Loxin – A Solution to Password-less Universal Login

Bo Zhu, Xinxin Fan and Guang Gong
Department of Electrical and Computer Engineering
University of Waterloo, Canada
{bo.zhu,x5fan,ggong}@uwaterloo.ca

***Abstract*—As the easiest and cheapest way of authenticating an end user, password based authentication methods have been consistently chosen by almost every new cloud service. Unfortunately, the explosive growth of cloud services and web applications has made it impossible for users to manage dozens of passwords for accessing different cloud services. The situation is even worse considering the potential application of massively parallel computing devices such as GPU and ASIC for efficient password cracking. Hence, from a usability viewpoint, passwords may have reached the end of their useful life.**

Motivated by a number of recent industry initiatives for online authentication, we present Loxin, an innovative solution for password-less universal login. Loxin aims to improve on passwords with respect to both usability and security. Loxin takes advantages of push message services for mobile devices and enables users to access multiple cloud services by using pre-owned identities, such as email addresses, together with few taps on their mobile devices.

In particular, the Loxin server cannot generate users' login credentials, thereby eliminating the potential risk of server compromises. Loxin is resistant to the most common attacks on cloud services such as replay attacks and man-in-the-middle attacks. We also discuss possible extensions for protecting Loxin from vendor lock-in and single point of failure, in order to ensure Loxin to be an open and stable authentication system. The application of the proposed Loxin security framework to the recent MintChip Challenge demonstrates the power of Loxin for building a real-world password-less mobile payment solution.

I. Introduction

With the advent of amazing cloud services and web applications, users frequently access services in their daily lives. Nowadays, we are likely to have more than ten accounts for computers, email accounts, websites, social networks, and various other cloud services, all with different passwords and security policies. Memorizing all passwords is both difficult and annoying, so people often end up in using simple passwords, or constantly forgetting less frequently used ones. It would be very useful if we can find an innovate way of accessing cloud services, which neither involves memorizing dozens of alphanumeric combinations, nor adds layers of complexity for users.

For password-based authentication methods, their security is mainly determined by the difficulty of guessing a user's password. Unfortunately, passwords usually have low entropy and are easier to guess than users think [2]. To further enhance the security of password-based web applications, a promising solution is to deploy a technology called *two-factor* or *multi-factor authentication*, in which a user is required to provide additional authentication information besides passwords. The second piece of information is typically generated by a physical token such as RSA SecurID [15] or a software application as Google Authenticator [12]. If different service providers set up their own two-factor authentication services, users may have to experience painful registration and login processes repeatedly.

A naive way to reduce users' burden for holding multiple passwords for different cloud services is to store users' credentials in a single device or service, and use certain key derivation functions to generate temporal passwords for sequential logins. However, this approach exposes the authentication server as a primary target of attackers. The other approach is to employ an Internet-scale identity system that defines standardized mechanisms enabling the identity attributes of its users to be shared between web applications and cloud services. A number of technologies and standards such as OpenID [13] and OAuth [5] have emerged to deliver an Internet-scale identity system during the past few years. The basic idea of those identity systems is to authenticate users with the aid of trusted Identity Providers (IDPs).

Recently, Bonneau *et al.* presented a comprehensive evaluation [1] for two decades of proposals to replace text passwords for general-purpose user authentication on the Internet. Their evaluation results have demonstrated the difficulty of replacing passwords and highlighted the research challenges towards designing a password-less login scheme. In this contribution, we propose Loxin, an innovative security framework for password-less universal login. After an initial registration process, Loxin enables a user to access multiple cloud services or web applications with only few taps on his/her mobile devices. This salient feature comes from the adoption of push message services for mobile devices and public-key cryptography. Different from most existing login solutions, the servers in Loxin are not able to generate users' credentials. Therefore, even if a Loxin server is compromised, attackers cannot impersonate a user in order to access cloud services. As a potential application of the Loxin security framework, we have applied it to build a password-less mobile payment solution for tackling the recent MintChip Challenge [16].

The remainder of this paper is organized as follows. Section II gives a detailed description of the Loxin design, followed by the security analysis of the Loxin framework in Section III. In Section IV, we discuss possible extensions of the Loxin security framework for a wide range of applications. Section V applies the Loxin security framework to tackle the MintChip Challenge, followed by the discussion of the related work in Section VI. Finally, we conclude this paper in Section VII.

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

II. Design of Loxin

This section describes the detailed design of Loxin, including the mechanisms to perform registration, authentication and revocation.

A. Architecture

The architecture of Loxin consists of the following components.

Loxin App
: An application installed on users' mobile devices.

Loxin Server
: A backend server for Loxin's service, which stores the registration information about the Loxin App.

Certificate Authority (CA)
: A trusted public-key certificate authority.

Identity Provider (IDP)
: A trusted identity provider, such as an email account provider.

Push Message Service (PMS)
: A third-party service that can send notifications to users' mobile devices. Such services include Google Cloud Message for Android and Apple Push Notification Service for iOS.

The adoption of the PMS makes the whole authentication process more convenient and user-friendly, but it is possible to complete the entire authentication process without the PMS. Possible extensions to achieve this will be discussed in Section IV.

B. Registration

Once the Loxin App is installed, it will perform a one-time registration process as illustrated in Fig. 1. The detailed steps are described below.

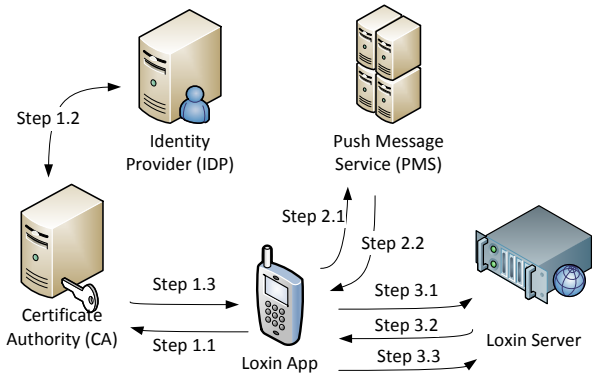

Fig. 1. Registration process of Loxin.

Step 1. Obtain a public-key certificate from CA.

Step 1.1 The Loxin App generates a pair of public key *PK* and private key *SK*. The Loxin App prompts the user to choose or enter an *ID* (e.g., email address) and then sends *ID* and *PK* to the CA.

Step 1.2 The CA first communicates with the IDP and verifies the user's *ID*, such as sending a verification email to the claimed address. This step is simplified in Fig. 1, since the details may vary for different providers.

Step 1.3 If the user's *ID* is verified, the CA sends its signed certificate $Cert(ID, PK)$, containing both *ID* and *PK*, back to the Loxin App.

Step 1 is only required to be completed once. After that, the user can log in to other cloud services by using this *ID*. Please note that the private key *SK* should be securely stored, and never be released outside the Loxin App.

Step 2. Register to a PMS.

Step 2.1 The Loxin App sends a registration request to a PMS.

Step 2.2 The PMS verifies the request and sends back credentials for registration, which can be used by other software and services to send messages to the Loxin App. Here we simply use a token *Tok* to represent all the PMS credentials.

Step 3. Register to the Loxin Server securely.

Step 3.1 The Loxin App sends a registration request, which contains $Cert(PK, ID)$ and *Tok*, to the Loxin Server.

Step 3.2 The Loxin Server responses with a random number R_{reg} and an expiration time T_{reg} for this request.

Step 3.3 The Loxin App signs *ID*, *Tok*, R_{reg} and T_{reg} with its private key *SK*. The signature

$$Sig_{\text{reg}}(ID, Tok, R_{\text{reg}}, T_{\text{reg}})$$

is sent to and verified by the Loxin Server. If the signature is valid, the Loxin Server stores the pair (ID, Tok) into its database for later use.

Steps 2 and 3 may need to be executed multiple times for updating *Tok* when the network environment changes. However, those steps can be performed in background without users' interactions.

C. Authentication

By using Loxin, users can authenticate their pre-owned identities to various cloud services even without pairing with or registering to those services first. This feature is able to remove or shorten registration processes and make cloud service more user-friendly.

When a user wants to log in to a cloud service from his/her computer using Loxin (see Fig. 2), a backend server of the cloud service will generate a random challenge for the user, and the Loxin Server will forward the challenge to the Loxin App via the PMS. Upon receiving the user's manual permission, the Loxin App will sign the challenge with the private key *SK* and send the signature to the cloud service for verification. The authentication process is illustrated in Fig. 2 and detailed below.

Step 1 The user enters and submits only *ID* to the cloud service.

Step 2 The cloud service generates a random number R_{auth}, an expiration time T_{auth}, and a callback address *URL* for this login request. In addition, a cryptographic hash value

$$tag = hash(ID, R_{\text{auth}}, T_{\text{auth}}, URL)$$

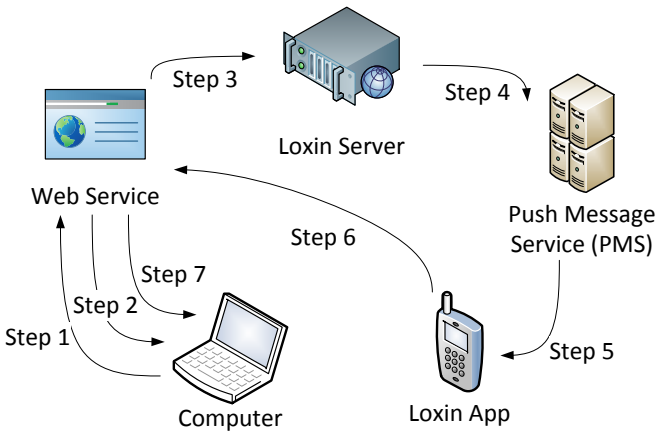

Fig. 2. Authentication process of Loxin.

is computed and displayed on the user's computer. The hash value may be represented by certain formats, such as figures or colorful barcodes, other than plain strings, so it can easily be visually checked by the user.

Step 3 The cloud service sends *ID*, R_{auth}, T_{auth}, and *URL* to the Loxin Server.

Step 4 The Loxin Server searches *ID* in its database in order to retrieve the corresponding *Tok*. The Loxin Server then uses *Tok* to send R_{auth}, T_{auth}, and *URL* to the PMS.

Step 5 The PMS forwards R_{auth}, T_{auth}, and *URL* to the user's Loxin App.

Step 6 The Loxin App recomputes the hash value *tag* based on the received *ID*, R_{auth}, T_{auth}, and *URL*. The Loxin App prompts the user to verify the correctness of basic login information, and compare the figures or barcodes shown on the computer and the Loxin App. Once *tag* and other information are approved, the Loxin App computes the signature

$$Sig_{\text{auth}}(ID, R_{\text{auth}}, T_{\text{auth}}, URL)$$

with the private key *SK*, and then sends Sig_{auth} and $Cert(ID, PK)$ to the cloud service's address *URL*.

Step 7 After verifying $Cert(ID, PK)$ and Sig_{auth}, the cloud service grants access to the user.

D. Revocation

When a user's phone is lost, the private key *SK* stored in the Loxin App might be compromised either. In this case, the user needs to contact the CA to revoke the certificate of the corresponding public key *PK*. For example, if the CA allows only one certificate for each *ID*, the user may go through the registration process (see Section II-B) again to revoke the old certificate.

Contacting the CA to revoke the lost certificate may be time-consuming, and the user's email account may be compromised as well once the mobile device is lost. One possible solution to secure the revocation process is generating a second pair of public and private keys, *PK'* and *SK'*, during the registration process. This second key pair should be stored out of the mobile device, e.g. printing on a paper, for security purpose. If the user's primary secret key *SK* may be leaked, the user can authenticate his/her identity by using *PK'* and *SK'* to the Loxin server to block further login requests associated with the leaked *PK*. *PK'* and *SK'* may also be used by the CA to verify users in order to revoke certificates.

In order to minimize the risk that the user's private key is used by adversaries, certain countermeasures may be deployed, e.g., requiring a short PIN to access the Loxin App and limiting the number of retrials. Please note that adding such a PIN will make the application less convenient, but it is still much more user-friendly than remembering and entering long passwords. Moreover, other information such as fingerprints and network locations may be considered to unlock the application instead of short PINs in the future, in order for improving usability and security.

III. Security Analysis

This section aims to analyze the security of Loxin. In addition, several methods are provided to further enhance the security of Loxin.

A. Against Man-in-the-Middle Attacks

In order to guarantee that the *tag* displayed on the computer is really the one generated by the cloud service provider, the Internet connection between the cloud service and the user's computer should be well protected by certain secure transport layer such as TLS. Next, *tag* shown by the Loxin App will be compared by the user to the one shown on the computer, and thus ensure both R_{auth} and T_{auth} are not replaced by any adversary in the middle.

Moreover, the disclosure of request information transmitted in the authentication process will not affect the security of the entire authentication protocol. As long as the *tag* shown on the web page is authenticated and matches with the one displayed on the Loxin App, the user will be successfully authenticated to the cloud service. Therefore, man-in-the-middle attacks cannot gain much benefits for attackers.

B. Against Replay Attacks

Both registration and authentication processes involve a random number to prevent adversaries from replay attacks. The random numbers are recommended to be at least 128 bits, such that it is infeasible for attackers to obtain two requests with a same random number in order to re-send the eavesdropped public-key signatures to impersonate the user.

Besides the random number, an expiration time is also included in the registration and authentication processes, which will keep the whole system safe even if a random number collision occurs in the long term.

C. Against Server Compromises

Since the private key *SK* never leaves the Loxin App, any backend server or cloud service does not have the knowledge of *SK*. Therefore, as long as the IDP and CA are secure, even if Loxin's backend servers are compromised, attackers will not be able to authenticate themselves to other cloud services.

D. Security Enhancements

One method to enhancing the security of Loxin is to sign the user's *ID* and public-key *PK* by multiple CAs. In this case, adversaries have to compromise all these CAs to generate a fake certificate. Additionally, if one CA does not update its revocation list promptly, cloud service providers can still check with other CAs. The other benefit is that the entire Loxin service will not be controlled by a single CA provider, a.k.a., vendor lock-in, since any CA works equivalently.

The other security enhancement is the public-key pinning, i.e., users' certificates are required to be signed by a small group of specific CAs. This will prevent dishonest CAs, whose root certificates have already been embedded in various operating systems, from creating fake certificates for Loxin.

If any users or organizations need a higher level of security, e.g., for protecting business secrets, hardware security modules (HSMs) can be used with Loxin. A HSM exposes only necessary interfaces, such as signature computation and verification, to operating systems and applications, which will minimize the possibility of leaking the private key *SK*.

E. Security Limitation

As we mentioned before, the Loxin system gives the ability of using one user's pre-owned *ID* to log in to other cloud services, and the *ID* has to be authenticated by the IDP during the registration process (see Fig. 1). For example, if one uses an email address as *ID*, the address may be authenticated via the email service provider to the CA. Therefore, the security of the Loxin system still relies on the trustworthiness of the IDP. In this sense, the security of Loxin is similar to that of OpenID. Nevertheless, Loxin allows cloud service providers to directly verify the users' signatures by public-key certificates without the help of the IDP, which improves scalability and reduces network protocol latency. Moreover, the protocols of Loxin enable a user to securely log in from multiple devices easily with one smart phone.

IV. Extensions

This section presents several methods to extend the original design of Loxin for a wide range of applications.

A. Two-Factor Authenticator

Loxin is fully compatible with traditional password-based authentication schemes, which means even if users initially do not trust the security of the Loxin system, they can still use Loxin as a convenient security enhancement, i.e. a two-factor authenticator.

This may help solving the adoption problem of early stages. Service providers can first add Loxin as a two-factor security enhancement, and then give users the option to use Loxin as a single authentication method.

B. Local Authentication

Typing passwords is particularly painful on the relatively small screen of a smart phone. The Loxin App can also be used to authenticate other applications installed on the smart phone. In this special case, the authentication process of Loxin can be executed locally without involving the Loxin Server or PMS. An application can broadcast a local login request within the phone, and once the Loxin App receives the request it can reply a proper signature upon the user's approval.

C. Authentication via Barcode

If the Loxin Server or PMS is offline, the authentication request from the cloud service will not reach the user in time. In this case, the cloud service can display a barcode (e.g., a QR code) to the user on the computer, which contains all the necessary information about the request. After scanning the barcode, the Loxin App can send the authentication signature to the cloud service directly. This method prevents the whole authentication process from the potential single point of failure of the Loxin Server.

D. Pairing without ID

It is possible to use Loxin service even without first telling the user's *ID* to cloud service providers. For example, after scanning the barcode as described in the previous subsection, the Loxin App will send the user's public-key certificate along with the signature, and then the cloud service can retrieve *ID* from the certificate. Thus the user does not need to manually enter *ID* during the entire authentication process. In the original design in Section II-C, it is possible to utilize some other factors, such as geographic and network information, to pair the Loxin App with the cloud service.

V. Loxin in Practice – Tackling the MintChip Challenge

In this section, we apply the Loxin security framework to build a password-less mobile payment solution called EasyChip for tackling the real-world MintChip Challenge [16] organized by the Royal Canadian Mint. With Loxin in place, a user can complete online transactions without creating additional accounts with multiple merchants, thereby offering an innovative password-less online payment service.

A. The MintChip Challenge

In 2012, the Canadian federal government announced in its budget that it would withdraw the penny from circulation in the fall of 2012. As a quick response, the Royal Canadian Mint unveiled its digital alternative called MintChip [16] to coinage and small bank denominations, and simultaneously launched the MintChip Challenge contest to encourage development of novel applications based on MintChip.

A MintChip is a secure smart card chip that can be encapsulated into different form factors (e.g., a MicroSD card) for easier connection to computers and mobile devices. The MintChip securely holds electronic money and enables a protocol to transfer it from one chip to another.

B. The EasyChip Solution

To tackle the MintChip Challenge, we have developed EasyChip [9], an Android application for password-less mobile payment based on the Loxin security framework in Section II. Using the EasyChip application on a smart phone, a password-less payment process works as described below.

1) Registration: In the Loxin framework, the Loxin App needs to first obtain a public-key certificate from CA. However, the MintChip inside a smart phone has already contained a unique 64-bit MintChip ID, a preloaded private/public RSA key pair, and the associated X.509 public-key certificate issued by the MintChip CA. Therefore, Steps 1.1 – 1.3 in the Loxin registration procedure can be omitted. Secondly, the Loxin App selects/creates an exiting/new email account and registers it to the Google Cloud Messaging for Android for the push message service. Finally, the Loxin App registers to the Loxin Server with the email account, the MintChip ID, the MintChip certificate, and the push message service token as described in Steps 3.1 – 3.3 of the Loxin security framework.

2) Authentication and Payment: A complete MintChip payment always involves two MintChip devices, namely a sender and a receiver. Moreover, the receiver's MintChip ID must be known by the sender. When a customer (i.e., a sender) wants to purchase a product from a merchant website (i.e., a receiver), the customer first enters the email address associated with the EasyChip App.

The merchant's web server, which is equipped with another MintChip, generates a MintChip Request message that contains the information such as the receiver's MintChip ID, the amount to pay, a URL specifying where the payment should be sent to, a random challenge, etc. The MintChip Request message and the customer's email address will be sent to the Loxin Server. Upon receiving the message, the Loxin Server looks up its database with the customer's email address and retrieves the push message service token. The Loxin Server then pushes the MintChip Request message to the customer's smart phone through the PMS.

When the customer confirms the payment request, the MintChip inside the customer's smart phone will immediately generate a signed MintChip Value message using the RSA signature scheme and send it back to the merchant's web server. After verifying the received MintChip certificate and digital signature, the payment will be processed. Note that the entire authentication and payment processes follow the Loxin security framework, and the customer does not need to enter any password.

VI. Related Work

In this section, we discuss several related products as well as the competitive advantages of Loxin.

A. RSA SecurID

RSA SecurID is a well-established product in the two-factor authentication market, which is a hardware token with a small screen showing a pseudo-random authentication code in every minute [15]. Each RSA SecurID shares a secret seed with its backend server. When a user submits the authentication code to a cloud service, the service provider will compute the number based on their own knowledge of the secret seed and then compare it with the one submitted by the user.

When compared to the design of Loxin, if the servers of RSA SecurID are compromised, attackers can compute any pseudo-random authentication codes after obtaining their secret seeds. This kind of incidents did happen in 2011 [3], which renders RSA SecreID less effective to serve as a secure two-factor authentication mechanism. Moreover, RSA SecurID also has a usability issue and users have to carry the extra hardware device. In addition, different cloud services usually do not share an identical secret seed, so user may be required to have multiple devices associated with various service providers.

B. Google Authenticator

Google Authenticator [12] is a software solution to the usability issue of RSA SecurID. It replaces the hardware device of RSA SecurID by a software application on users' mobile devices, and can be paired with many service providers such that users do not need to carry multiple devices.

However, Google Authenticator still shares seeds with its backend servers, and is required to be manually paired with each service provider similarly as RSA SecurID, which is not user-friendly when compared to Loxin.

C. Kerberos

Kerberos is a symmetric-key cryptography based protocol that allows users authenticate their identities to services by the help of a central Kerberos server [6]. A *ticket* will be issued by the central server for a specific service when the user wants to access the service.

Kerberos apparently suffers from single point of failure of the central Kerberos server. In addition, although a public-key cryptography based initial authentication extension is proposed in [10], the *ticket* issued in Kerberos system is still produced by symmetric-key algorithms. Thus once the database of the Kerberos server is compromised, the credentials of all users will be in danger.

D. Pico

Pico is a hardware solution proposed by Stajano in 2011 [8], which serves as a replacement of password authentications. Pico is recommended to be a dedicated device with capabilities such as camera and radio. It is hard to manufacturer and users are required to carry it all the time. Moreover, Pico has to be paired with each application in a similar way as RSA SecurID and Google Authenticator.

E. Twitter's Two-Factor Authentication

Recently, Twitter has upgraded its mobile applications to support a public-key cryptography based two-factor authentication solution [7], which has a similar idea as Loxin in the sense that the web server sends a login challenge to the user and requires it to be signed by the private key stored in the smart phone application.

As mentioned in [17], the design of Twitter's two-factor authentication mechanism has a security hazard that users cannot tell the differences between the fake login requests initiated by adversaries and the real ones by the users themselves, since the smart phone application does not provide the user with detailed information about login requests. The hash value *tag* used in the Loxin system can be adopted to defeat this kind of attacks. Moreover, the public key is only paired with Twitter,

which is similar to the method of Pico. To provide single-sign-on service to other service providers, the public key needs to be properly signed by trusted third-party CAs.

F. Mozilla Persona

Persona (formerly BrowserID) is a decentralized single-sign-on system developed by Mozilla for users and websites to release the burden of creating and managing passwords [14]. Persona adopts users' email addresses as identities and issues public-key certificates for these emails.

However, the design of Persona aims to provide in-browser solution and stores the public-key certificate in the local space of a browser. Therefore, to use on multiple devices, Persona may need to be set up many times, which is not as convenient as Loxin. With the help of push message services, Loxin allows a user to store his/her private key in a smart phone and access many services on multiple computers or devices.

G. PhoneAuth

PhoneAuth is a user-friendly two-factor authentication mechanism proposed in 2012 [4]. The login request is automatically signed by the smart phone application, if the user's smart phone is present and can be connected to the computer via Bluetooth. The whole two-factor authentication process does not need the user's interaction.

However, PhoneAuth requires the web browser to be capable of sending data to the user's smart phone via Bluetooth connection. In [4], the authors managed to achieve this function by developing an extension for the Chromium browser. The regular browsers without any modifications do not have such abilities, and it would be dangerous to open a web interface to physically access users' smart phones.

H. Duo Push

Duo Push is a commercial software application developed by Duo Security [11], which aims to provide a two-factor authentication with push message capabilities. However, the design details of Duo Push are not disclosed. Moreover, the authentication status of a user in Duo Push depends on the response from the verification servers of Duo Security, which makes Duo Push unsuitable for replacing password-based authentication solutions developed by other services and companies. Furthermore, the systems integrated with Duo Push may have the single point of failure. In this case, users cannot access cloud services when the verification servers of Duo Security are not working properly or being compromised.

VII. Conclusions

In this paper, we propose an authentication system called Loxin. We demonstrate that Loxin can be used to replace traditional universal authentication systems based on passwords, and it is secure against man-in-the-middle attacks and replay attacks. In particular, even if the servers of Loxin are compromised by attackers, the private keys of users are still safe and thus attackers cannot impersonate the users. This salient feature makes Loxin an attractive security solution for password-less web authentication. Several methods have been proposed to extend Loxin for different use cases, and to avoid single point of failure and vendor lock-in. We have also developed EasyChip, an Android application of the Loxin security framework, to demonstrate the power of Loxin for building a real-world password-less mobile payment solution.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. The research is supported by NSERC Discovery Grant.

References

[1] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano. *The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes*. IEEE Symposium on Security and Privacy - S&P 2012, pp. 553-567, IEEE Computer Society, 2012.

[2] L. S. Clair, L. Johansen, W. Enck, M. Pirretti, P. Traynor, P. McDaniel, and T. Jaeger. *Password exhaustion: Predicting the end of password usefulness*. Information Systems Security, pp. 37-55, Springer Berlin Heidelberg, 2006.

[3] A. Coviello. *Open Letter to RSA Customers*, 2011, available at https://www.sec.gov/Archives/edgar/data/790070/000119312511070159/dex991.htm.

[4] A. Czeskis, M. Dietz, T. Kohno, D. Wallach, and D. Balfanz. *Strengthening user authentication through opportunistic cryptographic identity assertions*. In Proceedings of the 2012 ACM conference on Computer and communications security, pp. 404-414, ACM, 2012.

[5] D. Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749, Internet Engineering Task Force (IETF), 2012.

[6] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer. *Kerberos: An authentication service for computer networks*. In Project Athena Technical Plan, 1987.

[7] A. Smolen. *Login verification on Twitter for iPhone and Android*. Twitter, Inc., 2013, available at https://blog.twitter.com/2013/login-verification-on-twitter-for-iphone-and-android.

[8] F. Stajano. *Pico: No More Passwords!*. The 19th International Workshop on Security Protocols Workshop, LNCS 7114, B. Christianson et al. (eds.), Berlin, Germany: Springer-Verlag, pp. 49-81, 2011.

[9] B. Zhu and X. Fan. *EasyChip*, 2012, available at http://mintchipchallenge.com/submissions/9469-easychip.

[10] L. Zhu and B. Tung. *Public key cryptography for initial authentication in Kerberos (PKINIT)*. RFC 4556, Internet Engineering Task Force (IETF), 2006.

[11] *Duo Push: One-Tap Authentication*, Duo Security, Inc., available at https://www.duosecurity.com/duo-push.

[12] *Google Authenticator Project – Two-Step Verification*, Google Inc., available at http://code.google.com/p/google-authenticator/.

[13] *OpenID Authentication 2.0 - Final*, OpenID Community, 2007, available at http://openid.net/specs/openid-authentication-2_0.html.

[14] *Persona Protocol Overview*, Mozilla Developer Network and individual contributors, available at https://developer.mozilla.org/en-US/docs/Mozilla/Persona/Protocol_Overview.

[15] *RSA SecurID Hardware Authenticators*, RSA Inc., available at http://www.emc.com/security/rsa-securid/rsa-securid-hardware-authenticators.htm.

[16] *The MintChip Challenge*, The Royal Canadian Mint, 2012, available at http://mintchipchallenge.com/.

[17] *Thoughts on Twitter's new Two-Factor Authentication*, Authy, 2013, available at http://blog.authy.com/twitter.