Enhancing Anti-phishing by a Robust Multi-Level Authentication Technique (EARMAT)

Adwan Yasin and Abdelmunem Abuhasan
College of Engineering and Information Technology, Arab American University, Palestine

Abstract: *Phishing is a kind of social engineering attack in which experienced persons or entities fool novice users to share their sensitive information such as usernames, passwords, credit card numbers, etc. through spoofed emails, spams, and Trojan hosts. The proposed scheme based on designing a secure two factor authentication web application that prevents phishing attacks instead of relying on the phishing detection methods and user experience. The proposed method guarantees that authenticating users to services, such as online banking or e-commerce websites, is done in a very secure manner. The proposed system involves using a mobile phone as a software token that plays the role of a second factor in the user authentication process, the web application generates a session based onetime password and delivers it securely to the mobile application after notifying him through Google Cloud Messaging (GCM) service, then the user mobile software will complete the authentication process – after user confirmation- by encrypting the received onetime password with its own private key and sends it back to the server in a secure and transparent to the user mechanism. Once the server decrypts the received onetime password and mutually authenticates the client, it automatically authenticates the user's web session. We implemented a prototype system of our authentication protocol that consists of an Android application, a Java-based web server and a GCM connectivity for both of them. Our evaluation results indicate the viability of the authentication protocol to secure the web applications authentication against various types of threats.*

Keywords: *Phishing, two-factor authentication, web security, google cloud messaging, mobile authentication.*

Received September 29, 2015; accepted June 1, 2016

1. Introduction

The internet evolution attracted most business institutions to provide their transactions online through web-based applications, among them, banks, stocks and e-commerce websites are widely spread nowadays.

Various attacks are arising on web-based systems, exploiting application's security weaknesses, browsers vulnerabilities and user's lack of experience to compromise the user critical information such as user's identity information and credentials. Among those attacks, Phishing attacks are continuously threatening users and websites with identity theft that leads to compromising the user account and being able to perform transactions on behalf of the user.

Phishing attacks rely on social engineering techniques and illegal usage of technology to obtain user's sensitive information. In a phishing attack, users of a particular service can be asked to sign-in to a clone of the original authentication system connected to a masqueraded domain name, where the attacker can steal their credentials and authenticate himself to the legitimate web site on behalf of the user.

The continuous and increasing phishing attacks on financial, retail and e-commerce websites urges the importance of developing new defence mechanisms against them, the number of unique phishing reports submitted to Anti Phishing Work Group (APWG) during Q4 of 2014 was 197,252. This was an increase of 18 percent from the 163,333 received in Q3 of 2014 [2].

Phishing attacks rely upon a mix of technical deceit and social engineering practices. The Phisher must persuade the victim to intentionally perform a series of actions that will provide access to confidential information. The user is usually fooled by following fake links sent by phishers through a communication channels such as email, web pages and instant messaging services [19]. Usually, the phisher must impersonate a trusted source (e.g., the helpdesk of their bank, automated support response from their favourite online retailer, etc.,) for the victim to believe. Once the user clicks on the fake link, he will be directed to the phishing site-which is a fake copy of the original one-and requested to enter his credentials, which are then captured by the phisher and used later on behalf of the user in the original web site.

Phishing attacks are the most critical type of security attacks of web applications, due to their continuous adaptation to security defence mechanisms, the Anti-Phishing work group [2] yearly report about phishing attacks shows huge number of phishing sites that are deployed monthly, see Figure 1. This gives a clear urge about the necessity of developing anti-phishing schemes that are capable of preventing this type of attacks.

Fighting phishing attacks has become an urgent and continuous action by organizations and the web-security research community, a set of mechanisms and

policies has been developed to prevent or detect the phishing attacks, including protecting the user's personal computer, increasing user awareness, phishing sites blacklisting, detection and prevention of phishing attacks.

The mechanisms of protecting the user's computer against malwares and spoofed emails will never give a full guarantee of solving the problem of phishing attacks; the best phishing detection algorithms still have an error rate not less than 7%. [1]

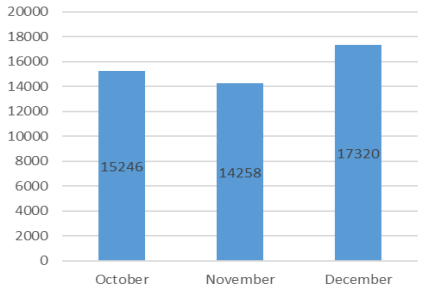


Figure 1. Unique phishing sites detected Oct.-Dec. 2014 [2].

- *Phishing attack vectors*: For a Phishing attack to be successful, the phishers usually use a set of attack methods that either exploit security weaknesses and vulnerabilities at the user's computer, the Internet Service Provider (ISP) network or the website itself. The most common methods include [19].
- *Man-in-the-middle attacks*: In this class of attack, the attackers situate themselves between the customer and the real web-based application, and proxies all communications between the user and the real web site, from this point, the attacker can observe and record all transactions including the user's credentials. For man-in-the-middle attacks to be successful, the attacker must be able to direct the customer to their proxy server instead of the real server. This may be carried out through a number of methods including Transparent Proxies, Domain Name System (DNS) Cache Poisoning, URL Obfuscation and Browser Proxy Configuration.
- *URL obfuscation attacks*: phishers try to obfuscate the final destination of the customer's web request through bad domain names, friendly login URL's, third party shortened URL's or host name obfuscation.
- *Cross-site scripting attacks*: commonly referred to as Cascading Style Sheet (CSS) or XSS and make use of custom URL or code injection into a valid web-based application URL or embedded data field. In general, these CSS techniques are the result of poor web-application development processes.
- *Domain hijacking*: also known as Domain Theft and is defined as the act of changing the registration of a domain name without permissions of the original domain owner. In the context of phishing attacks, the hijacker can replace the website with an identical website that records private information such as log-in passwords.
- *Content spoofing*: in this type of attack, the attacker tries to convince a user that a malicious content appearing in a website is legitimate, an obvious example of content spoofing is by including a fake login frame into a spoofed website that the user trusts, and fooling the user to enter his credentials in the fake login frame which is controlled by the attacker.
- *Pre-set session attacks*: In this class of attacks, the phishing message contains a web link to the real application server, but also contains a predefined Session ID field. The phisher keeps trying to access the real website with the predefined session id; once the fooled user uses the link to access the server, the phisher will be able to access the server also. This could happen in website with very poor security controls.
- *Observing customer data*: key-loggers and screen-grabbers can be used to observe confidential customer data as it is entered into a web-based application and then transmitted to hackers.

In the literature, we can categorize the proposed solutions to the phishing attacks into four categories; ranging from end users training into more complex, user transparent (technical) solutions.

The end user of an application is the key factor in preventing phishing attacks; a trained and experienced user can in most cases decide whether a link sent to him by email or SMS is a legitimate one or not, using a set of factors including URL correctness, the message language and whether the connection is secure or not, in addition to the contents and structure of the target website. Unfortunately, this assumption is not effective, as most users are not aware of the basic security concepts or features [7, 10], and thus relying on user's awareness or experience is not a dependable option for fighting Phishing attacks; especially for advanced and well prepared phishing attacks that are continuously arising.

On the other hand, three types of technical solutions and proposals are found in the literature of combating the phishing problem; Secure Socket Layer (SSL)/ Transport Layer Security (TLS) and third-party certification, anti-phishing tools and extending the traditional user authentication schemes with a second factor.

- *SSL/TLS and third-party certification*: TLS and its predecessor SSL protocols are implemented to secure communication between a client and a server through encrypting the connection data, and authenticating servers through Certificate Authorities (CA) based on Public Key Infrastructure (PKI).

Despite the robustness of the TLS/SSL security models, they are not considered as an ideal solution for the phishing attacks, due to that the root causes of

phishing attacks is based on fooling users to enter their sensitive information on a fake website after drawing an illusion for the user that he is accessing the original website; the only visible feature for websites that implement SSL is the lock icon that appears on the client browser, what if the phishing site itself is using SSL too? The same lock icon will appear too, for most novice users this could be an indication that the website is original while in fact it is not.

- *Anti-phishing tools*: which are toolbars or add-ons/extensions for internet browsers, e.g., Spoof guard [4] and Microsoft phishing filters. Those toolbars are used to detect phishing websites based on a set of factors, including: existence of website certificates, black listing of phishing sites, artificial intelligence approaches including fuzzy logic and Bayesian rules. probability of failure in anti-phishing tools is not less than 7% [1].
- *Extending user authentication with a second factor*: traditional methods for authenticating a user to a web application rely on one of the three "somethings" he knows, he has and he is, as clarified in Table 1. Those one-factor authentication models represent a one point of failure solutions; as they could be compromised "easily" by exploiting key loggers, malwares, eavesdropping, physical control, brute force, etc.,

Table 1. Authentication methods.

Authentication method	Details
Something you know	The traditional authentication method using ID/Password, PIN, Passphrase…
Something you have	Using One-time passwords, smart cards, hardware tokens…
Something you are	Prove the users' identities through the users' biometric information. Fingerprint or iris is used instead of the password. Require the expensive readable device.

Two-factor authentication (known as 2FA) is a technique patented in 1984 [15], the basic idea of 2FA is to identify users by means of two different components; e.g. password and a finger print together, or some hardware token. The popularity of 2FA rely on the assumption that an attacker is unlikely to possess both factors of authentication. Nevertheless, a set of obstacles arise when analysing current 2FA schemes, and we summarize them as follows:

1. The cost of the second factor: many 2FA schemes employ hardware tokens or other costly mechanisms (such as SMS and phone calls), which adds a cost for each login attempt for both the website vendor and/or the user.
2. Physical or logical security of the second factor: hardware token could be stolen; software applications could be vulnerable to threats.
3. Availability of the second factor: implementing 2FA requires the existence of the second factor any time the user wishes to access his account. This requirement could make a logistic problem for the user (especially for hardware tokens); especially if he is enrolled in more than one website that uses 2FA.
4. Usability of the second factor: 2FA schemes implies that the user needs to perform extra steps to access their account, such as plugging in hardware token, installing extra software, entering a second One Time Password (OTP), using cameras, Etc.
5. Professional phishing threats challenge: despite the robustness of current 2FA schemes over traditional one-factor authentication, it is still vulnerable to organized and intelligent threats that usually concentrate on attacking 2FA schemes during the registration Process [8].

In this paper, we review, analyse and evaluate a set of current 2FA schemes, and then propose a new 2FA protocol that is usable, zero-cost and secure. The rest of the paper is organized as follows: in section 2 we review the related work, in section 3 we draw our protocol design goals and assumptions, section 4 explains the protocol architecture, implementation and evaluation.

2. Related Work

James and Philip [14] proposed a Novel Anti Phishing framework based on Visual Cryptography. The framework generates an image captcha based on the user information in the registration phase; the image captcha is divided into two shares such that one of the shares is kept with the user and the other share is kept in the server. Then, the user's share and the original image captcha is sent to the user for later verification during login phase.

This proposed mechanism is secure against phishing attacks if the connection between the client and the server is encrypted by the SSL protocol. Moreover, a usability and accessibility problem could arise for the user as he is required to upload his share of the image each time he wants to login to the web application; a logistics problem would arise as the image share should be available on the computer from which the login process will take place.

Another authentication scheme proposed by Gal´an *et al*. [11], "A Strong Authentication Protocol based on Portable One-Time Dynamic URLs", this scheme relies on generating one-time dynamic and portable URL for each user once he logs in to the web application. This URL is generated specifically for the user in the specified session and then sent to the user through a predefined communication channel (usually SMS or email address). After generating the URL, the server encrypts it using a shared key with the user; when the user receives the encrypted URL he is required to decrypt it and then access the web application through this URL.

Dodson *et al*. [9] proposed Snap2Pass, a mobile based authentication system that aims at replacing the traditional password-based web authentication; leveraging either RSA model or symmetric key encryption. Snap2Pass is based on the challenge-response authentication model; where the server sends a challenge (encrypted token) to the user encapsulated with a Quick Response (QR) code, who in turn needs to scan, decrypt and send it back to the server for identity verification. While this scheme successfully replaces traditional password-based web authentication, two weakness points could harm both the usability and security of this scheme, namely the user's mobile internet connectivity need and the shared key distribution mechanism.

The authors in [17] proposed a user authentication scheme that leverages a user's Android smartphone and SMS to resist password stealing and password reuse attacks, such that the user identity is verified using the mobile application by sending an encrypted one-time secret to the server using SMS such that the server can verify the user's identity.

The authors in [21] proposed the concept of virtual password authentication, where a user-specified function is used to calculate the virtual password with a trade-off of security for a little more complexity for the user in computing the specified function. Users are authenticated using a dynamic password computed each time using the user's specified function.

Cronto [5] is a commercial transaction authentication system to protect online banking transactions against malware on the user's browser, on this scheme, the user needs to confirm his online transaction using his mobile phone; the website encapsulates an encrypted text containing the transaction details and a onetime code and sends it back in a QR code to the client browser, then the user needs to scan this code into his mobile and decrypt the transaction data per-device key it shares with the bank and display the transaction details in the phone screen, the user then confirms the transaction by entering the transaction password in the browser.

Xie *et al.* [20] proposed CamAuth, a 2FA scheme that leverages user's mobile as a second authentication factor, where user identity is proved using a combination of Diffie-Hellman keys exchanged between the client browser (through an extension or Add-on) and the server, and then verified using the user's mobile device via exploiting both the user PC and mobile cameras to exchange data that is encapsulated within a QR code. Three usability and deployability drawbacks could limit the adoption of such an authentication scheme:

1. Users are required to install a browser plugin to be used as part of the authentication process; this requirement will limit the user who wishes to access his account from public computers.
2. CamAuth assumes that the user PC is equipped with a camera to be used as a medium to exchange data with the user's mobile. This assumption is not true for a wide range of users whom PCs are not equipped with cameras, in addition to limiting the opportunities of users wishing to access their accounts from public or work computers.
3. The process of authentication and specially reading QR codes with both the PC's and mobile's cameras could result in usability inefficiencies and inconvenience for users.

Another category of 2FA schemes rely upon client side generation of one time passwords to be used as a second authentication token; a popular 2FA method that falls in this category is Google Authenticator (GA) [13]. GA is a mobile software that generates offline authentication codes that are used as a second authentication token; such that when the user access his account, he is requested to enter the generated code in addition to his credentials. GA generates authentication codes based on pre-shared secrets that were fed to the software in the registration process; they usually include user specific account details, code generation method (counter based or timestamp based), OTP characteristics, Etc., those pre-shared secrets and fed to the GA software through a QR code scanned with the user mobile camera.

Dmitrienko *et al*. [8] performed a security analysis that concluded that such schemes are vulnerable to attacks especially in the registration phase; a PC standing malware can intercept the QR code that encapsulates the pre-shared secrets, then the attacker can initialize his own version of GA and thus being able to generate valid authentication codes for the compromised account.

Czeskis *et al*. [6] proposed PhoneAuth, a 2FA scheme in which the user mobile is considered a second authentication factor in addition to the user credentials; the user is authenticated after signing the login ticket (generated by the server) with the client private key that resides in the user' mobile. The login ticket is communicated back and forth between the client browser and the mobile application through Bluetooth.

PhoneAuth is built upon the origin-bound certificate, which modifies TLS to realize strong client authentication. The deployment of PhoneAuth requires modification to current TLS, web browser, and smartphone firmware, which is not practical for average users. Second, PhoneAuth relies on Bluetooth for communications between the smartphone and PC.

However, Bluetooth can be subjected to a variety of attacks. The Bluetooth module of smartphone has to stay active all the time, which is certainly not power efficient for mobile devices.

Muppavarapu *et al*. [16] proposed an anti-phishing techniques to identify phishing websites using a

combined approach by constructing Resource Description Framework (RDF) models and using ensemble learning algorithms for the classification of websites with a true positive rate of 98.8%, which is definitely appreciable. As they have used random forest classifier that can handle missing values in dataset, they were able to reduce the false positive rate of the system to an extent of 1.5%.

3. Design Goals and Assumptions

Based on the security, deployability and usability analysis that we performed on current authentication methods, we have drawn a set of goals that our authentication scheme needs to achieve, including:

1. Apply the principle of mutual authentication of both the user and the server to eliminate replay attacks, Man in the middle attacks and phishing attacks.
2. Involve minimal possible user intervention in the second authentication factor.
3. No changes are needed on the user's PC or mobile phone for the authentication protocol to work.
4. No operation costs are added on the web application vendor or the user, as the communication between the web application and the user's mobile is initiated through the free Google Cloud Messaging service (or any other free cloud messaging service, e.g. Pushy).
5. The protocol should implement a fall-back mechanism to enable the user to access his account (with less privileges) in case of being not able to use his mobile phone, or in case the GCM notification service fails.

4. Protocol Architecture

Our mobile based authentication protocol meets its design goals by mutually authenticating the web application to the client and the client to the web application using Public Key Infrastructure PKI, session-based OPT and mobile Identity which is used to uniquely identify a mobile device International Mobile Equipment Identity (IMEI). Figure 2 shows the general steps involved in the authentication process.

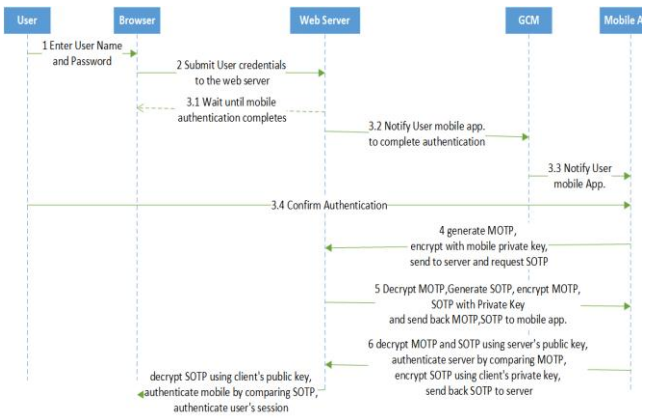


Figure 2. User authentication sequence diagram.

Based on the notations in Table 2, the authentication protocol steps are depicted in Figure 2, and explained as the following:

- *Step 1*: the user enters his credentials (U and PWD) via the login form, and then submits them to the server for verification.
- *Step 2*: the server verifies the received user's credentials. Upon negative verification, the login attempt fails and the user is notified through his browser. Upon success, the authentication protocol proceeds to step 3.
- *Step 3*: The server responds with a wait response for the client browser, informing him that he needs to complete the authentication process from his mobile device. The server looks up the registration token (RegToken) associated with the user (explained later) from local database, and sends a login notification to the user's mobile application via GCM; GCM will deliver the notification message (includes LID) to the user's mobile application directly.
- *Step 4*: once the login notification message is received by the user's mobile application, the application will show a confirmation dialog so that the user decides whether to allow and complete the authentication process or not. If the user rejects the login attempt, an encrypted (with MPR) negative response message (NLR) is sent back to the server over a secure connection SSL channel indicating the failure of the authentication, then the server verifies the response and responds to the user browser with a failure of authentication message. If the user accepts the login attempt, the protocol proceeds to step 5.
- *Step 5*: the mobile application generates a random code (MOTP), encrypt it with its private key (MPR) and sends a request to the server containing the encrypted MOTP, IMEI and LID to get the authentication token.
- *Step 6*: the server decrypts and encodes the received token, generates a OTP associated with the current login session Server One-Time Password (SOTP), specify its validity period, combine them as an authentication token, stores it locally, encrypts it using the server's private key (SPR) and sends back the encrypted authentication token to the client mobile application over a secure channel.
- *Step 7*: the mobile application receives the encrypted authentication token, decrypts it using the server's public key (SPU), decode the result and compare it with previously sent MOTP Then checks its validity and encrypt it using its own private key and sends it back to the server over a secure channel.
- *Step 8*: the server decrypts the received authentication token using the client's public key, verifies it and compares it with the previously stored OTP (SOTP) for the current session id. If the two

OPTs match, the server authenticates the current user session and notifies the user's browser to be redirected to the website main page.

Table 2. Table of notations.

Notation	Description
U	User name
PWD	User password
LID	Login Attempt ID
MOTP	Mobile One-Time Password
SOTP	Server One-Time Password
RegToken	Mobile App. Registration Token in GCM
GCM	Google Cloud Messaging Service
PID	Project ID that identifies the web application in GCM
SPU	Server Public Key
SPR	Server Private Key
MPU	Mobile Public Key
MPR	Mobile Private Key
NLR	Negative Login Result
IMEI	User mobile phone International Mobile Equipment Identity which is used to uniquely identify a mobile device.
NMSG	Notification Message

- *Assumptions and notes*: The following assumptions are made to supplement the general architecture of our authentication protocol: The protocol relies on GCM for notifying the user's mobile application of the current login session so that the user completes the authentication steps. User's sensitive data are never communicated through GCM. We use GCM as a notification service to enhance the user convenience and improve usability features in the authentication protocol. GCM is not considered a proprietary third party for two reasons: other free notification services could be used as well (e.g., Pushy [18]) and an alternative option is to adopt the design of enabling the user himself to initiate the process of completing the authentication using the mobile application. The authentication protocol implements a fall-back mechanism to enable the user to bypass it in case his mobile is not accessible at the time of login. The generated authentication token is session-specific; i.e., it is valid for the current user login session only, in case an attacker compromises it, it will be no longer valid for any other session initiated by the attacker.
- *Registration Phase*: In order for the authentication protocol to be activated, a set of initialization and registration steps are needed from both the client and the server, summarized as follows:

1. The web application needs to be associated with a project id in the GCM server.
2. The client's mobile application should be installed on the user's mobile.
3. The user needs to associate his mobile device with his web account using the mobile IMEI number, so that any login attempt from a mobile application will be verified using the user credentials and the mobile IMEI number.
4. The user needs to login on the mobile application using his username/ password pair.
5. In the first login to the mobile application, the application will register itself on GCM under the project id that was associated with the web application on point 1 above. Upon registering, GCM will generate a registration token associated with the mobile application who in turn will send it to the web application to be stored and paired with the user name to be used later for sending notifications from the web application to the user's mobile application.
6. The web application needs to generate an RSA key pair (private and public keys); the private key is stored securely in the server's key store; the public key is distributed to the clients in a secure key distribution mechanism.
7. The mobile application needs to generate an RSA key pair (private and public keys), store the private key in a securely in the mobiles key store.
8. The mobile application sends its public key, registration token and the mobile IMEI to the server to be associated and verified with the user's account.
9. We assume that no attacks happen during the registration phase for both the client and the server.

Figure 3 depicts the general steps involved in registering and sending notifications using GCM service.

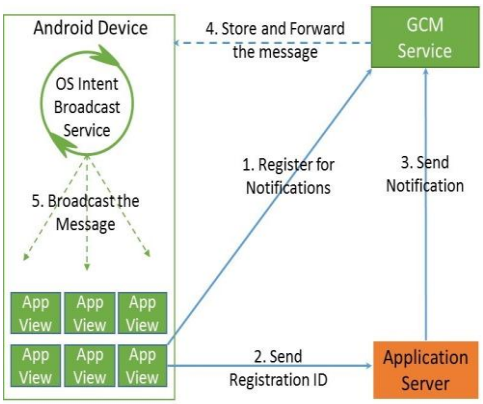


Figure 3. GCM collaboration framework [12].

- *Protocol implementation*: We have implemented a prototype system for our authentication protocol, including a web application and mobile application configured to use GCM. We built a java-based web server that handles a set of server side services for the authentication protocol, including: server registration with GCM, OTPs generator, registration and handling communication with the client mobile.

The client's mobile application is developed on android 5.1, and it compatible with android 2.2 and upward platforms as no special APIs are used except for support to GCM [12]. The mobile application is responsible for device registration, confirming and completing the authentication process, the application uses the necessary APIs for RSA key generation,

storage, encryption, decryption and communications with the server over SSL.

Figures 4, 5, and 6 show screenshots of the authentication process, Figure 4 shows the website login page where the user enters his credentials, figure 5 depicts the waiting message displayed on the user's browser until he completes the authentication from his mobile, Figure 6-a shows the login notification that appears on the user's mobile application upon the arrival of the notification message, in this notification the user decides whether to authenticate the session or reject it.

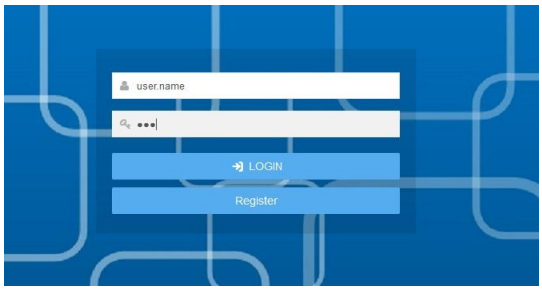


Figure 4. website login page.

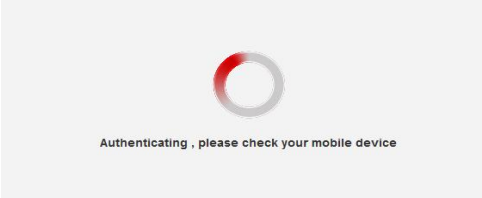
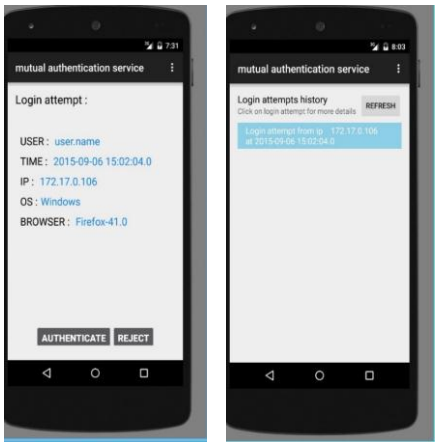


Figure 5. User authentication waiting message.



a) Auto. login notification. b) Manual login completion if GCM notification fails.

Figure 6. Login Notification.

- *Protocol fall-back mechanism*: Our authentication protocol can fall back to user's credentials only scheme in case the user's mobile is not compatible or inaccessible at the time of authentication. To strengthen the traditional password based authentication, the fall-back mechanism is supplemented by an SMS-based or email-based OTP that is delivered to the user to enter it in addition to his username/password credentials. The web application needs to treat login sessions in the fall-back mechanism in a less privileges mode; for example, the user will have limited authorizations on critical services such as resetting user password, financial transactions, etc.,

If GCM notification service fails, no notification message is received on the user mobile application to guide him through the authentication process. In this case, the user can initiate a request from his mobile application to query the available login attempts that are pending to be authenticated, and then continue the authentication process as usual, as depicted in Figure 6 (b).

- *Protocol management*: After putting the authentication protocol in production, a set of mechanisms and policies need to be defined to manage user's registration to and revocation of the authentication protocol features.
- *Trusting more than one device per user*: our authentication scheme supports trusting more than one mobile device for the user account, the user needs a full privileged active session to register a new device and associate it with his account, the user adds the IMEI of the device and associate it with his account, install the mobile application on the new device, register the mobile application with GCM service and sends the registration token, device IMEI, device public key to the server for verification and acceptance. In the multi device mode, when an authentication attempt is initiated from browser, all the trusted and linked mobile applications on all trusted devices are notified of the login attempt through GCM group messaging service. (no overhead is introduced on the server). Then the user can confirm and complete the authentication process from any device.
- *Revocation or unregistering a device*: users may want to revoke a device and disassociate it from their account (in case it is stolen, changed …). The revocation process could be done either from a fully authorized web session or by requesting that from the website vendor.
- *Protocol Evaluation*: We evaluated our authentication protocol using the web authentication assessment framework proposed by Bonneau *et al*. [3], we present an analysis of the 25 metrics of the usability, deployability and security of an ideal authentication scheme. In addition to analysing our protocol, we compare it to another four popular authentication schemes; passwords, google 2 step-verification [13], PhoneAuth [6] and CamAuth [20], the comparison results are shown in Table 3.

Table 3. Comparison of EARMAT protocol, passwords, GOOGLE 2-STEP VERIFICATION (2SV), PHONEAUTH (IN STRICT MODE) and CAMAUTH. Y=offers the benefit, S=somewhat offers the benefit.

	Usability Features								**Deployability Features**						**Security Features**									
z	Memorywise-Effortless	Scalable-for-Users	Nothing-to-Carry	Quasi-Nothing-to-Carry	Easy-to-Learn	Efficient-to-Use	Infrequent-Errors	Easy-Recovery-from-Loss	Accessible	Negligible-Cost-Per-User	Server-Compatible	Browser-Compatible	Mature	Non-Proprietary	Resilient-to-Physical-Observation	Resilient-to-Targeted-Impersonation	Resilient-to-Throttled-Guessing	Resilient-to-Unthrottled-Guessing	Resilient-to-Internal-Observation	Resilient-to-Leaks-from-Other-Verifiers	Resilient-to-Phishing	Resilient-to-Theft	No-Trusted-Third-Party	Requiring-Explicit-Consent	Unlinkable
Passwords		Y	Y	Y	Y	S	Y	Y	Y	Y	Y	Y	Y	Y		S						Y	Y	Y	Y
Google 2-step Verification			Y	Y	S	S	S	S	S			S	Y			S	Y			Y	Y	Y	Y	Y	Y
PhoneAuth – strict			S	Y	Y	S	S	S	Y	S	S	S		Y	Y	Y	Y	Y	S	Y	Y	Y	Y	Y	S
CamAuth			S	Y	S	S	S	S	S	S	S	S		Y	Y	Y	Y	Y	S	Y	Y	Y	Y	Y	Y
Our Protocol		S	S	Y	Y	Y	Y	S	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

The evaluation of our authentication protocol assesses its usability, deployability and security features. In terms of Usability, our authentication protocol, like others, assumes that the user memorizes his password, so it is not Memorywise-Effortless. In regards to Scalable-for-Users property, the current implementation of the protocol assumes that each web Application needs its own mobile application to be installed on the client mobile to complete the authentication process; we rate the protocol as Somewhat scalable for users based on that it is easy to manage more than one application in the user's mobile, taking into account that the user's intervention in the authentication process is limited to responding to the authentication confirmation request only. It is theoretically and practically feasible to alter our model such that only one mobile application is used to manage the user's authentications on more than one web application. And this will be our future work.

The protocol achieves somewhat Nothing-to-Carry property, and fully achieves the Quasi-nothing-to-carry usability feature, as mobile phones are ubiquitous these days.

The properties of Easy-to-Learn, Efficient-to-Use and Infrequent-Errors are also achieved as mobile applications in general are very common nowadays.

Easy-Recovery-from-Loss is Somewhat offered by the protocol, like other authentication schemes, our protocol will work with SMS-based or email-based OTP in case of failure to use the second authentication factor.

In terms of Deployability features, our authentication protocol is superior to Google 2SV, PhoneAuth and CamAuth schemes in achieving the Accessible, Negligible-Cost-for-Users, Server-Compatible, and Browser-Compatible features. Our protocol introduces zero configurations or changes to the user's browser, web server O.S. or mobile O.S. In addition to adding zero cost for either the website vendor or the user. For the Mature property, we think our protocol is able to be a mature authentication scheme, but as this requirement is measured after putting the protocol in production environment; we cannot in this phase empirically verify the mature property. The Non-Proprietary feature of our authentication protocol is achieved, no proprietary software, hardware or service is necessary for the protocol to work successfully; for GCM, we use it as a communication medium only, and it is implemented as one of existing set of free alternatives including Pushy service [18].

In terms of Security features, our authentication protocol is resilient to physical Observation, Targeted Impersonation, Throttled and Unthrottled Guessing, because the attacker will not be able to access the user's account even if he possesses his password until he gains access to his mobile device. Also, attacking the generated authentication token (including the OTP) will not enable the attacker to gain access to the user's account because the OTP is session-specific and not valid for any other web session. The protocol is also resilient to Internal Observation and to leaks from other verifiers, this resilience is achieved due to the session-specific OTP, data communication is carried over secure channels and Private keys are stored in protected areas in the mobile phone (either hardware protected areas (if supported by the mobile) or system key store). Our protocol is certainly resilient to phishing attacks and Theft due to two factor authentication.

Our protocol achieves the No-Trusted-Third-Party feature, as its dependence on GCM is for convenience purposes; i.e., showing up a notification to the user that an authentication process on his account is being done; while this feature is very important for online notification of possible authentication attacks; it is possible to deactivate this feature and rely on the user himself to start the mobile authentication steps. In addition, as mentioned earlier, GCM could be replaced by another services that offer the notification service.

Our protocol clearly achieves the Requiring-Explicit-Consent and Unlinkable feature, as the authentication is completed after user confirmation; i.e. no authentication process can be completed on the user's account without his explicit acceptance.

5. Performance Evaluation

To assess the performance of EARMAT, we conducted a performance test using an emulator in android studio with the following specifications: Device: Nexus 5, CPU: x86, RAM: 1.5 GB, Platform Version: Android 5.1 (lollipop). In this evaluation we measured response time and memory usage of the EARMAT mobile app; the results of the performance test showed that EARMAT -in average- spent 4.3 Milliseconds to complete the decryption and encryption processes, while consuming 0.06 MB of memory. The whole memory reserved by the application was 2.38 MB. Table 4 depicts the test results of six runs of the algorithm of decrypting and encrypting process using the emulator.

The complete authentication process will include also the time spent in communications between the web server and GCM server which in turn notifies the user's mobile application in an automated process that is expected to add a very little time fraction (in milliseconds).

Table 4. Performance test of EARMAT in the emulator.

Run #	Execution Time (ms)	CPU Usage	Memory Usage (MB)
1	5	11%	0.08
2	3	12%	0.06
3	3	12.5%	0.07
4	6	9.5%	0.06
5	6	14%	0.05
6	3	20%	0.05

These performance test results indicate that EARMAT implementation in most modern mobile devices will be feasible and the response time will be accepted by users, making it possible to adopt such an authentication scheme at a compromise of a couple of seconds latency.

6. Conclusions

In this paper, we analysed the web security threats regarding phishing and stealing users' private data, and reviewed and discussed the most recent 2FAschemes that are proposed to secure the authentication process and prevent compromising users' accounts. We also introduced a new user transparent 2FA scheme that augments the security of web authentication leveraging the ubiquitous mobile phones; our 2FA protocol realizes 2FA by achieving mutual authentication of both the web site client and the server by implementing RSA cryptography on the communicated authentication messages. Our protocol defeats Man in the Middle attacks in addition to phishing attacks. We implemented a system prototype of the authentication protocol to ensure its feasibility.

In addition, we evaluated the protocol against the 25 features of the assessment framework regarding Usability, Deployability and Security of 2FA schemes. In future, we plan to extend our protocol to support single mobile application to manage a set of user account to further improve the protocol scalability of users' property.

References

[1] Abu-Nimeh S., Nappa D., Wang X., and Nair S., "A Comparison of Machine Learning Techniques for Phishing Detection," *in Proceedings of the Anti-phishing Working Groups 2nd Annual eCrime Researchers Summit*, Pittsburgh, pp. 60-69, 2007.

[2] Anti-Phishing Working Group. http://www.antiphishing.org/, Last Visited, 2015.

[3] Bonneau J., Herley C., Oorschot V., and Stajano F., "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes," *in Proceedings of the IEEE Symposium on Security and Privacy*, San Francisco, pp. 553-567, 2012.

[4] Cao Y., Han W., and Le Y., "Anti-phishing based on Automated Individual White-list," *in Proceedings of the 4th ACM Workshop on Digital Identity Management*, Alexandria, pp. 51-60, 2008.

[5] Cronto, www.cronto.com/, Last Visited, 2016.

[6] Czeskis A., Dietz M., Kohno T., Wallach D., and Balfanz D., "Strengthening user Authentication Through Opportunistic Cryptographic Identity Assertions," *in Proceedings of the ACM Conference on Computer and Communications Security*, Raleigh, pp. 404-414, 2012.

[7] Dhamija R., Tygar J., and Hearst M., "Why Phishing Works," *in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Montréal, pp. 581-590, 2006.

[8] Dmitrienko A., Liebchen C., Rossow C., and Sadeghi A., "Security Analysis of Mobile Two-Factor Authentication Schemes," *Intel Technology Journal*, vol. 18, no. 4, pp. 138-161, 2014.

[9] Dodson B., Sengupta D., Boneh D., and Lam M., "Secure, Consumer-friendly Web Authentication and Payments With a Phone," *in Proceedings of International Conference on Mobile Computing, Applications, and Services*, Santa Clara, pp. 17-38, 2010.

[10] Downs S., Holbrook M., and Cranor L., "Decision Strategies and Susceptibility to Phishing," *in Proceedings of the 2nd Symposium on Usable Privacy and Security*, Pittsburgh, 2006.

[11] Gal´an E., Castro J., and Alcaide A., and Ribagorda A., "A Strong Authentication Protocol based on Portable One-Time Dynamic URLs," *in Proceedings of International Conference on Web Intelligence and Intelligent Agent Technology*, Toronto, 2010.
[12] Google Cloud Messaging, https://developers.google.com/cloud-messaging/gcm, Last Visited, 2015.
[13] Google, "Google 2-step Verification," http://www.google.com/landing/2step/, Last Visited, 2015.
[14] James D. and Philip M., "A Novel Anti Phishing framework based on Visual Cryptography," *in Proceedings of International Conference on Power, Signals, Controls and Computation*, Thrissur, pp. 1-5, 2012.
[15] Method and apparatus for positively identifying an individual, http://www.google.com/patents/US4720860, Last Visited, 2015.
[16] Muppavarapu V., Rajendran A., and Vasudevan S., "Phishing Detection using RDF and Random Forest," *The International Arab Journal of Information Technology*, vol. 15, no. 5, pp. 817-824, 2018.
[17] Prajitha M., Rekha P., Amrutha A., "A Secured Authentication Protocol Which Resist Password Reuse Attack," *in Proceedings of International Conference on Innovations in Information Embedded and Communication Systems*, Coimbatore, pp. 1-5, 2015.
[18] Pushy, https://pushy.me/, Last Visited, 2015.
[19] The Phishing Guide Understanding & Preventing Phishing Attacks, http://www-935.ibm.com/services/us/iss/pdf/phishing-guide-wp.pdf, Last Visited, 2015.
[20] Xie M., Li Y., Yoshigoe K., Seker R., and Bian J., "CamAuth: Securing Web Authentication with Camera," *in Proceedings of IEEE 16th International Symposium on High Assurance Systems Engineering*, Daytona Beach Shores, pp. 232-239, 2015.
[21] Umadevi P. and Saranya V., "Stronger Authentication for Password using Virtual Password and Secret Little Functions," *in Proceedings of International Conference on Information Communication and Embedded Systems*, Chennai, pp. 1-6, 2014.

Adwan Yasin is a full professor, Former dean of Faculty of Engineering and Information Technology of the Arab American University of Jenin, Palestine. Previously he worked at Philadelphia and Zarka Private University, Jordan. He received his PhD degree from the National Technical University of Ukraine in 1996. His research interests include Computer Networks, Computer Architecture, Cryptography and Networks Security.

Abdelmunem Abuhasan is a Master student at the Arab American University with particular interests in computer security, web security and software engineering. He is working since ten years as the manager of software development department at the Arab American University. He holds a B.A. in Computer Science from the Arab American University.