Computing (ICNSC), 2014, Page(s): 76 - 80

[7] Misbahuddin M, Ahmed M.A, Rao A.A, Bindu C.S, Khan M.A.M, “A Novel Dynamic ID-Based Remote User Authentication Scheme”, in the proceedings of Annual IEEE Indicon Conference, Delhi, 2006

[8] Mohammed Misbahuddin; Mohammed Aijaz Ahmed; M.H. Shastri, “A Simple and Efficient Solution to Remote User Authentication Using Smart Cards”, in the proceedings of IEEE International Conference on Innovations in IT (IIT ‘06), Dubai, 2006

[9] Omar Cheikrouhou, Manel Boujelben, Anis Koubaa, Mohamed Abid, Attacks and Improvement of “Security Enhancement for a Dynamic ID-based Remote User Authentication Scheme”, in the proceedings of IEEE

International Conference on Computer Systems and Applications, 2009.

[10] Bruce Schneier, Applied Cryptography, 2nd edition. John Wiley & Sons, 1996.

[11] Cremers CJF, "Scyther - Semantics and Verification of Security Protocols", Phd Thesis, http://alexandria.tue.nl/extra2/200612074.pdf

[12] Cas Cremers, "The Scyther Tool Verification, Falsification, and Analysis of Security Protocols", Tool Paper, http://people.inf.ethz.ch/cremersc/downloads/papers/The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols.pdf

[13] http://www.internetlivestats.com/internet-users/ Last accessed March 31, 2015

[14] http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536, Last accessed March 31, 2015

[15] The World in 2013 - http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2013-e.pdf Last accessed March 31, 2015

[16] Aadhaar Authentication Overview, Aadhaar Authentication Overview, https://uidai.gov.in/auth.html Last accessed March 31, 2015

[17] http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, Last accessed March 31, 2015

[18] http://en.wikipedia.org/wiki/National_identity_cards_in_the_European_Economic_Area, Last accessed March 31, 2015

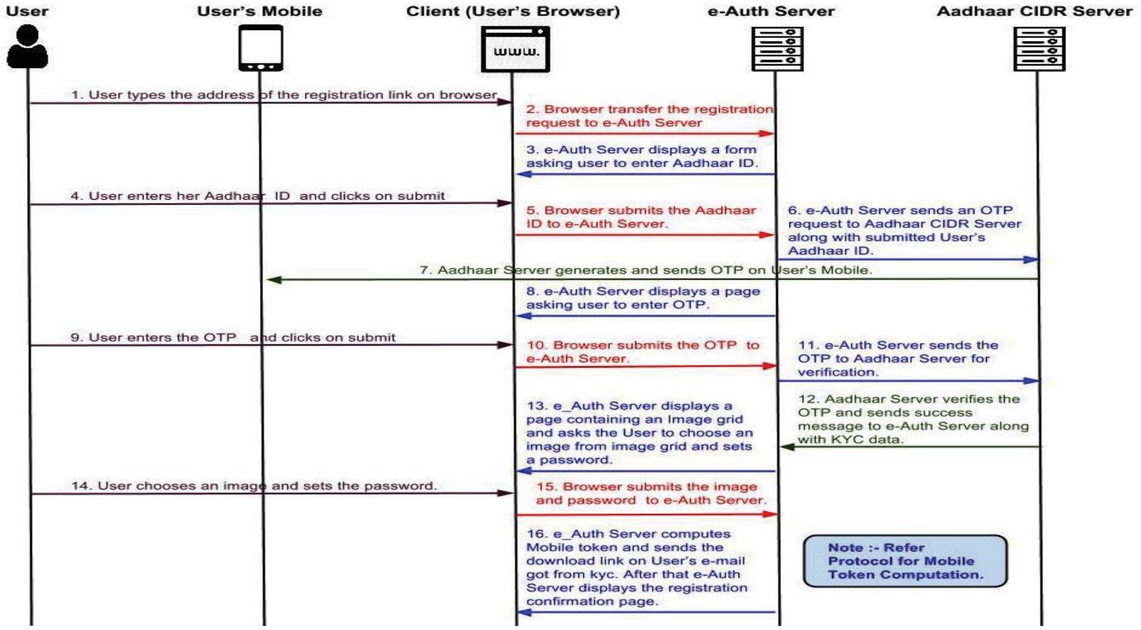

User
User's Mobile
Client (User's Browser)
www.
e-Auth Server
Aadhaar CIDR Server
1. User types the address of the registration link on browser
2. Browser transfer the registration request to e-Auth Server
3. e-Auth Server displays a form asking user to enter Aadhaar ID.
4. User enters her Aadhaar ID and clicks on submit
5. Browser submits the Aadhaar ID to e-Auth Server.
6. e-Auth Server sends an OTP request to Aadhaar CIDR Server along with submitted User's Aadhaar ID.
7. Aadhaar Server generates and sends OTP on User's Mobile.
8. e-Auth Server displays a page asking user to enter OTP.
9. User enters the OTP and clicks on submit
10. Browser submits the OTP to e-Auth Server.
11. e-Auth Server sends the OTP to Aadhaar Server for verification.
12. Aadhaar Server verifies the OTP and sends success message to e-Auth Server along with KYC data.
13. e_Auth Server displays a page containing an Image grid and asks the User to choose an image from image grid and sets a password.
14. User chooses an image and sets the password.
15. Browser submits the image and password to e-Auth Server.
16. e_Auth Server computes Mobile token and sends the download link on User's e-mail got from kyc. After that e-Auth Server displays the registration confirmation page.
Note :- Refer Protocol for Mobile Token Computation.

Fig 4: Registration Process of proposed Scheme

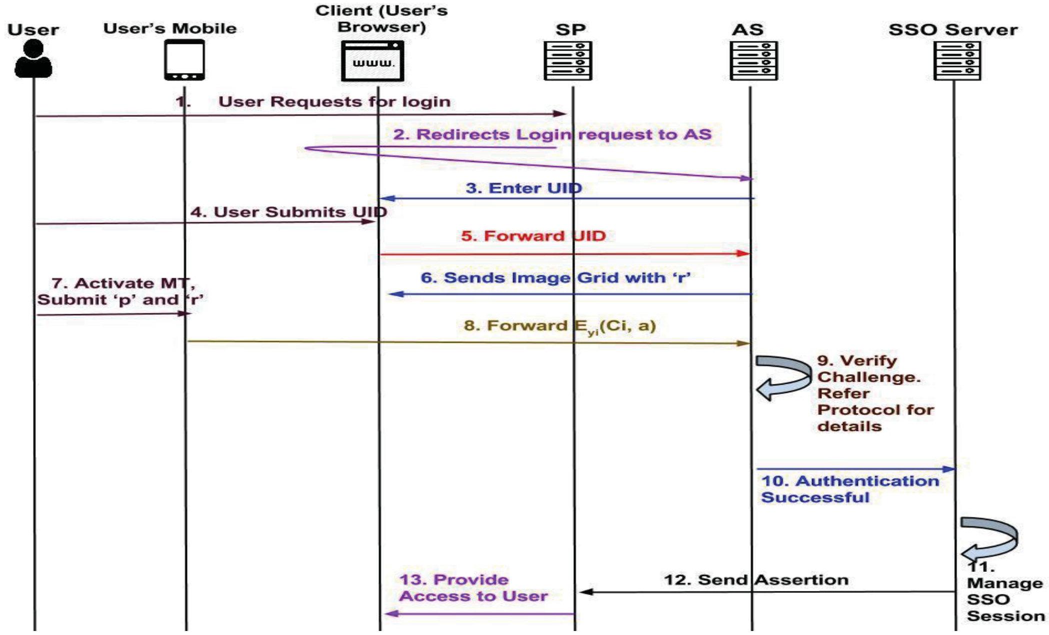

User
User's Mobile
Client (User's Browser)
www.
SP
AS
SSO Server
1. User Requests for login
2. Redirects Login request to AS
3. Enter UID
4. User Submits UID
5. Forward UID
6. Sends Image Grid with 'r'
7. Activate MT, Submit 'p' and 'r'
8. Forward E_yi(Ci, a)
9. Verify Challenge. Refer Protocol for details
10. Authentication Successful
11. Manage SSO Session
12. Send Assertion
13. Provide Access to User

Fig 5: Authentication Process of proposed Scheme