QR-SSO : Towards a QR-Code based Single Sign-On system

Syamantak Mukhopadhyay
School of Electronics and Computer Science
University of Southampton
Southampton, UK
sm19g10@ecs.soton.ac.uk

David Argles
School of Electronics and Computer Science
University of Southampton
Southampton, UK
da@ecs.soton.ac.uk

***Abstract*— Today internet users use a single identity to access multiple services. With single sign-on (SSO), users don't have to remember separate username/password for each service provider, which helps the user to browse through the web seamlessly. SSO is however susceptible to phishing attacks. This paper describes a new anti phishing SSO model based on mobile QR code. Apart from preventing phishing attacks this new model is also safe against man in the middle & reply attacks.**

Keywords-authentication; single sign-on; phishing; onetime password; QR code

I. INTRODUCTION

Internet is becoming more and more user centric each day. With the advent of web 2.0 internet users are becoming more inclined to use services from multiple content and service providers (CSP or SP). Most SPs provide user registration service whereby a user can create his/her own account and maintain it. As such a user has to maintain separate user accounts (username and password) for each of the SPs he/she uses. A study shows that today a typical user needs to maintain about twenty five different accounts which require password and uses eight of them in a given day[1]. Not only is this approach annoying to the user, it also raises some serious security questions, e.g. password fatigue [1].

Single sign-on(SSO) is one approach which aims to address the root cause of this problem[2]. With single sign-on, a user can create one account and use that account to login once and use multiple services hosted in different domains. There are three components or actors of a single sign-on system. A IdP or "identity Provider", a "Service Provider" (SP) or "Relying Party" (RP) and the user. RP relies on IdP(s) to authenticate user credentials. SSO approach outsources the responsibility of user authentication from service providers to IdPs. Not only does SSO reduces the burden of the user, a SSO system can also enable users to share contents between different service providers[3]. Many commercial solutions exist which provide SSO service, such as OpenID[4], Information Card[5] and SAML [6] based SSO Shibboleth.

In the next section we will go through these models and see their limitations.

II. SINGLE SIGN-ON PROCESS

A. *Existing Systems*

Shibboleth : Shibboleth is an open source single sign-on solution which is best suited for portal or Intranet applications[7]. It uses Security Assertion Markup Language (SAML), an OASIS specification for xml based security assertions. There are two main components of shibboleth system namely "*Identity Provider" (IdP)* and *"Service Provider" (Sp).*

In this system if a user wants to access a service or a resource of a SP, the SP redirects the user's web browser to a WAYF server. WAYF server displays a set of organizations to the user from which the user chooses one. Once the user chooses an organization, user's browser is redirected to corresponding login page of IdP and user provides his login credentials. Once the user successfully logs in, user's browser is redirected back to SP who decides whether to enable access for this user or not.

OpenID: OpenID[4] provides a user centric authentication model in a sense user can chose to implement his/her own OpenID provider or selects from a list of existing OpenID providers. Main components of an OpenID system are *User-Agent, Relying Party (RP)* and *OpenID Provider (IdP).*

User initiates the authentication process with RP using "User Agent". RP then, depending of user provided identifier, discovers user's IdP and redirects "User Agent" to IdP with an authentication request. User performs authentication at IdP side with username/password and IdP then redirects "User Agent" to RP again with a security assertion message specifying whether authentication has succeeded or failed. Based on this assertion RP decides whether to grant the user permission to access its services or not.

Major advantage of OpenID is it doesn't require any pre-established contract between RPs and IdPs. But it is susceptible to phishing attack[8].

Information Card: Information Card is based on real world multi identity concept[9]. Like Driving license, passport etc an Information Card user can have different identity sectors. Each

sector contains a different assertion which can be provided by different identity providers. When a user accesses services from a RP he/she logins in with one of the identity sectors or cards instead of username/password, with rest of the identity sectors remains hidden from RP.

Microsoft Cardspace is a SSO solution from Microsoft based on information card approach[10]. Information card approach provides more flexibility than username/password approach. As the information cards or sectors can be encrypted, it is also more secure than simple username/password. But with respect to OpenID, it is a very heavy system. Different Information Card identity sectors are stored in user computer which makes it susceptible to various security/privacy issues[8].

B. Phishing Vulnerability in SSO

Whoever there are certain security limitations of SSO systems. SSO involves crossing security domains between different SPs. Moreover most IdPs rely on username/password as their preferred authentication method. And as such it is susceptible to phishing attacks[11] [12] [13].

From the above discussion, login procedure of most of the Single Sign-On systems can be generalized to a set of common steps[14]. At the first phase user establishes unique identity by registering to an IdP. At the second phase user accesses services provided by different RPs by the following steps:

1. User requests a Service Provider or Relaying Party (say RP1) to access services provided by the RP1. RP1 initiates the SSO process.

2. RP1 redirects the request to Identity Provider (IdP1) in order to authenticate the user.

3. User authentication is done by username/password method or certificates.

4. IdP1 asserts user credentials to RP1 by sending an application ticket (AuthToken1) or assertion.

5. Based on the assertions RP decides whether to provide service to the user or not.

6. If the user wants to access services from another service provider RP2, RP1 will forward AuthToken1 to RP2. RP2 then can verify user credentials with IdP1 just by sending AuthToken1 to IdP1. User doesn't need to provide his/her credentials again.

Fugure 1. provides the pictorial representation of the steps.

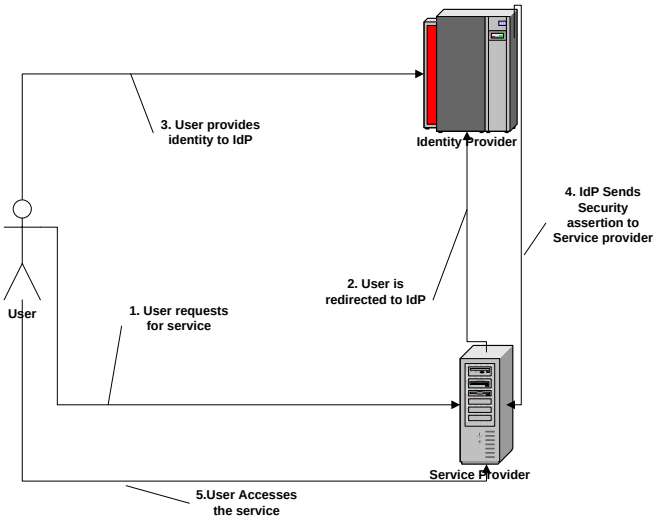

Figure 1. SSO general model

It is at stage 2 & 3 that the system is susceptible to phishing. Malicious RP can display a phishing page and as the user enters his/her username and password, RP can obtain details about user credentials. Since there is no way for the user to authenticate RP, he/she becomes vulnerable to phishing attack[15].

As can be seen most of the SSO systems are susceptible to "verifier impersonation" or phishing attacks as they use password for credential verification[16]. Phishing is a major issue in today's internet oriented life which causes massive financial loss every year[17]. It is therefore important to prevent phishing in SSO.

In the next section we will see previous works that have been done to prevent phishing in SSO systems.

III. PREVIOUS WORKS NO ANTI PHISHING MECHANISM FOR SSO

Various client side solutions exist which can detect a phishing page. E.g. Personal icon from myOpenID which can be used on a particular user's PC only. Solutions are provided by Verising (Validation Certificate for IE7 and seatbelt for Firefox) but they are browser dependent and not cost effective.

An improved SSO solution has been proposed by [18] based on Kerberos[19]. This model uses two passwords instead of one, one by authentication server and one by ticket granting server. Although this model adds one extra level of security, it is of little help to prevent phishing in a distributed web applications. Phishing web pages can be created to simulate this two phase approach and obtain both the passwords.

Another approach is to use mobile SIM in authentication phase of a SSO[20]. As proposed in this model, during login phase SIM is authenticated and proof of authentication is presented to the identity provider (IdP). IdP then lets the user login successfully. One drawback of this approach is that the authentication is carried out at the client side.

Lee & Jeun proposed a new approach to address the issue of phishing in SSO[15]. Every time user wants to access the service from a RP, a new token will be generated and will be sent to the user's email address. User then can login with the token. Although it solves the phishing problem, this solution breaks the basic philosophy of SSO. User needs to sign on to his/her email first to access the generated token. It can be thought of a SISO (single identity sign on) as it requires users to sign in twice.

You & Jun proposed a solution to phishing problem in SSO by using I-PIN[21]. But this solution can't be implemented globally.

We believe one possible solution to phishing problem in SSO is to use a onetime password approach. In the next section we will propose our solution for SSO with onetime password scheme.

IV. NEW SSO MODEL WITH ONETIME PASSWORD

Apart from addressing the issue of phishing in SSO, the new model should also be simple enough, so that it can be adopted in a real life scenario. In other words the proposed model should not introduce any new steps or complexity into the SSO process for the users. We will select our onetime password generation schema with these goals in mind.

PKI based onetime password generation model described by [22] provides good security features. But this model requires several additional steps during user registration and authentication phases.

For our proposed model we will make use of QR-code based onetime password schema[23]. Our idea is based on the assumption that most of the internet users today are equipped with a mobile phone that has a camera.

The proposed model has three entities: an Identity Provider (IdP), a Service Provider (SP) and a User (U_A). The model is divided in two phases: User Registration and User Verification.

TABLE I. NOTATIONS

Notation	Meaning
ID_A	Username or identity of the User
RP_A	Root password of the user
X_A	Secret key of the user
E_{QR}	Encoded QR code
D_{QR}	Decoded QR code

a. Notations used in this model.

A. *User Registration Phase*

1. During this phase U_A provides his/her ID_A and RP_A to IdP.
2. Based on RP_A, IdP calculates a key X_A using a one way hash function.
3. IdP sends X_A to user's mobile device and it gets stored as a secret key.

Fugure 2. below provides the pictorial representation of the steps.

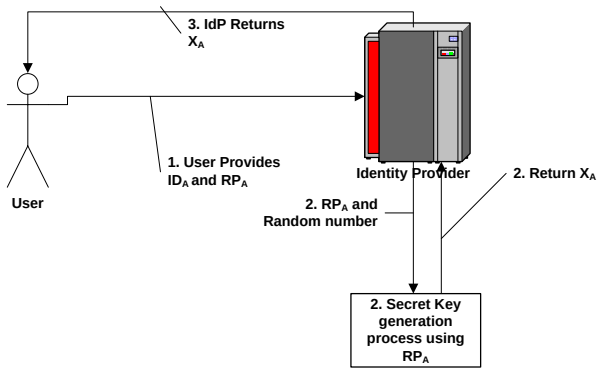

Figure 2. User Registration Phase

B. *User Verification Phase*

1. User wants to access a service provided by SP.
2. SP redirects user to its IdP.
3. User provides his/her identity to IdP.
4. IdP then based on user's identity calculates X_A. IdP uses X_A and a random number to calculate the QR code E_{QR}.
5. IdP then sends E_{QR} and a timestamp T1 to the user.
6. User uses the embedded camera in his/her mobile device and stored X_A to decode the QR code to D_{QR}. D_{QR} and timestamp T2 are then sent to the IdP.
7. IdP checks the validity of D_{QR} and acceptability of T2 and based on that sends a security assertion to SP.
8. SP then, based on the security assertion can allow or deny use of its services.
9. If the user now wants to access services from another service provider (SP1), SP can forward the assertion token to SP1 maintaining the basic principles of SSO.

Fugure 3. below provides the pictorial representation of the steps.

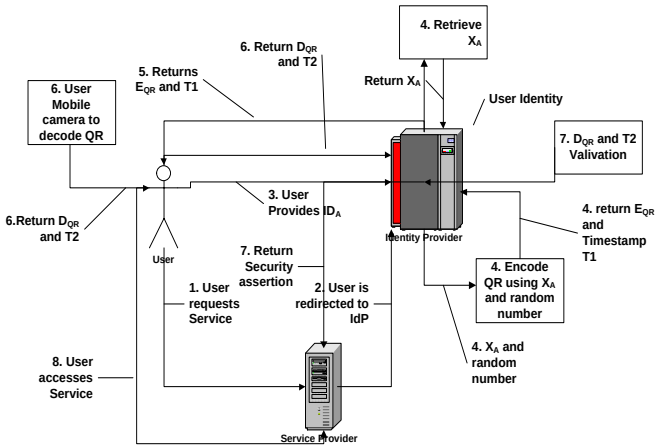

Figure 3. User Verification Phase

Generation of the secret key X_A should be dynamic. i.e if the user's X_A is compromised due to loss of mobile device or any other reason, he/she will be able to generate a new secret key. Since X_A is generated from RP_A, All the user needs to do is to reset his/her RP_A.

V. Proposed System Overview

As can be seen from the earlier discussion, the proposed single sign on model requires changes both in Identity Provider (or identity server) as well as in the client side to make use of the QR-Code. We will now discuss a formal model, named QR-SSO which will help realizing the new Single Sign-on model. In specific we will formally define the goals of the system in terms of features that we believe the proposed system must support at minimum.

A. *Identity Server Features*

This section defines the minimum set of features the proposed system must support in the identity server side.

1. The system must adhere to the basic principles of Single Sign-on. I.e. once the user logs in and the server generates an authentication token, subsequent request for user authentication must return the same authentication token.
2. Login process must support both user-id/password (root password) based login as well as QR-Code based login.
3. The system must not introduce any complicated steps in the user sign on process.
4. When logged in with root password, user must be able to regenerate his/her secret key.

B. *QR-Code Enabled Client Features*

This section defines the minimum set of features the proposed system must support in the client side.

1. Storage and resetting of secret key in the client side should not introduce complicated steps.
2. The process of sending decoded QR value to the identity server to the identity server should be automated if possible.

C. *Identity Server Architecture*

In conjunction with the features described, we now present the logical layer cake of the identity server. A block diagram of the architecture is depicted in figure 4.

1) *Data Access Layer*

The lowest layer in the identity server is the data access layer which includes an authentication database and a set of APIs to perform CRUD (create, read, update, delete) operations on the database. On top of the data access Layer two different login handlers are defined to handle both QR-Code based login and user-id/password (root password) based login.

2) *QR-Code Login Handler*

This component handles login process when login is initiated via QR-Code. The sub-components are-

a) User-Id to Secret key Mapper

This subcomponent maps the user-id with a secret key which are stored in the authentication database.

b) QR-Code Generator

As the name suggests, this sub-component contains the logic to generate QR-Code.

c) QR-Code Encoder

This sub-component encodes the QR-Code with the secret key of the user and also adds additional information to the QR-Code such as timestamp.

d) QR-Code Validator

This sub-component validates the decoded value sent by the user by comparing the decoded QR-Code and timestamp with the original values.

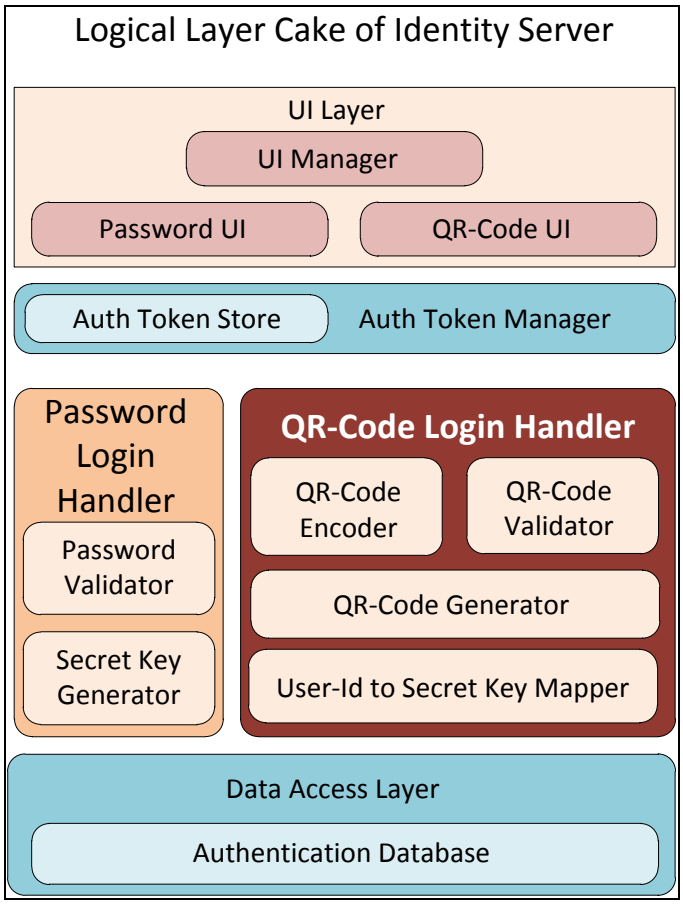

Figure 4. Logical Layer Cake of Identity Server

3) *Password Login handler*

This component handles the login process when it is initiated with root password. This component contains two sub-components, which are-

a) Password Validator

This sub-component validates the username/password of the user (common to all username/password based systems).

b) Secret Key Generator

This sub-component generates a new secret key for the logged-in user. Since it is under the control of "Password Login handler", secret key can only be regenerated when logged-in with root password.

4) Auth Token Manager

On top of the login handlers, we have an authentication token manager. This component is responsible for managing authentication tokens for logged-in users and can forward tokens to service providers if requested. This ensures that the Single Sign-on principles are not broken and a logged-in user is never requested to login again.

5) UI layer

At the top of the architecture we have a UI layer which contains three components.

a) UI Manager

This sub-component manages the UI flow of the login process and invokes either password based login UI or QR-Code based login UI based on the request type.

b) Password UI

This subcomponent handles the internal logic of password based login UI.

c) QR-Code UI

This subcomponent handles the internal logic of QR-Code based login UI.

D. QR-Code Enabled Client

As mention earlier, in this new SSO model a user can login to the identity provider with the QR-Code simply by using his/her mobile device. But in order to decode the encoded QR-Code properly the mobile device also needs to store and use the secret key of the user in the decoding process. Fortunately this can easily be done with a simple extension or plugin to the mobile device as supported by most mobile operating systems (e.g. Android, iOS).

In essence we can design a simple mobile plugin which would make the mobile device of the user capable of interacting with the identity server and decoding QR-Code. The secret key itself can be stored within mobile plugin's internal memory. The plugin must provide interface to the user to set the secret key when needed. It ensures that the plugin satisfies the first feature mentioned before. While the implementation details of the plugin may vary based on the mobile platform, the internal workings of the plugin can be described in the logical steps defined in table 1.

1. Get the Secret Key from Internal Store
2. Decode the QR Code
3. If the mobile is web enabled {
    a. Send the decoded value using https
4. }
5. Else {
    a. Display the decoded value to be entered manually.
6. }
7. Users logs in!

Table 1. Mobile plugin logic

As mentioned in table 1, the decoded value can be sent directly to the identity server provided that the mobile device is web enabled. This ensures that second feature required by the client is satisfied.

E. User Interaction

From the user perspective this new model is quiet simple to use. When the user is redirected to Identity Provider's login page, he/she needs to provide the identity (username) only.

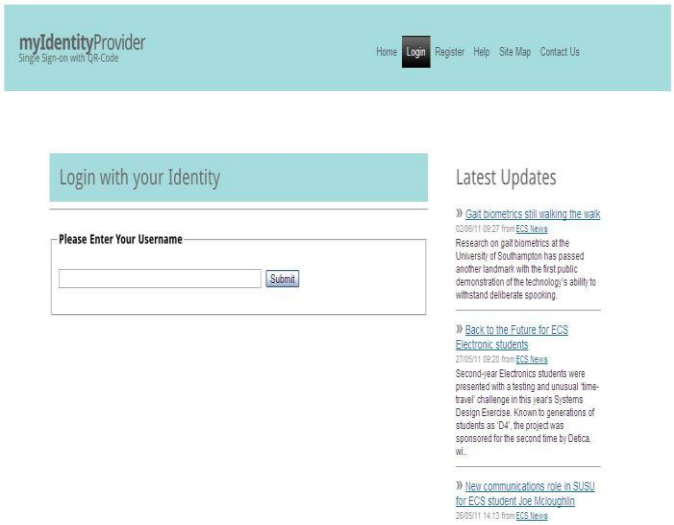

Figure 5. User Login step 1

Upon receiving the identity of the user, IdP generates the QR-Code using the mechanism described before.

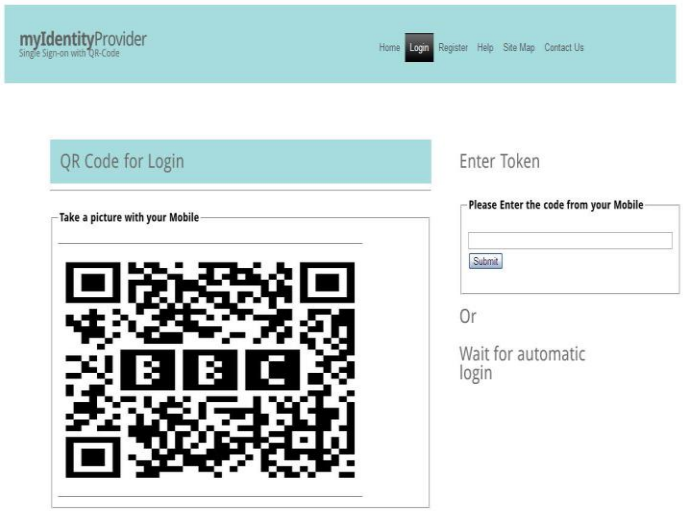

Figure 6. User Login step 2

The user then uses his/her mobile device to take a picture of the QR-Code. An app in user mobile device uses the secret key X_A already stored in it (during registration phase) to decode the QR-Code. If the user's mobile device is web enabled, the app can directly send the decoded value and timestamp to the IdP using HTTPS. Alternatively this decoded value will be displayed to the user who can then enter the value manually to login. In either case, this schema doesn't introduce any new complications for users.

VI. Security Analysis

A. *Phishing attack*

Since the root password RP_A is never disclosed during the verification phase, this model is fairly resistive to phishing attacks. Further if the secret key X_A is compromised at any stage, U_A can change it by simply resetting RP_A at the IdP side. Since X_A is generated using one way hash function, it is unfeasible to derive RP_A from X_A.

B. *Other attacks*

Assuming timestamp difference (T1-T2) acceptable by the IdP is minimum, If a man in the middle intercepts D_{QR} and tries to emulate the user, timestamp of D_{QR} would be expired. In addition, as E_{QR} is generated using a random number, after the allowed time interval IdP will select a new random number. Hence this model is fairly safe from both man in the middle attacks and reply attacks[23].

VII. Conclusion

In this paper we went through a brief overview of SSO process and analyzed its vulnerability against phishing attacks. We then presented a new SSO model with mobile QR code based onetime password schema. Security analysis of our model shows that apart from preventing phishing attacks, our model is safe against man in the middle & reply attacks as well. Based on this analysis we have proposed a formal system named QR-SSO which can be used to implement our model. This proposed system is simple from usability perspective and since most users today are equipped with camera embedded mobile device, this model can be adopted universally.

References

[1] D. Florencio and C. Herley, "A large-scale study of web password habits," in *16th International World Wide Web Conference, WWW2007, May 8, 2007 - May 12, 2007*, Banff, AB, Canada, 2007, pp. 657-666.

[2] R. Dhamija and L. Dusseault, "The Seven Flaws of Identity Management: Usability and Security Challenges," *Security & Privacy, IEEE,* vol. 6, pp. 24-29, 2008.

[3] S. San-Tsai, *et al.*, "Towards Enabling Web 2.0 Content Sharing beyond Walled Gardens," in *Computational Science and Engineering, 2009. CSE '09. International Conference on*, 2009, pp. 979-984.

[4] OpenID, "OpenID Authentication 2.0," December 5, 2007 2007.

[5] A. Nanda and M. B. Jones, "Identity Selector Interoperability Profile," July 2008 2008.

[6] Oasis, "Security Assertion Markup Language (SAML) V2.0," 15 March 2005 2005.

[7] "Internet2. Shibboleth System," *http://shibboleth.internet2.edu/*.

[8] S.-T. Sun, *et al.*, "OpenIDemail enabled browser: towards fixing the broken web single sign-on triangle," presented at the Proceedings of the 6th ACM workshop on Digital identity management, Chicago, Illinois, USA, 2010.

[9] "Identity Metasystem Interoperability Version 1.0 Committee Draft," *http://informationcard.net/specifications*, 10 November 2008 2008.

[10] "Microsoft Cardspace," *http://www.microsoft.com/windows/products/winfamily/cardspace/default.mspx*.

[11] E. Maler and D. Reed, "The Venn of Identity: Options and Issues in Federated Identity Management," *Security & Privacy, IEEE,* vol. 6, pp. 16-23, 2008.

[12] W. D. Yu, *et al.*, "A phishing vulnerability analysis of web based systems," in *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, 2008, pp. 326-331.

[13] Z. A. Khattak, *et al.*, "A study on threat model for federated identities in federated identity management system," in *Information Technology (ITSim), 2010 International Symposium in*, 2010, pp. 618-623.

[14] P. B. Tiwari and S. R. Joshi, "Single sign-on with one time password," in *Internet, 2009. AH-ICI 2009. First Asian Himalayas International Conference on*, 2009, pp. 1-4.

[15] L. HwanJin, *et al.*, "A New Anti-phishing Method in OpenID," in *Emerging Security Information, Systems and Technologies, 2008. SECURWARE '08. Second International Conference on*, 2008, pp. 243-247.

[16] O. Hyun-Kyung and J. Seung-Hun, "The Security Limitations of SSO in OpenID," in *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, 2008, pp. 1608-1611.

[17] D. Irani, *et al.*, "Evolutionary study of phishing," in *eCrime Researchers Summit, 2008*, 2008, pp. 1-10.

[18] J. Yang, "An Improved Scheme of Single Sign-On Protocol Based on Dynamic Double Password," in *Environmental Science and Information Application Technology, 2009. ESIAT 2009. International Conference on*, 2009, pp. 572-574.

[19] B. C. Neuman and T. Ts'o, "Kerberos: an authentication service for computer networks," *Communications Magazine, IEEE*, vol. 32, pp. 33-38, 1994.

[20] I. Jrstad, *et al.*, "Releasing the potential of OpenID & SIM," in *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on*, 2009, pp. 1-6.

[21] Y. Jae-Hwe and J. Moon-Seog, "A Mechanism to Prevent RP Phishing in OpenID System," in *Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference on*, 2010, pp. 876-880.

[22] H. C. Kim, *et al.*, "A Design of One-Time Password Mechanism Using Public Key Infrastructure," in *Networked Computing and Advanced Information Management, 2008. NCM '08. Fourth International Conference on*, 2008, pp. 18-24.

[23] L. Kuan-Chieh, *et al.*, "A One-Time Password Scheme with QR-Code Based on Mobile Phone," in *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, 2009, pp. 2069-2071.