Mobile App-Browser Extension Based Authentication

Ritwik Desai[1], **Gaurav Varshney**[2], **Manoj Misra**[1] **and Sajal Jindal**[1]
[1]Department of CSE, IIT Roorkee, Roorkee, India
[2]Area of CSE, NIIT University, Neemrana, India

Abstract— *Phishing attacks are one of the most serious cyber threats faced by users on the Internet. By masquerading as an authentic website, the attackers try to steal users' private and sensitive information such as login details, credit card details, etc. Phishing attacks have led to heavy identity and financial thefts. These attacks involve the use of social engineering techniques to deceive the users. Many schemes have been proposed to detect phishing attacks but the number of such attacks have not declined since its inception. In fact, new attacks like active man in the middle (MITM) phishing attacks have emerged. These attacks allow the attackers to perform active phishing in real time. The current popular authentication schemes fail to address these attacks. In this paper, we propose a novel mobile app-browser extension based authentication (MABEBA) technique to solve the problem of active MITM phishing attacks. A browser extension on the browser establish a two way authentication channel between the user and the website. The experimentation and results depicts that it can be used for secure user authentication in security sensitive environments.*

Keywords: Active MITM phishing, real time phishing, mutual authentication, QR code, WebRTC.

1. Introduction

Phishing is an online deception technique with the intention of stealing users' personal information such as login details, credit card numbers, social security numbers, etc. [1]. It is a form of fraud on the internet in which attackers masquerade as authentic websites and lure users into entering their private and sensitive details. The attackers make use of various social engineering techniques like email spoofing to fool users [2], [3]. The users are unaware of the fact that they are being deceived and hence they input their login information on the fraudulent website. More new tactics of phishing are those which works in real time. In an active man in the middle phishing attack (MITM) the attackers directly logs in to the user's accounts in real time after capturing the login information of the users on the phishing websites. It involves relaying of user inputs received on the phishing website to the authentic website in real time [4], [5]. Another specific MITM is control relay MITM attack where an attacker installs a remote desktop module on user's computer and deceives the user into entering the credentials directly on his own computer. User thinks that he is interacting with his computer while he is interacting with attacker's desktop via a stealthy remote logging module. Malicious browser extensions can also capture login information of user's and hence also are coming up as a big threat to secure authentication schemes. According to the study, thirty percent of the phishing attacks against websites using some form of one-time password based authentication are utilizing MITM techniques to bypass the scheme [6], [7]. We will be discussing the solution for active MITM phishing attacks in this paper.

1.1 Active MITM phishing attacks

It has been identified that schemes such as Google's 2-step verification [8] are vulnerable to active MITM phishing attack. An attack on such one-time password (OTP) based scheme can be carried out by building a phishing website that resembles the login page. The task of the phishing website is to relay username, password and the generated OTP to the real website. For this, the phishing server has a browser automation module that automates the input of username, password and OTP onto the real website. Selenium [9] library can be used to write the browser automation module. Selenium is widely used for automated browser testing and it provides an API to programmatically fill the input fields of the webpage and simulate a click on the submit button to mimic the real user. We have analyzed MITM phishing attacks on other schemes too including QR code based login schemes and push notification based login schemes. Our analysis suggests that a real time phishing attack can compromise such schemes.

1.2 Motivation and Contribution

Currently there are a lot of one way user authentication schemes [10] that only provide a mechanism for client authentication. There is no proper mechanism for server authentication. Therefore, web forgery [11] is possible in such schemes as the user does not get a chance to verify whether the website he is communicating with is authentic or not. Popular schemes such as two-factor authentication schemes also come under one-way authentication because website never authenticates himself in front of the user and the attackers can relay one-time secret such as OTPs, soft tokens in real time. Also one of the main reason behind the failure of most of the user authentication schemes is that the input of credentials is done manually on the websites and such information can be captured by the attackers on a phishing website in real time.

In this paper we propose MABEBA, whose novelty is in the fact that the authentication tokens generated and used for mutual authentication are not accessible to the users and are used in an automated and authentic fashion to complete the lo-

gin process, thus removing the vulnerability of active relaying faced by OTP and other one-time secret based schemes.

1.3 Paper organization

The paper is organized as follows: Section 1 provides an introduction to MITM phishing attacks. Section 2 discusses existing secure authentication schemes and their analysis. Section 3 outlines the proposed scheme and briefs out implementation and results. Section 4 concludes the paper.

2. Related Work

As existing authentication schemes are not sufficient to avoid phishing attacks researchers are working on sophisticated authentication schemes which can tackle this problem. Two factor authentication schemes that require clients to enter an additional second factor have made the life of the phishers difficult than what it was before. The second factor can be biometric information, a one time secret or PIN which can be an OTP such as Google's 2-step verification [8] or can be a QR code [12], [13]or a hardware device which generates a secret or needs to be plugged in for sending a one time secret to the authentication servers. However we have reviewed in one of our paper that all these latest schemes have drawbacks either in terms of security or usability or deploy ability. There are some phishing prevention schemes that do not fall under a particular category. These include shared secret based schemes that make use of a pre shared secret between the client and server such as a four character secret [2], a watermark image [14], shares computed using visual cryptography, [15] etc.Picture password schemes that use text or image based CAPTCHA [16], [17] . Other such schemes include password manager based schemes that store credentials protected by a master password to manipulate credentials before sending to the server [18]–[20].

The research to develop secure authentication scheme is expanding with innovative ideas coming to the table every single day. One of the new addition to anti phishing is additional parameter of proximity added to the authentication process. For example the schemes such as [5] perform authentication based on the proximity of the trusted mobile device and the client machine. This proximity is evaluated based on GPS coordinates of the mobile device and IP address of the client machine. The transfer of information between mobile and the client machine is done through QR code. The limitation is that the IP address can be spoofed by a malicious insider to mimic the location of the user. The use of QR codes in authentication process has increased as it provides an easy one click path to initiate the authentication. In [13], authors propose a single sign-on (SSO) scheme which involves authentication based on an encrypted QR code. It claims to prevent MITM attacks based on time stamps of the QR codes but doesn't provide any practical analysis. In [21], authors propose a novel way to communicate between trusted mobile device and the client machine using QR codes. It prevents MITM attacks using Diffie-Hellman key exchange. The limitations are that it requires two QR scans and a web-cam on the client machine. Additionally, it uses public cryptography, requiring management of multiple keys and is vulnerable to session hijacking. Then there are client certificate based schemes such as [4] in which the trusted mobile device stores the server and client certificates and provides them to the browser during the SSL handshake. The problem with this scheme is that it requires heavy browser side code changes to change the default working of the SSL handshake and also requires management of certificates, their keys and their revocation. Authors in [22] make use of one-time origin bound certificates (OBC) [23] to provide mutual authentication but it makes the scheme platform dependent and requires client side code changes. The scheme described in [17] proposes a new flash based OTP captcha that the user clicks during login. It is difficult for the attackers to relay the OTP since the captcha characters move in a server known trajectory. This requires major server side changes in order to generate and maintain the captcha trajectory and also use of flash plug-in may lead to other security threats and browser crash. In [24] authors propose a visual hash based scheme in which the user is required to enter the password in a browser extension and compare the visual hash generated with the website. The scheme requires heavy user intervention and fails on password leak. Hardware token based schemes require users to carry additional hardware other than their mobile devices. The scheme in [25] requires the user to carry RFID tokens and scanner while the scheme described in [26] requires a USB token. These schemes are complex to deploy and have a complex recovery procedures. The scheme in [18] stores the passwords in the browser extension protected by a master password. The password is only entered on the authentic website, but it faces challenges of secure storage of the passwords and the possibility of the master password leak. Schemes in [19] and [20] modify the password entered on the web page based on the SSL certificate and the domain name, respectively, using a one-way hash function deployed as a browser extension. Thus, a fake website will always receive the wrong password. But the challenges with such schemes are that the user never knows the actual password sent to the real website, therefore the user cannot use the website or service on mobile devices as they don't have the extension facility. An extensive study of ways in which the current authentication schemes can be compromised is given in our videos [1][2] and in one of our latest paper [27].

Based on analysis of existing schemes we have identified that in a secure authentication scheme:

1) The degree of automation of the scheme should be high; for example, the time spent by the user to manually interact with the scheme should be low. More manual interaction and inputs causes the system more prone to

[1] drive.google.com/file/d/113KFOF-zVH5BVp4wjpkb1qjXcGksySTL/view

[2] www.youtube.com/watch?v=6FThk1Iystw&feature=youtu.be

phishing as a user can be always deceived by same look and feel of anything which he is made to interface with.
2) The tokens must be entered by machines which have capability of recognizing original and fraud input interfaces based on one or other measures.
3) Any additional hardware device other than a trusted mobile device should not be needed to implement the scheme or to realize it for authentication in practice.
4) The scheme must not interface with third parties for verification to decrease the risk of credential leakage.
5) It should be easy for the current websites to migrate to the new scheme without any major server side changes. The scheme should work with the existing APIs without any modifications to the browser code. Recovery of the account should be simple if the trusted device is lost.
6) The scheme should not use any metrics such as the IP address for authentication that can be spoofed. Also the use of IP address reveal user location which many users do not want to share with websites.

3. Proposed Scheme

3.1 Introduction to MABEBA

To meet the needs of a secure user authentication scheme we have proposed a mobile app-browser extension based authentication (MABEBA) scheme that involves the use of a trusted mobile application and browser extension on the client's machine (desktop/laptop) to provide secure user authentication on websites. During every login attempt, users request a login token from the server. This request is facilitated by the browser extension. The browser extension contacts the authentic server and requests the token on behalf of the user. The token is maintained by the browser extension and is not available to the other browser extensions or the user. The users input their usernames and passwords and the server authentication part is done by the browser extension by providing valid user tokens to the authentic website without requiring any user intervention. The browser extension ensures server authenticity. When the server receives a token, it is validated and the corresponding user is authenticated. The use of the proposed scheme can be divided into three parts. These are: Registration, Login and Recovery.

3.1.1 Registration

The user registration procedure involves web registration and mobile registration. Three entities take part in the overall registration procedure, namely - mobile app, the browser on the client's machine and the web server. The browser of the client machine is used for web registration followed by the mobile registration on the mobile device using the mobile app with the details obtained from web registration. `Web registration`

1) The user provides relevant information like username (U_{ID}), password (pwd), email address (email) etc to the website opened in the browser.
2) The web server validates the information provided by the user and stores it into the server database. The password is stored as a hashed password (hpwd) that is generated from the user provided password and a random salt ($salt_1$) using [28].
3) Apart from this, the web server also generates a secret S that is shared with the user during mobile registration. The secret is generated using password based key derivation function 2 (PBKDF2) [29]. It requires pwd, a random salt ($salt_2$), iterations(iters) and key length(keylen). S is also stored with the rest of the information.
4) Finally, the web server creates mobile registration metadata (MRM) required for mobile registration. The MRM consists of website name (W_{ID}), token url (T_{URL}), U_{ID} and S. The server encrypts the MRM with a key KR generated in a similar fashion as S but with a different salt ($salt_3$) to get the Encrypted Mobile Registration Metadata (EMRM). The server then renders a QR code as a response holding the API endpoint (R_{URL}) that is mapped to EMRM and the ($salt_3$). This salt is one-half of the information required to decrypt EMRM and the pwd is the other.

The registration message exchanges are described in Fig 1. `Mobile Registration`

1) Mobile registration starts when the QR code is scanned by the mobile app. QR code provides the (R_{URL}) and the ($salt_3$). The app downloads EMRM from (R_{URL}) and decrypts it using K_R computed using PBKDF2 with $salt_3$ and the pwd provided by the user.
2) The MRM details are stored in the mobile database.

We use 100,000 iterations for 256-bit AES key generation using PBKDF2. This is suitable for a device like Motorola E. The CPU usage and time taken to generate the key depend on the number of iterations. We have built the mobile app over the Android platform (iOS can also be used). Using the Android keystore API [30], the metadata details are securely stored into the app database. The keystore API provides app specific keys that are accessible only within the app's context to encrypt the data. Further protection can be provided by using PIN locks.

Figure 1 shows the exchange of messages between the app, browser and the server. Here, the browser and the server communicate over HTTPS and R_{URL} is a HTTPS URL.

3.1.2 Login

The login procedure is achieved using the mobile app, browser and browser extension on client's machine and the web server. The login procedure is initiated using the mobile app. The login procedure is as follows:

1) User opens the login page on the browser. The login page has the usual username and password fields. But apart from those details, a login token is also required for a successful login.
2) The token is requested by the browser extension with the help of the mobile app on behalf of the user. First the mobile app and the browser establish a secure peer-to-peer (P2P) connection using Web real-time communication (WebRTC) [31]. This connection is encrypted using DTLS protocol. The connection is established when the mobile app scans the QR code rendered by the browser extension. The semantics of this connection is discussed in section 3.2.
3) After the successful P2P connection establishment, the app requests a token to the server through the browser extension by creating a token request (T_R). This request consists of the U_{ID}, T_{URL}, timestamp (T) and encrypted request data (ERD). The request data (RD) consists of U_{ID} and T. The ERD is computed by the encryption of RD using a one-time secret

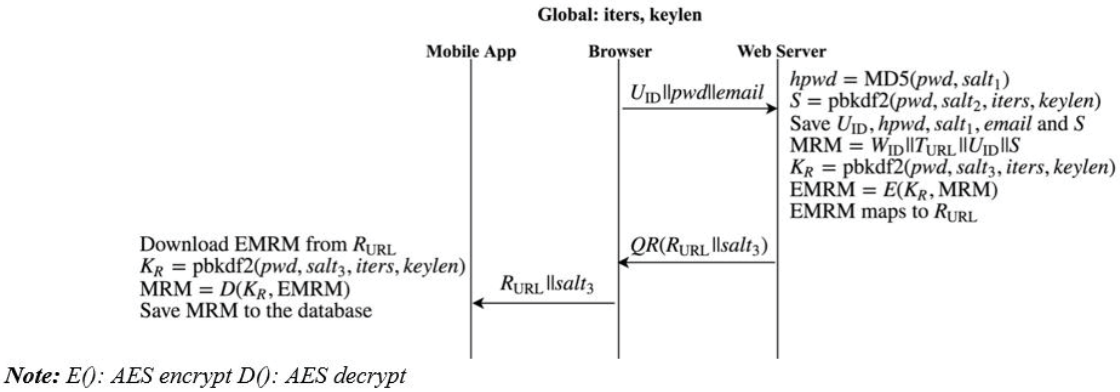

Fig. 1
REGISTRATION MESSAGE EXCHANGE

key S' obtained from S, U_{ID} and T by using keyed-hash based authentication code (HMAC) [32].

4) When the browser extension receives TR, it makes a secure (HTTPS) POST request to the web server on behalf of the user at T_{URL}. T_{URL} is a REST [33] API endpoint. The POST body contains U_{ID}, T and ERD.
5) When the web server receives the request from the browser extension, it verifies the request by decrypting ERD using S' and checking that T is in the acceptable limit of the current times. The limit is kept at 15 seconds but can be modified as per developer requirements.
6) After verifying the request, the web server generates a token (TKN). TKN contains U_{ID}, expiry time (E_T), encrypted token data (ETD) and a signature (sign). Token data (TD) is composed of browser ID (B_{ID}) that identifies the subsequent requests, and a random nonce. ETD is obtained by encrypting TD by K. K is generated from server key SK, U_{ID} and E_T using HMAC [32]. The signature helps in checking the integrity of the token and is calculated using K, U_{ID}, E_T and TD.
7) After TKN generation, the web server saves token details and sends TKN to the browser extension.
8) When the browser extension receives the token, the user is notified. Now, the user inputs U_{ID} and pwd and submits the login form.
9) In the background, the webpage requests the browser extension for TKN as shown in Figure 2. The browser extension provides TKN if and only if the domain of the original TR is same that of the current web page. This prevents server spoofing.
10) When the web server receives U_{ID}, pwd and TKN, it grants the user access after successful verification. The expiry of the token is kept at sixty seconds, which is enough for the user to enter login details.

The login message exchanges are given in Figure 2. Thus, MABEBA provides mutual authentication. The client is authenticated using the token request and the login credentials and the server is authenticated by the browser extension by providing the token to the legitimate website hence eventually providing a two way authentication channel. The server authentication is achieved without any user intervention. Also by entering username and password on the authentic browser extension instead of websites the threats from malicious browser extensions that sniffs user information reduces to almost zero. Figure 2 shows the message exchange during login. Here, the connection between mobile app and browser extension is encrypted using DTLS and the connection between browser extension and server is done over HTTPS.

3.1.3 Account recovery

If the trusted mobile device is lost, the account can be recovered through the registered email address. The users can request for the reset of the shared secret with the server and an email will be sent to their registered email address. By visiting the link in the email and providing the password, a new QR code is generated, similar to the registration QR code. The users then can register this QR code with the new app. A password reset is also done with the help of a registered email address.

3.2 WebRTC

WebRTC [31] is a set of APIs for Real-Time Communication between browsers and mobile applications. Currently it is supported by major browsers like Chrome and Firefox and mobile platforms like Android and iOS. WebRTC provides data channels for end to end encrypted peer to peer (P2P) communication between two devices using DTLS encryption. Key components of WebRTC are described as follows:

3.2.1 Session Description

Real time communication (RTC) session description includes the session metadata for establishing the P2P connection using WebRTC. It includes DTLS parameters, transport information, interactive connectivity establishment information, i.e. STUN/TURN information.

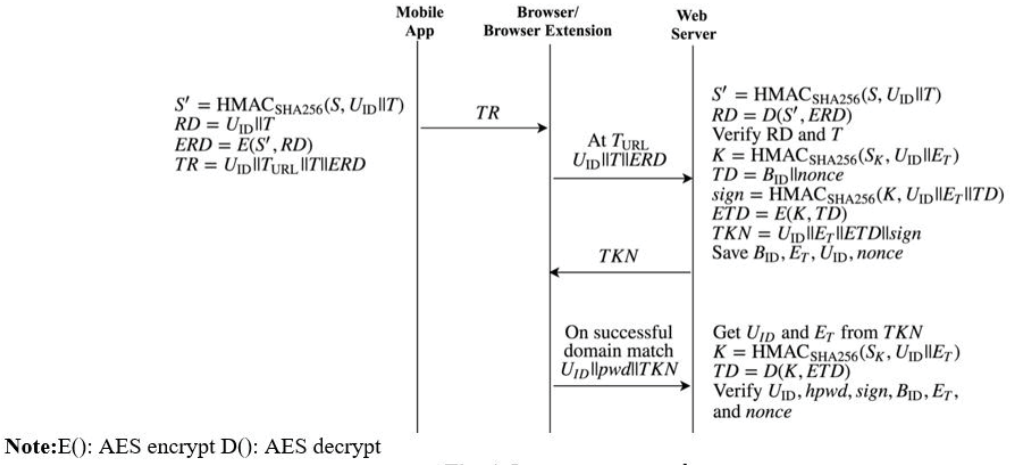

Fig. 2
LOGIN MESSAGE EXCHANGE

3.2.2 STUN/TURN server

If the communicating parties are behind NAT, it is not possible for them to use their private addresses to communicate. Using the STUN [34] protocol, the devices can obtain their public addresses. But if the devices are behind the symmetric NAT, the STUN protocol fails and the TURN [35] protocol is used to relay the data. The STUN/TURN server facilitates the STUN or TURN requests. But if all the peers are on the local network, this server is not required. According to the study in [36], STUN works 86% of the time.

3.2.3 Signaling server

The main objective of the signalling server is to exchange the session descriptions of the peers so that they can identify each other. The peers have to trust the signalling server for providing them with the correct session descriptions.

3.3 P2P connection

We have used a wrapper API over WebRTC called PeerJS [37]. It provides a Web socket signalling server implementation and the WebRTC data channel JavaScript API. It's easy to deploy it on the browser and in the web view of an Android app. The signalling server opens a web socket [38] connection with every peer. In the case of MABEBA, there is a connection with the browser on a client's machine and the web view of the Android device. The signalling server assigns a unique peer ID to each device. Using this peer ID the exchange of the session descriptions is facilitated. When the browser gets its peer ID, it is rendered as a QR code by the browser extension. The mobile app then scans the peer ID and asks for the session description of the browser extension to the signaling server using it and also provides its own session description. The signalling server exchanges the session description based on the peer ID. Now the two peers establish a P2P connection using WebRTC. The STUN [34] or TURN [35] protocol is used if required. After establishing a P2P connection using WebRTC, the peers agree on a shared secret using Diffie-Hellman key exchange. This secret is truncated to a six-digit code that the users can easily verify. This is done to ensure the app has established a connection with the browser extension without any MITM attacks. The established P2P connection is encrypted by DTLS. This connection is end to end encrypted. WebRTC provides reliable data channel over SCTP.

Table 1
TIME TAKEN

Task	Time (Sec)
Registration Time (local Wi-Fi network)	23.525
Registration Time (mobile on 3G)	25.161
Login Time (local Wi-Fi network)	8.897
Login Time (mobile on 3G)	12.348

Table 2
RESOURCE USAGE

Device	CPU usage	RAM usage
App (registration)	1 to 56 %	17MB
App (login)	1 to 47 %	11MB
Browser extension	1.4 to 14.1 %	64MB(idle) - 94MB(running)

3.4 Results and Findings

3.4.1 Time and resource usage

The time taken for registration and login is calculated as an average of over twenty different trials. The resource usage is measured using the task manager for the browser extension and using Android Studio for the mobile app. The results are shown in Table 1 and Table 2.

3.4.2 Theoretical attack analysis

The theoretical attack analysis involves evaluating the scheme against brute force attacks and cryptanalysis. The attackers exhaustively try all the key combinations in order

to find the correct secret keys for the token request or the token generated by the server. MABEBA uses 256-bit AES algorithm for encryption and decryption.

- `Attacks on token generation`: Every token request is encrypted using a one-time AES key and is further communicated over the encrypted DTLS channel. Thus it is practically impossible for the attacker to break the 256 bit AES encryption to generate the token request.
- `Attacks on token`: The token is encrypted using one-time 256 bit AES key and it is maintained by the browser extension. The extension only provides the token to the authentic website based on domain match. Hence, the attacks on the token are practically impossible.
- `Key Generation`: The AES keys are generated using PBKDF2, which provides key stretching. This makes secret key calculation difficult even for weak passwords.

3.5 Comparison using Bonneau et al. [39]

The comparison with other techniques has been performed on the following broad parameters: Usability, Deployability and Security. We have taken the parameters of Bonneau et al. assessment framework [39] for a comparison.

4. Conclusions and future work

4.1 Conclusions

Phishing attacks are one of the major problems on the Internet incurring huge losses for banking and financial companies. The most popular way of authentication, Google's 2-Step verification, is vulnerable to active MITM phishing attacks. Mobile-Extension based authentication (MABEBA) scheme proposed in this paper helps in mitigating threats of phishing attacks by providing mutual authentication. MABEBA offers the following benefits:

1) Though it requires additional WebRTC infrastructure, it removes the requirement of carrying hardware tokens, installing web cams and Bluetooth on the PCs to use the existing advanced authentication schemes.
2) It does not require the users to manage certificates and public-private key pairs.
3) MABEBA is compatible with the popular browsers and mobile platforms. Therefore, it is platform independent.
4) It does not require any additional software to be installed on the system such as Flash.
5) It is possible for the websites to use MABEBA by only doing some minor server side changes to store additional secret and token details.
6) The token is hidden to the users and they cannot input it on the phishing website. Therefore, MABEBA prevents passive and active MITM phishing attacks.
7) The degree of automation is high compared to OTP based schemes which rely on user input for the OTP.

4.2 Future Work

1) Currently, MABEBA uses web view in the app for WebRTC communication. Web view is required to use the JavaScript API of WebRTC. The overhead of the web view is noticeable and adds to the P2P connection establishment time. We plan to implement the mobile app using native libraries to improve the login time of the scheme in the future.
2) We also plan to perform user testing of the MABEBA scheme to determine usability parameters like ease of use, learning curve, etc.

References

[1] PhishTank, "What is phishing? @ONLINE," Apr 2015. [Online]. Available: https://en.wikipedia.org/wiki/Phishing/

[2] G. Varshney, A. Sardana, and R. C. Joshi, "Secret information display based authentication technique towards preventing phishing attacks," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, ser. ICACCI '12. New York, NY, USA: ACM, 2012, pp. 602–608. [Online]. Available: http://doi.acm.org/10.1145/2345396.2345494

[3] G. Varshney, R. C. Joshi, and A. Sardana, "Personal secret information based authentication towards preventing phishing attacks," in *Advances in Computing and Information Technology*. Springer, 2012, pp. 31–42.

[4] B. Parno, C. Kuo, and A. Perrig, "Phoolproof phishing prevention," in *International Conference on Financial Cryptography and Data Security*. Springer, 2006, pp. 1–19.

[5] S.-H. Kim, D. Choi, S.-H. Jin, and S.-H. Lee, "Geo-location based qr-code authentication scheme to defeat active real-time phishing attack," in *Proceedings of the 2013 ACM workshop on Digital identity management*. ACM, 2013, pp. 51–62.

[6] H. N. Security, "Real time phishing attacks increase@ONLINE," Oct 2015. [Online]. Available: https://www.helpnetsecurity.com/2010/11/10/real-time-phishing-attacks-increase/

[7] S. Intelligence, "Real time phishing takes off @ONLINE," Oct 2015. [Online]. Available: https://securityintelligence.com/real-time-phishing-takes-off/

[8] Google, "Stronger security for your google account @ONLINE," Oct 2015. [Online]. Available: https://www.google.com/landing/2step

[9] Selenium, "Selenium browser automation @ONLINE," Oct 2015. [Online]. Available: https://www.seleniumhq.org/

[10] C.-M. Leung, "Visual security is feeble for anti-phishing," in *2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication*. IEEE, 2009, pp. 118–123.

[11] Wikipedia, "Phishing - website forgery @ONLINE," Oct 2015. [Online]. Available: https://en.wikipedia.org/wiki/Phishing#Websiteforgery

[12] B. Dodson, D. Sengupta, D. Boneh, and M. S. Lam, "Secure, consumer-friendly web authentication and payments with a phone," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2010, pp. 17–38.

[13] S. Mukhopadhyay and D. Argles, "An anti-phishing mechanism for single sign-on based on qr-code," in *Information Society (i-Society), 2011 International Conference on*. IEEE, 2011, pp. 505–508.

[14] A. P. Singh, V. Kumar, S. S. Sengar, and M. Wairiya, "Detection and prevention of phishing attack using dynamic watermarking," in *Information Technology and Mobile Communication*. Springer, 2011, pp. 132–137.

[15] D. James and M. Philip, "A novel anti phishing framework based on visual cryptography," in *Power, Signals, Controls and Computation (EPSCICON), 2012 International Conference on*. IEEE, 2012, pp. 1–5.

[16] B. B. Zhu, J. Yan, G. Bao, M. Yang, and N. Xu, "Captcha as graphical passwordsa new security primitive based on hard ai problems," *IEEE transactions on information forensics and security*, vol. 9, no. 6, pp. 891–904, 2014.

[17] C.-M. Leung, "Depress phishing by captcha with otp," in *2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication*. IEEE, 2009, pp. 187–192.

[18] K.-P. Yee and K. Sitaker, "Passpet: convenient password management and phishing protection," in *Proceedings of the second symposium on Usable privacy and security*. ACM, 2006, pp. 32–43.

[19] Y. Joshi, D. Das, and S. Saha, "Mitigating man in the middle attack over secure sockets layer," in *Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on*. IEEE, 2009, pp. 1–5.

Table 3
COMPARISON WITH EXISTING SCHEMES USING BONNEAU ET AL.'S FRAMEWORK

		[8]	[40]	[21]	[5]	[13]	[12]	[17]	[16]	[41]	[42]	[43]	[44]	[45]	UP	PW
1.	Mem. wise effortless	✗	✗	✗	❍	✗	✓	✗	✗	✗	✗	✗	❍	❍	✗	❍
2.	Scalability for users	✗	✗	✗	✗	✗	✓	❍	❍	❍	❍	❍	✓	✓	✓	✓
3.	Nothing to carry	❍	❍	❍	❍	❍	❍	✓	✓	✗	✗	✗	❍	✓	✓	❍
4.	Physically effortless	❍	❍	✗	✗	✗	✗	✗	✗	✓	❍	❍	❍	✓	✓	✓
5.	Easy to learn	✓	❍	❍	✗	❍	❍	✗	✗	✓	✓	✓	✓	✓	✓	✓
6.	Efficient to use	❍	✓	❍	❍	❍	❍	✗	✗	✓	✓	✓	✓	✓	✓	✓
7.	Infrequent errors	❍	✓	✗	✗	❍	❍	✗	❍	✓	✓	✓	✓	✓	✓	✓
8.	Easy recovery from loss	❍	❍	❍	❍	❍	❍	✓	✗	❍	❍	❍	❍	❍	✓	❍
9.	Accessible	❍	❍	✗	✗	✗	✗	✗	✗	✓	❍	✓	❍	✓	✓	❍
10.	Negligible cost/user	✗	❍	❍	❍	❍	❍	✗	✓	✓	✗	✗	❍	✓	✓	❍
11.	Server compatible	❍	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	❍
12.	Browser compatible	✓	✓	❍	❍	❍	❍	❍	❍	✓	✓	❍	✓	✓	✓	✗
13.	Mature	✓	✓	✗	✗	❍	❍	✗	✗	❍	✓	✓	✓	✓	✓	❍
14.	Non-Proprietary	✓	❍	✓	✓	✓	✓	✓	✓	✗	✗	✗	❍	✓	✓	✓
15.	Res. to physical observation	❍	❍	✓	✓	✓	✓	❍	❍	❍	❍	✓	❍	❍	✗	✓
16.	Res. to target impersonation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
17.	Res. to throttled guessing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
18.	Res. to unthrottled guessing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
19.	Res. to internal observation	✗	✗	✗	❍	✗	✗	❍	❍	✗	✗	❍	❍	✗	✗	✓
20.	Res. to leak from other verifiers	✓	❍	✓	✓	❍	✓	✓	✓	❍	❍	❍	✓	❍	✓	✓
21.	Res. to Phishing	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
22.	Res. to Theft	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓
23.	No Trusted Third Party	✓	✗	✓	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✓	✓
24.	Requiring explicit consent	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
25.	Unlinkable	✓	✓	✓	❍	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Benefit Offered Count	12	10	10	9	8	11	11	12	14	11	13	14	14	18	17

Note: ✓: Offer the benefit, ✗: Benefit is not offered, ❍: Partially offers the benefit, Benefit offered count: Number of benefits completely offered by the given scheme, UP: Username Password based traditional scheme, PW: Proposed Work

[20] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, "Stronger password authentication using browser extensions." in *Usenix security*. Baltimore, MD, USA, 2005, pp. 17–32.

[21] M. Xie, Y. Li, K. Yoshigoe, R. Seker, and J. Bian, "Camauth: Securing web authentication with camera," in *2015 IEEE 16th International Symposium on High Assurance Systems Engineering*. IEEE, 2015, pp. 232–239.

[22] 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*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 404–414. [Online]. Available: http://doi.acm.org/10.1145/2382196.2382240

[23] M. Dietz, A. Czeskis, D. Balfanz, and D. S. Wallach, "Origin-bound certificates: a fresh approach to strong client authentication for the web," in *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 317–331.

[24] R. Dhamija and J. D. Tygar, "The battle against phishing: Dynamic security skins," in *Proceedings of the 2005 symposium on Usable privacy and security*. ACM, 2005, pp. 77–88.

[25] J.-C. Liou, G. Egan, J. K. Patel, and S. Bhashyam, "A sophisticated rfid application on multi-factor authentication," in *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*. IEEE, 2011, pp. 180–185.

[26] Google, "Using security key for 2-step verification @ONLINE," Oct 2015. [Online]. Available: https://support.google.com/accounts/answer/6103523

[27] G. Varshney, M. Misra, and P. Atrey, "Secure authentication scheme to thwart rt mitm, cr mitm and malicious browser extension based phishing attacks," *Journal of Information Security and Applications*, vol. 42, pp. 1 – 17, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212618300140

[28] R. L. Rivest *et al.*, "Rfc 1321: The md5 message-digest algorithm," *Internet activities board*, vol. 143, 1992.

[29] B. Kaliski, "Rfc 2898; pkcs# 5: Password-based cryptography specification version 2.0," 2000.

[30] A. Developers, "Android keystore system @ONLINE," Mar 2016. [Online]. Available: http://developer.android.com/training/articles/keystore.html

[31] WebRTC, "Web real-time communication@ONLINE," Mar 2016. [Online]. Available: http://webrtc.org

[32] H. Krawczyk, R. Canetti, and M. Bellare, "Hmac: Keyed-hashing for message authentication," 1997.

[33] Wikipedia, "Representational state transfer @ONLINE," Feb 2016. [Online]. Available: https://en.wikipedia.org/wiki/Representationalstatetransfer

[34] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session traversal utilities for nat (stun)," Tech. Rep., 2008.

[35] R. Mahy, P. Matthews, and J. Rosenberg, "Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun)," Tech. Rep., 2010.

[36] W. Stats, "Webrtc statistics and metrics @ONLINE," Mar 2016. [Online]. Available: http://webrtcstats.com

[37] PeerJS, "Peerjs @ONLINE," Feb 2016. [Online]. Available: http://peerjs.com

[38] I. Fette and A. Melnikov, "Rfc 6455: The websocket protocol," *IETF, December*, 2011.

[39] 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," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012, pp. 553–567.

[40] SAASPASS, "Multifactor authentication @ONLINE," Jan 2019. [Online]. Available: https://saaspass.com/

[41] Tricipher, "White paper preventing man in the middle phishing attacks with multi-factor authentication @ONLINE," Jan 2019. [Online]. Available: https://www.globaltrust.it/documents/press/phishing/PhishingSolutionWhitepaper.pdf

[42] V. Beal, "Rsa secure id @ONLINE," Jan 2019. [Online]. Available: https://www.webopedia.com/TERM/R/rsa_secure_id.html/

[43] Yubico, "Your key to a safer internet @ONLINE," Jan 2019. [Online]. Available: https://www.yubico.com/

[44] G. Varshney and M. Misra, "Push notification based login using ble devices," in *2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Nov 2017, pp. 479–484.

[45] Lastpass, "Lastpass remembers all your passwords, so you don't have to @ONLINE," Jan 2019. [Online]. Available: https://www.lastpass.com/