Out-of-Band Authentication Using 2-Factor Image Matching

H. Karen Lu, Asad Ali, Benoit Famechon, and Najam Siddiqui
Digital Identity & Security, Thales
Austin, TX, USA

Abstract - This paper presents a unique approach to online multi-factor authentication called 2-Factor Image Matching (2FIM). While existing solutions focus on only one side of the security vs. usability equation, our approach seamlessly blends the two. It delivers a robust and intuitive out-of-band authentication that is as simple as matching two images. However, this simple manual image match by a user is backed by strong cryptographic protocols. The paper describes the constraints and challenges of out-of-band authentication and channel binding, and how our approach addresses them. We illustrate the application flows for authentication and device registration use cases, and explain the underlying security and cryptography of such a solution. Finally, we compare it with commonly used solutions. We also show how push notifications and other spam messages can be avoided to improve existing industry specifications, such as MobileConnect from GSMA.

Keywords: Authentication, multi-factor, out-of-band, mobile device, spam prevention

1. Introduction

How do you prove who you are in an online transaction? While the use of Internet and online service industry has grown exponentially over the last two decades, we still seem to be struggling to implement an optimum method of authenticating users. Passwords are universally considered a weak form of authentication, but are still widely used. The reason is simple; there is no other option that adds security without also adding complexity and inconvenience for the user. For example, use of secure elements that rely on Public Key Infrastructure (PKI) to deliver two-factor authentication (2FA) certainly enhances security, but forces users to carry a dedicated device, such as a smart card. The ubiquity of smart phones has at least solved part of the problem. Users no longer have to be forced to carry a 2FA device, but rather happily carry their phones, which can be used as a second factor device for authentication. However, even use of smart phones as 2FA devices can encumber the login flow, if not done right.

To solve the above problems, this paper presents a unique approach to online multi-factor authentication, called 2-Factor Image Matching (2FIM). The 2FIM provides another authentication factor, “what you have”, which is typically a user’s smartphone. Combining 2FIM with the main authentication factor, e.g. password, makes a 2FA. The 2FIM is typically used with two devices, an access device, e.g. a Personal Computer (PC), and a smartphone. It provides better security as well as usability. As we will explain, there are several design decisions to consider for achieving an optimum mix of security and usability including: number of steps or actions required of the user, channel binding between login device and the 2FA device, and preventing unsolicited spam notifications to the 2FA device.

The rest of the paper is organized as follows. Section 2 provides background to the problem. It introduces out-of-band channel, how it can be tied to main login channel, and identifies a flaw in the out-of-band authentication flow proposed by GSMA [1]. Section 3 explains the detailed design of our proposed solution to address these aforementioned problems, along with the authentication flows. Section 4 outlines some development choices and the approach we took to implement our solution. Section 5 offers a security and usability analysis of our design. Finally, we offer our conclusions and suggest a future direction of research in section 6.

2. Background and Related Work

Using mobile phone for out-of-band authentication brings some challenges. Let’s first introduce some of the concepts required to understand the pain points of such solutions before we go into details on how our proposed design addresses them.

2.1 Out-of-Band Channel

When accessing an online resource, a user can use a computer, a tablet or even a mobile phone. For authenticating a user, a mobile device can be used in addition to or instead of a username and password. The out-of-band authentication channel is called out-of-band because it uses a communication channel that is different from the primary channel used for accessing online resources. For example, if a user connects to a web server using a web browser from a PC, the primary channel of communication is Hypertext Transfer Protocol Secure (HTTPS). The web server can ask for a second form of authentication over a separate channel such as cellular network connecting to the smart phone.
The following are some examples of out-of-band authentication:

1. Text Short Message Service (SMS): The authentication server sends a simple text message (SMS) code to user’s mobile device and asks the user to enter this code on the screen of user’s primary access device, e.g. PC

2. Text code input: The authentication server displays a text string on the browser, and asks the user to enter same text on the phone. The application (app) on the phone then connects back to the authentication server to confirm that the user has entered the correct code.

3. Quick Response (QR) code scanning: The authentication server displays a QR code on the browser, and asks the user to take a picture of this code with this mobile phone. The app on the mobile phone then relays the information in QR code back to the authentication server as a proof of user confirmation of login or transaction.

4. One Time Password (OTP), with optional push notification: The user’s mobile has an OTP app. The authentication server login page asks the user to either generate an OTP using the app and enter it to the login page or, have the app generate an OTP and send it to the authentication server.

Now that we have explained the use of separate channels for accessing online resource and for authenticating a user, let us see how the communication on these two channels is reconciled. This is referred to as channel binding.

2.2 Channel Binding

Channel binding is the capability implemented in an authentication server to recognize and match the following two distinct channels of communication:

1. The session started by the user to access the online resource on a first channel.
2. The user’ out-of-band authentication from the mobile device on a second channel.

Without channel binding, the authentication server cannot claim it supports Multi-Factor Authentication (MFA) using out-of-band channel.

The example of out-of-band authentication described earlier improves security but can also be abused, specifically if, for user convenience, the text/code or authentication request is provided to the user via text SMS or mobile push notification.

Since usernames are often easily guessed (e.g. email and phone number are public information, but often used as usernames), they can be entered on a login page by anyone with malicious intent. As a result, users could be bombarded with unwanted MFA authentication request notifications on their phones. In other words they could be spammed. A user may accidentally accept some of the malicious authentication requests, which may allow an attacker to access user’s online account.

To address this spam, an industry solution has been proposed by GSMA, called MobileConnect spam code.

2.3 GSMA Spam Code

The GSMA MobileConnect specification enables Mobile Network Operators (MNOs) to act as Identity Providers so that users can authenticate to Service Providers using their mobile devices. Acknowledging that such use of a mobile phone can be open to spam, GSMA has proposed a workaround. It requires that users enter an alphanumeric code called a Personal Access Code (PAC) before a notification is sent to their phone.

However, this solution is detrimental to user experience for two reasons: Firstly, the user has to remember this PAC, in addition to an already existing Personal Identification Number (PIN) code used for authorizing transaction in the MobileConnect solution. This can easily become confusing for the user. Secondly, there is now an additional step in the authentication process; typing the PAC. The design we propose addresses this.

2.4 Image-based Authentication

Image-based authentication methods [2][3][4] lets a user select an image or images to authenticate himself to the server. The reference [5] uses a scrambled keypad, which changes each time, for user authentication. For comparison purpose, we can consider the numbers in the keypad as images. The Windows 10 picture password [6] allows a user to select a picture and make a gesture on the picture as a way to authenticate.

The 2FIM authentication is different from existing image-based authentication methods in three different aspects.

1. The 2FIM is out-of-band authentication, which uses another communication channel that is different from the main communication channel for additional authentication. Existing image-based authentication method uses one channel.

2. The 2FIM uses two devices, an access device, e.g. a PC, which allows a user to interact with the remote server and a user’s mobile phone for the user to select images. Existing image methods involve one access device, at which the client software, e.g. a web browser runs. Either the user uses the client to send a user pre-registered image as a password [2], or the server send a set or sets of images to the client.
for the user to pick one or more pre-registered images [3].

3. In 2FIM, the client response is different each time. In the existing methods, the client always sends the same data, i.e. image password, to the server. The rest of the paper will show that 2FIM offers better security and usability.

3. Authentication with 2FIM

This paper proposes a new out-of-band authentication interaction flow, call 2-Factor Image Matching (2FIM) to solve the problems mentioned above. The new approach does not rely on any of the existing methods; text SMS, push notification, use of camera for QR code, or entering a text code on mobile device. Each of these existing methods has its own user experience flows, as we will explain later in section 5.1. In our proposal the user initiates the communication and has a simple one-click interaction with his mobile phone. The phone then sends a message to the authentication server (2FIM Server) using strong cryptographic keys. Figure 1 shows the parties involved in this design.

![Figure 1. High level authentication flow](image)

At a high level, the authentication flow includes the following steps:

1. A user opens a browser on an access device, i.e. a PC, and goes to the authentication server, which supports 2FIM, to authenticate, and enters a user identifier.

2. The user opens the 2FIM App on his phone and provides consent to login to the 2FIM authentication server.

3. The 2FIM App on the phone sends the user consent to 2FIM authentication server to complete the login.

3.1 Design

The above seemingly simple solution has a difficult challenge. How does the authentication server know that the acceptance coming from user’s mobile phone (step #3 in Figure 1) corresponds to the session started by user in step #1? To address this issue, we need to associate the following two separate channels of communication.

1. Channel between an access device used by the user and the authentication server. This is indicated with step #1. The access device is typically a web browser on a PC talking to the authentication server. User wants to be authenticated over this channel, and establish a login session.

2. Channel between the user’s mobile phone and the authentication server. This out-of-band channel is used to confirm user action to login or authorize a transaction. This is indicated in step #3.

Existing approaches, e.g. text code and QR code, for associating these two channels have cumbersome user experience. Our method for the association is simpler and faster.

During the 2FIM-based authentication, the authentication web page displays an image (it can be a picture, photo, icon, text, numbers or any graphical information) on the screen of the user’s access device, from which the user is trying to access the authentication server. The user will need to match this image on the screen with one of the images from the 2FIM App on his mobile device based on a specified matching rule, for example, finding the same, similar, or opposite image.

Each image is associated with a dataset that contains information for assisting the authentication, for example challenge, transaction identifier, validity date, etc. Those datasets are either one-time use only or reusable based on a policy. Once the one-time use only option is used, datasets are removed from the 2FIM mobile app as well as from the backend server.

The dataset can be embedded within the graphical display. In this case the 2FIM App can retrieve the dataset (challenge, transaction identifier, etc.) to be used for cryptographic operation by extracting them from the image. Alternatively, it can also reference an image. For example, a mobile device has a set of images and associated dataset already for authentication. We can refresh the dataset without changing the image set. In this case, the dataset could refer to the image to be displayed so that the application can match image to its associated data.

Based upon the image selected, the associated or embedded dataset is used as input for the derivation of cryptographic keys to encrypt the message sent from the 2FIM App to the authentication server. Section 3.4 describes the details of the derivation and encryption.
3.2 Mobile Device Registration

To use the mobile device for multi-factor authentication, the user needs to register the mobile device to the authentication server. This section describes the mobile device registration and how images and datasets are loaded in the 2FIM mobile App.

Figure 2 describes the steps for mobile phone registration (aka binding). This is required before mobile phone can be used in online authentication. It shows how device specific keys are loaded into the 2FIM mobile App, along with device specific image sets. For other mobile devices, such as tablets, a comparable registration flow can be created.

1. A user opens a web browser on the PC (which will be the access device) and authenticates to the authentication server (AS).
2. The user then initiates the process to bind his mobile phone to his online account at AS.
3. First step in this binding process is to enter the mobile device phone number.
4. Optionally the AS can use this phone number to send an SMS link to mobile device. This link is to download the 2FIM mobile App. Alternatively, the user can search for the 2FIM mobile App from the mobile phone’s app store.
5. The user downloads the 2FIM App to the mobile phone. This App is a vanilla App, with no user specific data. It does however, contain 10 stock image sets. All downloaded 2FIM Apps will have the same images. In addition the App has a shared secret key $K_s$ which is also the same for all the 2FIM Apps. This shared secret is only used for bootstrapping during registration. The App will be customized to user specific keys in the following steps.
6. To initialize the 2FIM App, user enters his unique User Identifier (UserID), e.g. phone number. Or the App could automatically get the number from the phone.
7. The AS in the meantime has displayed a series of 4 images. These images are picked randomly from the list of stock image each 2FIM App has. The user is asked to open his 2FIM mobile App and select the same images, in the same order.
8. The user opens the 2FIM App (downloaded and initialized in steps 5, 6). The user then selects the same image sequence on the phone as shown on the PC browser screen (Step 7 above).
9. Based on this selection, the 2FIM App computes two session keys: $K_e$ and $K_m$. These two keys are used to establish a secure connection with AS. $K_e$ is for encryption, and $K_m$ is for message integrity. Computation of $K_e$ and $K_m$ is based on a key derivation function that takes three inputs: UserID, image pattern selected by user, and $K_s$. All three values are known to the backend server, AS. As such authentication server can also compute the same values of $K_e$ and $K_m$. This makes secure handshake indicated in steps 10, and 12 possible.
10. The mobile App sends a message to authentication server to bind the 2FIM App to user account. The message is constructed as follows:
    
    $\text{Message} = \text{UserID} + \text{Encrypt} \{ \text{PhoneNumber + UserID} \} K_e, K_m$

    Note that the phone number and UserID are encrypted using the generated session keys, $K_e$ and $K_m$. UserID in clear is prepended to the start of the message so authentication server can use it to compute $K_e$ and $K_m$ itself. The entire message is sent over TLS which offers transport level security. As such the clear UserID is still protected in transit by TLS.
11. The authentication server reads UserID from start of the message, and uses this to generate session keys $K_e$ and $K_m$ itself. It then uses these keys to decrypt encrypted payload and matches the UserID and phone number. Once this is confirmed, it now generates two device specific keys for this device: $K_{d-e}$ and $K_{d-m}$, one
for encryption and one for integrity of messages. These keys are stored on the server, indexed to the UserID.

12. These new device keys are sent to the 2FIM App. This exchange is still protected by session keys, $K_e$ and $K_m$.

13. The 2FIM App stores these device keys locally. All subsequent exchange with authentication server will be done using these device specific keys.

14. The first use of these device keys is to get device specific images from the server. The server either generates the image or pulls it from its database. In either case images are stored on the server, index to the UserID, and also sent to the mobile device.

15. The 2FIM App stores these images locally. The authentication will use these new images instead of the stock images that came with App.

16. The authentication server indicates to user that device binding is completed. The user can now use his mobile phone for strong out-of-band authentication.

3.3 Authentication

This section describes authentication details. It relies on the device specific information already exchanged during device registration described in section 3.2.

Figure 3 describes the steps for user login (aka authentication) using the 2FIM App.

1. User opens web browser on a PC and points to the login page of the AS. This is the server to which the user wants to authenticate.

2. User enters his unique ID. This can be a username or a phone number. The authentication server identifies the mobile device to which this account binds to. This binding should already have been completed.

3. The authentication server displays a single image in the browser window. See Figure 4 for a visual example. This image is taken from $N$ images that are already provisioned on the 2FIM App. User is instructed to open the mobile device application and to match the displayed image with a corresponding image shown on the mobile phone according to a matching rule.

4. The user opens his 2FIM App from his mobile phone.

5. The user then selects the image shown in step #3, from the many images in the 2FIM App. See Figure 5 for a visual example of this display.

Figure 3. Authentication

Figure 4. Display on user’s access device

Figure 5. Display on user’s mobile device
6. Once user has selected the image, the 2FIM App derives two session keys: $K_{s-e}$ and $K_{s-m}$. One is for encryption and one for integrity of message to be sent to authentication server. The derivation is based on a key derivation function that takes three inputs: image selected, and device key $K_{d-e}$ or $K_{d-m}$, and a random nonce. The nonce ensures that the input to session key derivation are different each time, even if the same image is selected for multiple authentication attempts. As such the session key will always be different.

7. Using the session keys, $K_{s-e}$ and $K_{s-m}$, the 2FIM App encrypts the response message. The contents of this response message include the confirmation to the login request, along with some meta-data. Timestamp can also be added to help prevent replay attacks.

8. The encrypted response is send back to the authentication server over Transport Layer Security (TLS). The nonce used in step 6 and the user Identifier (ID) are sent to authentication server in the “clear”, as part of the response header.

9. The authentication server knows the phone number from which the response is coming, and also gets UserID from response header. It first does its own derivation of session keys $K_{s-e}$ and $K_{s-m}$. It has all the information it needs to do this: UserID, nonce, and image that the user should have selected.

10. Using its own version of session keys, the authentication server now decrypts the response from the 2FIM App.

11. If response is valid and is ‘Yes’, user is allowed to login. Otherwise, authentication is rejected.

3.4 Key Derivation

This section describes how cryptographic keys are derived from image(s) selected by the user. Key derivation is done in step #9 of Registration flow (Figure 2) and step #6 of authentication flow (Figure 3). The key derivation function is as follows:

\[
\text{Derived Key} = PBKDF2(\text{PRF}, \text{SelectedImageData}, \text{UserID}, \text{IterationCount}, \text{KeyLength})
\]

Where,

- \(PBKDF2\) is the Password-Based Key Derivation Function 2, as described in PKCS #5 and RFC 2898.
- \(PRF\) is a pseudorandom function of two parameters, a key and a message. For example, HMAC-SHA1.
- \(\text{SelectedImageData}\) is the data associated with the image(s) that the user selected. For registration flow it is a sequence of 4 images selected from a set of 10 images (in the right order). For authentication flow it is the single image selected from the 6 available on the phone. In both cases, the authentication server is able to also perform the same image “selection” since it knows what image the user should have selected. Each image is associated with data, where length of data is configurable. For example it could be 32 bytes (256 bites) which is far more than data length of a character in a typical alpha-numeric PIN, 1 byte. This increases the entropy of possible derived keys.

- \(\text{UserID}\) is the user identifier.

- \(\text{IterationCount}\) is the number of iterations desired. This value is configured during registration.

- \(\text{KeyLength}\) is the length of the derived key. This key length must be set to twice the length of session key. The output key material can then be divided into two keys: one for encryption and one for MAC (message authentication code). For example, KeyLength must be 512 bits when using AES256 symmetric encryption.

Suffice to say that other key derive functions can also be used.

3.5 Refresh

To mitigate the risks of images and device keys being compromised, both the images and key material can be refreshed.

3.5.1 Image Refresh

The images and/or datasets associated with the images on a 2FIM App can be refreshed based on various policy options. For example, once used the image is deleted and a new image is sent from authentication server to the 2FIM App. Alternatively, the same images could be reused for subsequent authentications. The user, or AS could trigger a forced refresh when needed.

3.5.2 Device Key Refresh

Similarly, there can be several options for refreshing the device keys $K_{d-e}$ and $K_{d-m}$. Even though these are key derivation keys for deriving session keys (which then encrypt the message from App to authentication server), refreshing even the key-derivation-keys from time to time offers stronger security.

4. Implementation

To demonstrate viability and ease of use, authentication features of 2FIM were implemented as part of a proof of concept (PoC) project. This implementation included both server and mobile components. The server component relied on an open source identity provider server, KeyCloak and was developed in Java. The mobile application was built using the Android Software Development Kit (SDK). We chose Android over IPhone Operating System (iOS) due to the greater development and deployment flexibility offered by Android.
The focus of our PoC implementation was more on demonstrating the usability aspect of 2FIM. For this reason, some security related checks such as dynamic provisioning of images and re-synchronization cryptographic data were not completed. Although these security details are fully understood and defined in the 2FIM architecture, they were skipped in the interest of time. Instead we relied on a preset list of images that were used on both the server and the client. The authentication functionality was demonstrated to a wide audience at end of the PoC. The usability aspect of the solution were well appreciated by all. Some of these usability observations are described in section 5.1.

5. Analysis

The authentication scheme described in this paper brings improved usability for users while still delivering cryptographically secure two factor authentication for online access using mobile devices. Let’s analyze the approach’s the usability and security, respectively.

5.1 User Experience

To analyze the user experience of our authentication method, we first compare with existing methods conceptually, and then measures some metrics based on the PoC with internal users. Table 1 compares commonly used channel binding methods with ours (select image).

As reflected in our usability study, the 2FIM method offers the best user experience. It also prevents spam because out-of-band communication between user’s phone and authentication server is triggered by the user and not the server. Instead of the server sending a message or push notification to the mobile device, and asking for the response, it is the mobile device that unilaterally sends the “response” message. The server simply waits for it. Furthermore, the user is not asked to remember and enter an additional pass code.

The proposed design also reduces network messaging as there is no longer a push to trigger the authentication. As the user goes to a login page, he is aware/informed that he should use his 2FIM App to complete the authentication. There is an extra effort to open the App on the mobile phone, but users are accustomed to that in today’s digital age. The hint on when to do this is provided on the web page user connects from.

Let us measure various aspects of user experience: timing on how long it takes a user to complete the mentioned uses cases with various methods, and complexity of the user interaction like data to memorize and their frequency of use. We also looked at how the user experience complexity will evolve when security of the methods are increased.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>User Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Code</td>
<td>The authentication server displays a text code which the user can see in the web browser. The user then manually enters this code in the mobile phone. The code is used as challenge input for the computation of the cryptogram that is then returned to the backend server as user acceptance.</td>
<td>Manual typing of code can be cumbersome.</td>
</tr>
<tr>
<td>QR Code</td>
<td>The authentication server displays a QR code which the user can see in the web browser. The user then uses the camera feature in his phone to scan the QR code. Data extracted from this scan is used as input to generate cryptogram, as in method #1.</td>
<td>Use of camera can be cumbersome. It is also not intuitive for all users.</td>
</tr>
<tr>
<td>2FIM</td>
<td>The authentication server displays an image which the user can see in web browser. The user then matches this image to a set of images shown on his phone, and clicks on the matching image on the phone. Data extracted from this image or associated with this image is used to format and encrypt acceptance response to authentication server.</td>
<td>Selection of image by visual comparison is easy and fast.</td>
</tr>
</tbody>
</table>

5.1.1 User interaction complexity

Our time measurements to perform 2-factor authentication using three different authentication methods with varying cryptographic strengths show a pattern that is in line with general expectation. The authentication methods studied were SMS code, QR code and image authentication using 2FIM. Figure 6 summarizes the results of user interaction when 2-factor authentication is done with each of these methods for a range of cryptographic entropy, determined by the length of “key” in bytes. This key size if represented on x-axis, and the corresponding time taken by user to authenticate using that key size is on y-axis.
Figure 6: User experience compared to method complexity

SMS code where user manually enters the OTP code shows a linear increase in time. This is expected since it will take a user twice as long to enter an 8-character OTP as it would to enter a 4-character OTP. However, this method becomes quite cumbersome as characters increase, and is therefore generally capped at 8 characters.

QR code on the other hand can be used to generate larger key sizes without much impact on user experience. It add only a slight delay for large key sizes since these large key sizes make the QR code more compact and therefore slightly harder to scan. However, this method has a higher startup time of about 5 seconds regardless of key size. This is because of the time to start camera on smartphone and point to QR code, and focus it in the frame. As an extreme case, the phone may not be able to capture the high resolution QR code.

The image based authentication using 2FIM has a very little startup time, about 2 seconds, and it does not change regardless of the key size. This is because user interaction – selection of a single image on the smartphone screen – and the cryptographic data representing key material are two separate entities. The same image can be associated with data of any size. That is why graph of 2FIM authentication method is a horizontal line in Figure 6. While complexity of other methods increases with key size, 2FIM offers the same easy to use authentication regardless of the cryptographic strength selected.

5.1.2 Limitation

In order to prevent spamming, 2FIM does not use push notification. A user needs to manually open the 2FIM App on her mobile device, which some users may feel inconvenient. Future works could look at ways to remove this inconvenience perhaps by a mechanism to invoke the application on the mobile device very easily.

5.2 Security

The 2FIM-based authentication has several advantages from the security perspective. It prevents spamming of mobile devices, which is not possible with currently available solutions. Spam can cause security issues from at least two aspects. A user may accidentally accept some of the malicious authentication requests, allowing the attacker to access user’s account. A frustrated user may also intentionally turn off the 2-factor authentication due to annoying spam messages. In the context of GSMA Mobile Connect, 2FIM provides a simpler usability of the anti-spam feature by eliminating the need for the user to remember an additional specific code. It also simplifies its management: the feature is always ON, and admins do not have to wrestle with selectively enabling the feature for customers who complain about receiving spam requests on their phone. There is also no need for additional setup of Personal Access Code which is a hassle for users and customer care alike. As such it greatly improves the spam prevention proposal identified in GSMA MobileConnect specification.

The 2FIM-based authentication has a layered security protection, including security channel, key refresh, user response encryption, and so on, to ensure the authenticity, confidentiality, and integrity of the communications. The authentication challenge also changes at every authentication. These security controls are totally invisible to the end users. Furthermore, 2FIM also turns the often complex user registration into simple operations that are more secure and convenient.

The device keys are secrets that a 2FIM App must protect. The App can use the secure key store present in modern smartphones to keep the device keys. For better security, the device keys can be refreshed.

Although anyone who has access to the phone can authentication, this is no different from how other OTP apps work. For applications that require more stringent security, the app can require its own password or biometric.

6. Conclusions

In current digital era where applications and data are being pushed to the cloud, all access to these resources starts with some form of user authentication. This makes strong, intuitive and easy to use authentication an integral part of any system that protects resources and only grants access to authorized users. Users want security when accessing online portals for business or personal use, but not at the expense of usability, and they don’t want to receive unwanted messages on their devices. Enterprises allow their employees to use personal smartphones as two-factor authentication devices, as long as security is not compromised. The 2FIM-based authentication presented in this paper addresses all these concerns. It delivers an...
authentication solution that is simple to use, yet cryptographically secure. It also prevents spam messages to phones better than the approach suggested by GSMA specification. Future works include further improving the user convenience factor without compromising security.

7. Acknowledgements

The authors would like to thank Michael Hutchinson for his constructive comments and detailed review of the paper.

References