Loxin – A Universal Solution to Password-Free Login

Bo Zhu, Xinxin Fan, and Guang Gong

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
{bo.zhu,x5fan,ggong}@uwaterloo.ca

Abstract. As the easiest and cheapest way of authenticating an end user, password based approach has been consistently chosen by implementers of every new computer or mobile device based web service. Unfortunately, the explosive growth of web applications has made it impossible for users to manage dozens of passwords for accessing different web services. The situation is even worse considering the potential application of massively parallel computational devices such as general purpose GPUs and FPGA arrays for efficient password cracking. Hence, from a usability viewpoint, passwords have reached the end of their useful life.

Motivated by a number of recent industry initiatives for online authentication, we present Loxin, an innovative and universal solution for password-free login. Loxin aims to improve on passwords with respect to both usability and security. Loxin takes advantages of popular push message services on mobile devices and enables users to access multiple web services using pre-owned identities such as email addresses in the system together with few taps on their mobile devices. In particular, the Loxin server cannot generate users' login credentials for web access, thereby eliminating the potential risk of server compromise. The security analysis shows that Loxin is resistant to the most common attacks on web services such as replay attacks, man-in-the-middle attacks, and server compromise attacks. The application of the Loxin security framework to the recent MintChip Challenge demonstrates the power of Loxin for building a real-world password-free mobile payment solution.

Keywords: Loxin, authentication, password, mobile device, security

1 Introduction

With the advent of amazing web applications on the Internet, users frequently access web services in their daily lives. Nowadays, we are likely to have more than 10 accounts for computers, email accounts, websites, social networks, and various other 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 forgetting their permutations. These practices open the convenient door for hackers, especially when we conduct online transactions using computing devices. What we really need today is an innovate way of accessing web services that does not involve memorizing dozens of alphanumeric combinations, and does not add layers of complexity for users.

In the password-based authentication, the security is determined by the difficulty of guessing a user's password. Unfortunately, passwords usually have low entropy and are easier to guess than users think. To further enhance the security of password-based web applications, a promising solution is to deploy a technology so-called *two-factor or multi-factor authentication*, in which a user is required to provide more than a single authentication factor, typically a user's password. The second piece of authentication information is typically generated by a physical token like a RSA SecurID [1] or a mobile device with the Google Authenticator application [2]. While the two-factor authentication is able to enhance the security of web access significantly, different service providers may have to set up their own two-factor authentication services and users have to experience painful registration and login processes.

A naive way to reduce the user's burden for holding multiple passwords for different web-based services is to store users' access credentials in a single server, and use certain key derivation functions to

generate temporal passwords for sequential logins. However, this approach exposes the authentication server as the 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 applications and web servers. A number of technologies and standards such as OpenID [3] and OAuth [4] 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). Although the Internet-scale identity systems provide federated single sign-on services, the adoption of them is quite limited since many websites are willing to act as IDPs but few are willing to accept existing identity systems as relying parties.

Recently, Bonneau *et al.* [5] presented a comprehensive evaluation for two decades of proposals to replace text passwords for general-purpose user authentication on the web. Their evaluation results have demonstrated the difficulty of replacing passwords and highlighted the research challenges towards designing a password-free login scheme. In this contribution, we propose Loxin, an innovative and universal security framework for password-free login. After an initial registration process, Loxin enables a user to access multiple web services with only few clicks on his/her mobile devices. This salient feature comes from the usage of popular push message services on mobile devices. Different from most existing login solutions, the sever in Loxin is not able to generate users' authentication credentials for web access. Therefore, even if the server is compromised in Loxin, an attacker cannot impersonate a user to access web services. As a potential application of the Loxin security framework, we have applied it to build a password-free mobile payment solution for tackling the recent MintChip Challenge.

The remainder of this paper is organized as follows. Section 2 gives a detailed description of the Loxin design, followed by the security analysis of the Loxin framework in Section 3. In Section 4, we discuss possible extensions of the Loxin framework for a wide range of applications. Section 5 applies the Loxin framework to tackle the MintChip Challenge, followed by the discussion of the related work in Section 6. Finally, we conclude this paper in Section 7.

2 Design of Loxin

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

2.1 Architecture

The architecture of Loxin consists of the following four components.

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

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

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

Push Message Service (PMS). A third-party service that can send notifications to users' mobile devices. Such services include Google Cloud Message for Android [13] and Apple Push Notification Service for iOS [14].

2.2 Registration

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

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

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

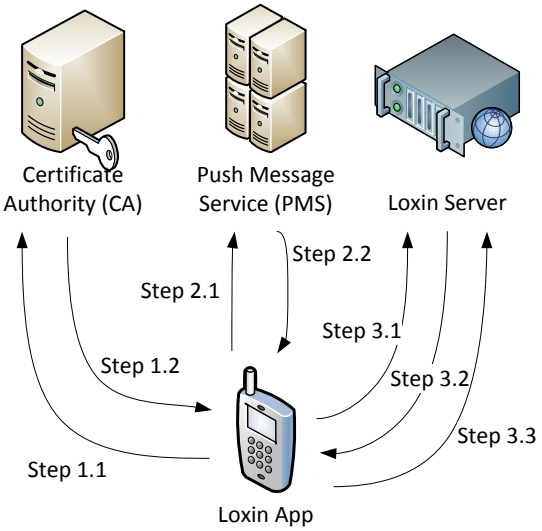

Fig. 1. Registration process of Loxin.

Step 1.2 The CA first verifies the user's ID (e.g., sending a verification email to the claimed address). If the user's ID is verified, the CA sends its signed certificate $Cert(ID, PK)$, containing both ID and PK, back to Loxin App.

This step is only required to be conducted once. Please note that the private key SK should be securely stored and never be released outside Loxin App.

Step 2. Register to a PMS.

Step 2.1 Loxin App sends a registration request to a PMS.
Step 2.2 The PMS verifies the request and sends back credentials for the registration, which can be used by other softwares and services to send messages to Loxin App. Here we simply use a token Tok to represent all the credentials.

Step 3. Register to Loxin Server securely.

Step 3.1 Loxin App sends a registration request, which contains $Cert(PK, ID)$ and Tok, to Loxin Server.
Step 3.2 Loxin Server responses with a random number R_{reg} and an expiration time T_{reg} for this request.
Step 3.3 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 Loxin Server. If the signature is valid, Loxin Server stores the pair (ID, Tok) into its database for later use.

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

2.3 Authentication

A significant difference between Loxin and other related products [1, 2, 6] is that, by using Loxin, users can authenticate themselves to various web services even without pairing with or registering to these

services first. This feature will remove or shorten registration processes and make web services more user-friendly.

Whenever a user wants to login a website from his/her computer using Loxin (see Figure 2), the authentication process is conducted as follows.

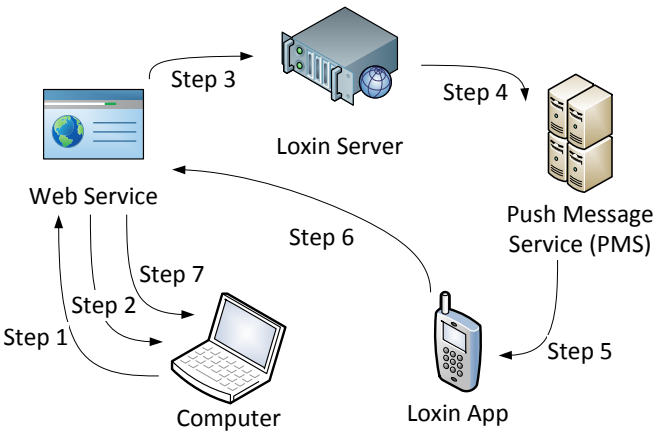

Fig. 2. Authentication process of Loxin.

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

Step 2 The web 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

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

is computed and displayed on the user's computer.

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

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

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

Step 6 Loxin App recomputes the hash value tag based on the received ID, R_{auth}, T_{auth}, and URL. Loxin App requests the user to compare tag with the one shown in on the computer, and to verify the correctness of other basic information (see Figure 3 for an example). If tag and other information are verified and approved, 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} along with $Cert(ID, PK)$ to the web service's address URL.

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

2.4 Revocation

When a user's phone is lost, the private key SK stored in Loxin App might also be compromised. Under such circumstance, the user needs to contact the CA to revoke the certificate of the corresponding public key PK. For example, if the CA only allows one certificate for each ID, the user may go through the registration process (see Section 2.2) again to revoke the old certificate. For minimizing the risk that the user's private key is leaked and used by adversaries, certain countermeasures can be deployed, e.g., requiring a short PIN to access Loxin App and limiting the retrial times.

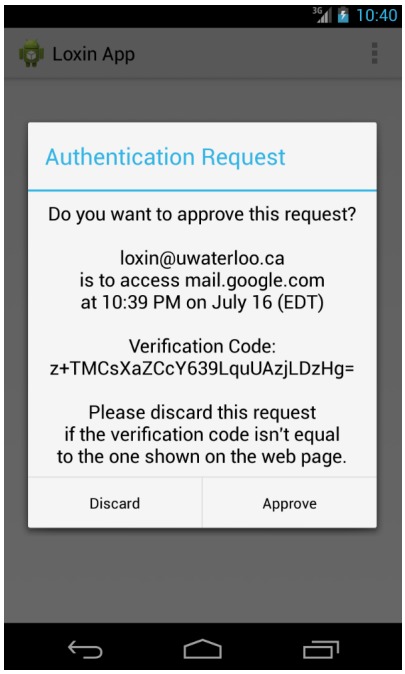

Fig. 3. A confirmation dialog of Loxin App.

3 Security Analysis

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

3.1 Preventing Man-in-the-Middle Attacks

In order to guarantee that the *tag* displayed on the computer is correct, the Internet connection between the web service and the user's computer should be well protected by certain security transport layer like TLS. Note that the hash value *tag* shown by Loxin App is checked by the user manually, which will ensure both R_{auth} and T_{auth} are not replaced by an adversary in the middle. Therefore, the authentication process of Loxin can defeat man-in-the-middle attacks. In addition, 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 on Loxin App, the user will still be successfully authenticated to the web service.

3.2 Preventing Replay Attacks

Both registration and authentication processes involve a random number and an expiration time to prevent adversaries from replay attacks, i.e., re-sending the eavesdropped messages to impersonate the user.

3.3 Defeating Attacks on Servers

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

3.4 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, web 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., the user's certificate is required to be signed by a small group of specific CAs. This will prevent dishonest CAs, whose public keys have already been embedded in various operating systems, to generate fake certificates for Loxin.

If 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 minimizes the possibility of leaking the private key *SK*.

4 Application Extensions

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

4.1 Two-Factor Authenticator

If users do not trust the security of the Loxin system, they can still use Loxin as a convenient two-factor authenticator, which is compatible with traditional password-based authentication schemes.

4.2 Local Authentication

Typing passwords is particularly inconvenient on the relatively small screen of a smartphone. Loxin App can also be used to authenticate other applications installed on the smartphone. In this special case, the authentication process in Loxin can be executed locally without involving Loxin Server or PMS. An application can send a local login request to Loxin App and then receives a proper signature as a response.

4.3 Authentication via Barcode

If Loxin Server or PMS is offline, the authentication request from the web service will not reach the user in time. In this case, the web 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, Loxin App can send the authentication signature to the web service directly. This method prevents the Loxin system from the potential single point of failure of Loxin Server.

4.4 Pairing without ID

It is possible to use Loxin service even without first inputing *ID* to the web service. For example, after scanning the barcode as described above, Loxin App will send the user's public-key certificate along with the signature, and then the web 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 2.3, it is possible to utilize some other factors, such as geographic and network locations used in Bump API [21], to pair Loxin App with the web service.

4.5 Push to Browser

To implement the last step of the authentication process in Loxin (i.e., Step 7 in Figure 2), the web service may have to maintain a long-live connection with the user's computer. After verifying the signature sent from Loxin App, the web service needs to authenticate the user actively from the server side by the push technology, such as Comet [7] and Channel API of Google App Engine [8]. Otherwise the client is required to periodically poll the web server status. Either the push technology or long-polling will increase the load and complexity of the web service.

For user authentication in web browsers, the aforementioned issue may be solved by leveraging the push message APIs for different browsers, such as the pushMessaging API for Chrome App [9] and the SimplePush API for Firefox [10]. Instead of sending the signature from Loxin App to the web service directly, the signature may be sent to the user's web browser via these push message APIs, which then forwards the signature to the web service. In addition, the request to Loxin Server (see Step 3 in Figure 2) can also be initiated from the user's computer, which will further reduce the web service's burden. In this case, the web service can stay largely unchanged and the entire process is shown in Figure 4.

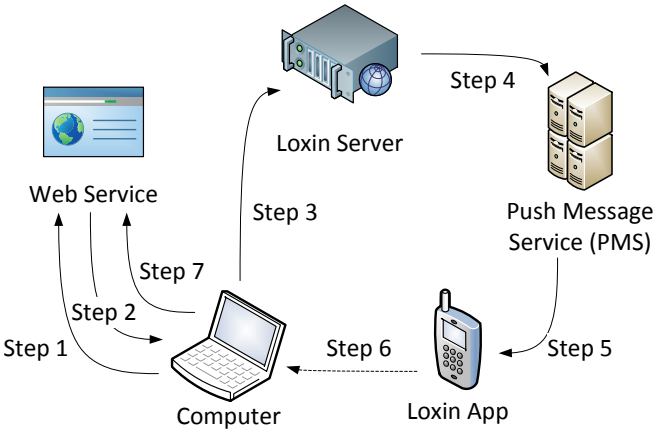

Fig. 4. Authentication process that saves web service cost.

5 Loxin in Practice – Tackling the MintChip Challenge

In this section, we apply the Loxin security framework to build a password-free mobile payment solution called EasyChip for tackling the real-world MintChip Challenge [15] 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-free online payment service.

5.1 The MintChip Challenge

On March 29, 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 for MintChip [15].

A MintChip, as illustrated in Figure 5, 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. The main goal of the MintChip is to facilitate small-value transactions, such as micro-transactions (under $10) and nano-transactions (under $1). Unlike existing digital wallets [17–19] where customers' financial information (e.g., credit/debit card) is stored into a fully embedded secure element or in the cloud, MintChip does not have any link to your bank account or credit card and no personal data is exchanged during a transaction.

Fig. 5. A pair of MintChips (centre) and accessories from the Royal Canadian Mint.

5.2 The EasyChip Solution

To tackle the MintChip Challenge, we have developed the EasyChip [20], an Android application for password-free mobile payment based on the Loxin security framework in Section 2. Using the EasyChip application on a smartphone, a password-free payment process works as described below.

Registration In the Loxin framework, Loxin App needs to first perform a registration process (i.e., obtain a public-key certificate from CA). However, the MintChip inside a smartphone has already contains 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, the first steps (i.e., Steps 1.1 and 1.2) in the Loxin registration procedure can be omitted. Secondly, Loxin App selects/creates an exiting/new email account and registers it to the Google Cloud Messaging for Android (GCM) [13] for the push message service. Finally, Loxin App registers to 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.

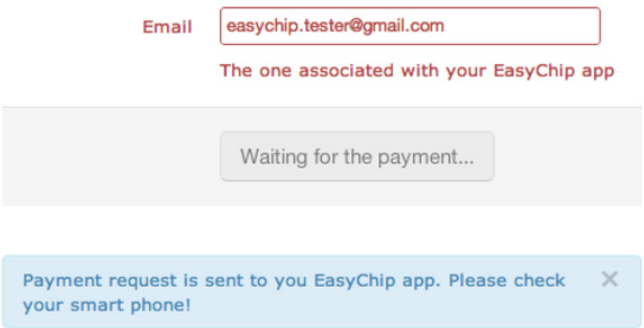

Fig. 6. A customer inputs the email address.

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 will first input the email address associated with the EasyChip App, as shown in Figure 6.

The merchant's web server, which is equipped with another MintChip, generates a MintChip Request message[1] [16] that contains the information such as the receiver's MintChip ID, the amount to pay, a URL specifying where the payment should be sent, a random challenge, etc. The MintChip Request message and the customer's email address will be sent to Loxin Server. Upon receiving the message, Loxin Server looks up its database with the customer's email address and retrieves the push message service token. Loxin Server then pushes the MintChip Request message to the customer's smartphone through the PMS, as shown in Figure 7.

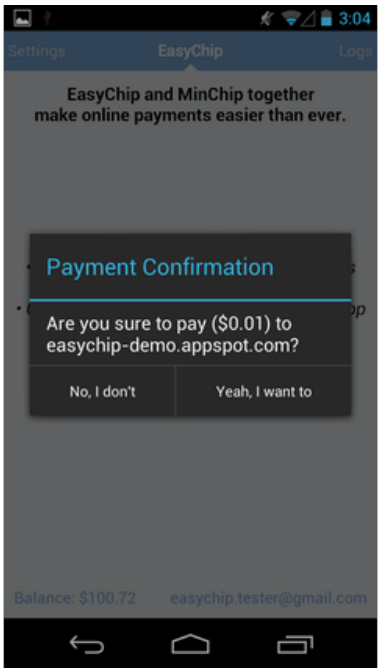

Fig. 7. The customer confirms the payment.

When the customer confirms the payment request, the MintChip inside the customer's smartphone will immediately generate a signed MintChip Value message using the RSA signature scheme [16] and send it back to the merchant's web server. After verifying the received MintChip certificate and digital signature, the payment has been made and the transaction is successful (see Figure 8). Note that the entire authentication and payment processes follow the Loxin security framework and the customer does not need to input any password.

6 Related Work

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

[1] We did not generate and display the hash value on the merchant's website for simplicity.

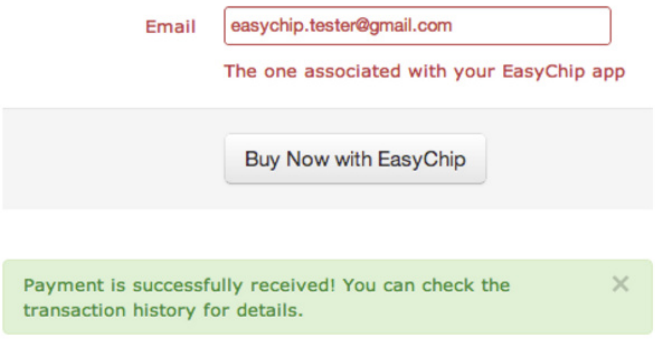

Fig. 8. The online transaction succeeds.

6.1 RSA SecurID

RSA SecurID is a well-established product in two-factor authentication market, which is a hardware token with a small screen showing a pseudo-random authentication code in every minute [1]. Each RSA SecurID shares a secret seed with its backend server. When a user submits the authentication code to a web 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 pseudo-random numbers after obtaining the secret seeds. This kind of incidents did happen in 2011 [11], which renders RSA SecreID less effective to serve as a two-factor authentication mechanism. Moreover, RSA SecurID also has a usability issue and users have to carry the extra hardware device. In addition, different web services usually do not share an identical secret seed, so user may be required to have multiple devices associated with various service providers.

6.2 Google Authenticator

Google Authenticator [2] 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 server, and is required to be manually paired with each service provider similarly as RSA SecurID, which is not user-friendly when compared to Loxin.

6.3 Pico

Pico is a hardware solution proposed by Stajano in 2011 [6], 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.

6.4 Duo Push

Duo Push is a commercial software application developed by Duo Security [12], which aims to provide a two-factor authentication with push message capabilities. However, the detailed design of Duo Push is 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 web services when the verification servers of Duo Security are not working properly or being compromised.

7 Conclusions

In this paper, we propose a universal authentication system called Loxin. We demonstrate that Loxin can be used to replace traditional password-based authentications and 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 a unique security solution for password-free web authentication. Several methods have been proposed to extend Loxin for different use cases, avoid single point of failure, and reduce the web service cost. We also develop EasyChip, a practical application of the Loxin security framework, to demonstrate the power of Loxin for building a real-world password-free mobile payment solution.

References

1. RSA SecurID Hardware Authenticators, RSA Inc., The Security Division of EMC, available at http://www.emc.com/security/rsa-securid/rsa-securid-hardware-authenticators.htm.
2. Google Authenticator Project – Two-Step Verification, Google Inc., available at http://code.google.com/p/google-authenticator/.
3. OpenID Authentication 2.0 - Final, OpenID Foundation, available at http://openid.net/specs/openid-authentication-2_0.html.
4. D. Hardt, "The OAuth 2.0 Authorization Framework", RFC 6749, Internet Engineering Task Force (IETF), Oct 2012.
5. 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.
6. 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.
7. A. Russell, "Comet: Low latency data for browsers", *The Dojo Toolkit*, 2006.
8. Channel Java API Overview, Google Inc., available at https://developers.google.com/appengine/docs/java/channel/.
9. chrome.pushMessaging, Google Inc., available at https://developer.chrome.com/extensions/pushMessaging.html.
10. SimplePush, Mozilla Foundation, available at https://wiki.mozilla.org/Services/Notifications/Push/API.
11. A. Coviello, "Open Letter to RSA Customers", available at https://www.sec.gov/Archives/edgar/data/790070/000119312511070159/dex991.htm.
12. Duo Push: One-Tap Authentication, Duo Security, Inc., available at https://www.duosecurity.com/duo-push.
13. Google Cloud Messaging for Android, available at https://developer.android.com/google/gcm/index.html.
14. Local and Push Notification Programming Guide, Apple Inc., https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/RemoteNotificationsPG.pdf.
15. The MintChip Challenge, The Royal Canadian Mint, available at http://mintchipchallenge.com/.
16. MintChip Developer Resources, The Royal Canadian Mint, available at http://developer.mintchipchallenge.com/.
17. Google Wallet, available at http://www.google.ca/wallet/.
18. Isis Wallet, available at https://www.paywithisis.com/.
19. Paypal Digital Wallet, available at https://www.paypal-promo.com/anywhere/desktop/index.html.
20. EasyChip, available at http://mintchipchallenge.com/submissions/9469-easychip.
21. The Bump API: Why are you waiting for NFC?, Bump Technologies, Inc., available at http://bu.mp/company/api.