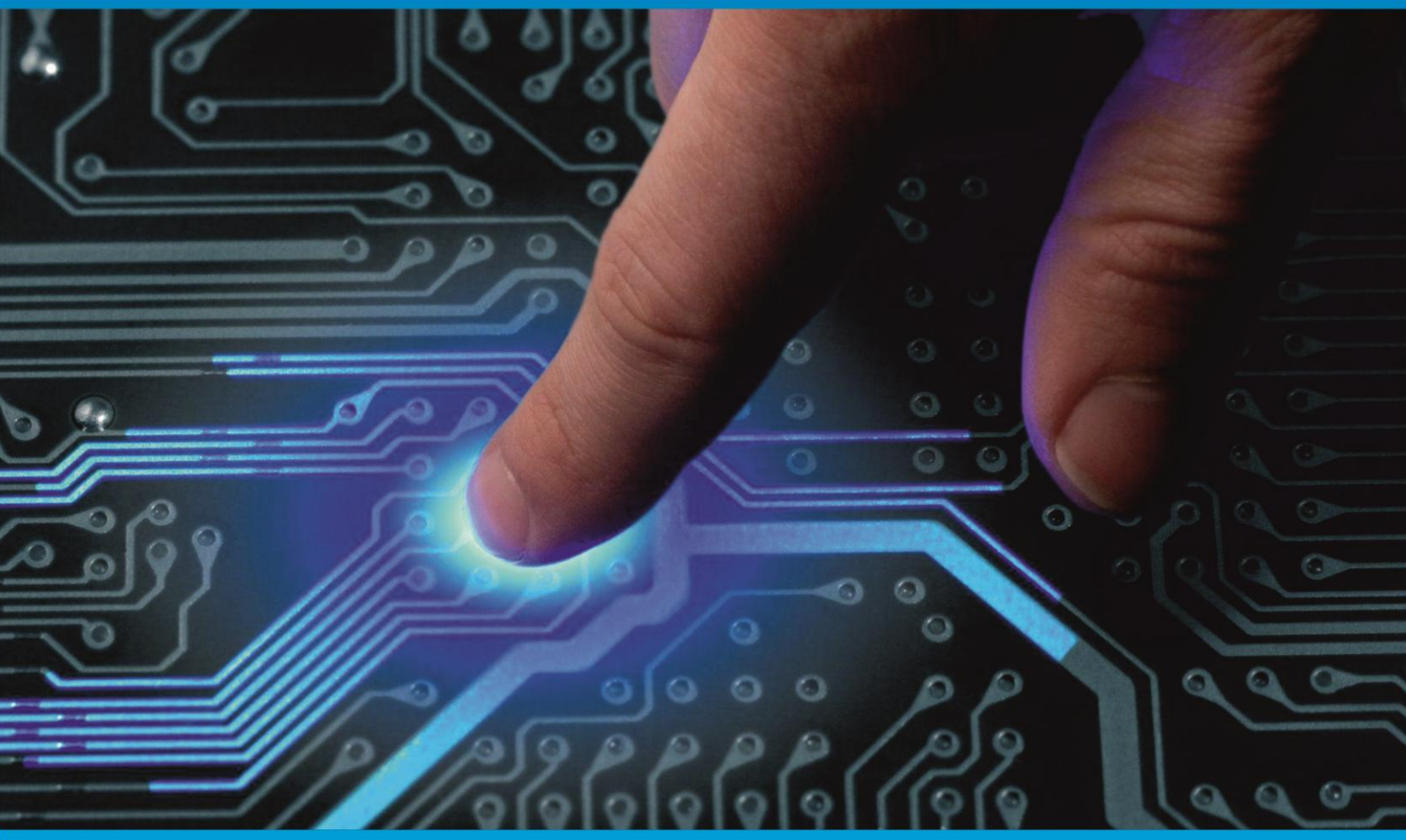
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 4, April 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.488

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

Visual Authentication Protocol to resist Keylogging in Online Transaction

Mrs.E.Benitha Sowmiya[#1]M.E., Mrs.B.Narmada[#2] M.E.,Mr.S.Aravindh[#3] M.Tech., MBA.

Assistant Professor, Department of CSE, Bharath Institute of Higher Education and Research, Chennai ,Tamilnadu,India [#1]

Assistant Professor, Department of CSE, Bharath Institute of Higher Education and Research, Chennai ,Tamilnadu, India[#2]

Assistant Professor, Department of CSE, Bharath Institute of Higher Education and Research, Chennai ,Tamilnadu, India[#3]

ABSTRACT: Key logging or keyboard capturing is the action of recording (or logging) the keys struck on a keyboard,So that the person using the keyboard is unaware that their actions are being monitored. It also has very legitimate uses in studies of human-computer interaction. There are numerous key logging methods, ranging from hardware and software-based approaches to acoustic analysis. Involving human in authentication protocols is not easy because of their limited capability of computation and memorization. We demonstrate visualization design can enhance not only the security but also the usability of authentication. We propose two visual authentication protocols: one is a one-time-password protocol, and the other is a password-based authentication protocol.Ourapproach for real-world deployment were able to achieve a high level of usability for security requirements.

KEYWORDS: Authentication,Smartphone, Malicious code, Keylogger

I.INTRODUCTION

Threats againstelectronic and financial services can be classified into two major classes: credential stealing and channel breaking attacks. Credentials such as users' identifiers, passwords, and keys can be stolen by an attacker when they are poorly managed. For example, a poorly managed personal computer (PC) infected with a malicious software (malware) is an easy target for credential attackers. On the other hand, channel breaking attacks—which allow for eavesdropping on communication between users and a financial institution—are another form of exploitation. While classical channel breaking attacks can be prevented by the proper usage of a security channel such as IPSec [13] and SSL (secure sockets layer), recent channel breaking attacks are more challenging. Indeed, "keylogging" attacks— or those that utilize session hijacking, phishing and pharming, and visual fraudulence— cannot be addressed by simply enabling encryption.

Chief among this class of attacks are keyloggers. A keylogger is a software designed to capture all of a user's keyboard strokes, and then make use of them to impersonate a user in financial transactions. For example,whenever a user types in her password in a bank's sign- in box, the keylogger intercepts the password. The threat of such keyloggers is pervasive and can be present both in personal computers and public kiosks; there are always cases where it is necessary toperform financial transactions using a public computer,although the biggest concern isthat a user's password is likely to be stolen in these computers. Even worse, keyloggers, often rootkitted, are hard to detect since they will not show up in the task manager process list.

To mitigate the keylogger attack, virtual or onscreen key- boards with random keyboard arrangements are widely used in practice. Both techniques, by rearranging alphabets randomly on the buttons, can frustrate simple keyloggers. Unfortunately, the keylogger, which has control over the entire PC, can easily capture every event and read the video buffer to create a mapping between the clicks and the new alphabet. Another mitigation technique is to use the keyboard hooking preven- tion technique by perturbing the keyboard interrupt vector table. However, this technique is not universal and can interfere with the operating system and native drivers.

Considering that a keylogger sees users' keystrokes, this attack is quite similar to the shoulder-surfing attack. To prevent the shoulder-surfing attack, many graphical password schemes have been introduced in the literature [15], [18]. How- ever, the common theme among many of these schemes is their unusability: they are quite complicated for a person to utilize them. For some users, the usability is as important as the security, so they refuse to change their online transaction experience for higher security. The shoulder-surfing attack, however, is different from keylogging in the sense that it allows an attacker to see not only direct input to the computer but also every behavior a user makes such as touching some parts of screen. To adopt shoulder-surfing resistant schemes for prevention of keylogger is rather excess considering the usability. Notice that while defending against the shoulder- surfing attack is out of the scope of this work, and could be partly done using other techniques from the literature intended for this purpose, the promising future of smart glasses (like Google glasses) makes the attack irrelevant to our protocols if it is to be implemented using them instead of mobile phones.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

It is not enough to depend only on cryptographic tech- niques to prevent attacks which aim to deceive users' visual experience while residing in a PC. Even if all necessary information is securely delivered to a user's computer, the attacker residing on that user's computer can easily observe and alter the information and show valid-looking yet deceiving information. Human user's involvement in the security protocol is sometimes necessary to prevent this type of attacksbut humans are not good at complicated calculations and do not have a sufficient memory to remember cryptographically-strongkeys and signatures. Thus, usability is an important factorindesigning a human-involving protocol.

Our approach to solving the problem is to introduce an inter- mediate device that bridges a human user and a terminal. Then, instead of the user directly invoking the regular authentication protocol, she invokes a more sophisticated but user-friendly protocol via the intermediate helping device. Every interaction between the user and an intermediate helping device is visualized using a Quick Response (QR) code. The goal is to keep user-experience the same as in legacy authentication methods as much as possible, while preventing keylogging attacks. Thus, in our protocols, a user does not need to memorize extra information except a traditional security token such as pass- word or PIN, and unlike the prior literature that defends against should-surfing attacks by requiring complex computations and extensive inputs. More specifically, our approach visualizes the security process of authentication using a smartphone- aided augmented reality. The visual involvement of users in a security protocol boosts both the security of the protocol and is re-assuring to the user because she feels that she plays a role in the process. To securely implement visual security protocols, a smartphone with a camera is used. Instead of executing the entire security protocol on the personal computer, part of security protocol is moved to the smartphone. This visualization of some part of security protocols enhances security greatly and offers protection against hard-to-defend against attacks such as malware and keylogging attack, while not degrading the usability. However, we note that our goal is not securing the authentication process against the shoulder- surfing attacker who can see or compromise simultaneously both devices over the shoulder, but rather to make it hard for the adversary to launch the attack.

Scope and Contributions

In this paper, we demonstrate how visualization can enhance not only security but also usability by proposing two visual authentication protocols: one for password-based authentica- tion, and the other for one-time-password. Through rigorous analysis, we show that our protocols are immune to many of the challenging attacks applicable to other protocols in the literature. Furthermore, using an extensive case study on a prototype of our protocols, we highlight the potential of our protocols in real-world deployment addressing usersshortcomings and limitations. The original contributions of this paper are as follows:

•Two protocols for authentication that utilize visualization systemby means of augmented reality to provide both highthat these protocolssecure under several real-world attacks including keyloggers. Both protocols offer advantages due to visualization both in terms of security and usability.

• Prototype implementations in the form of Android applications which demonstrate the usability of our protocols in real-world deployment settings.

We note that our protocols are generic and can be applied to many contexts of authentication. For example, a plausible scenario of deployment could be when considering the terminal in our system as an ATM (Automated Teller Machine), public PC, among others. Furthermore, our design does not require an explicit channel between the bank and the smartphone, which is desirable in some contexts; the smartphone can be replaced by any device with the needed functionality of capture photos .This property enables us to expand our visual authentication protocols into the service context using smart wearable devices, which will be mentioned insection IV.

II.SYSTEM AND THREAT MODEL

A. System Model

Our system model consists of four different entities (or participants), which are a user, a smartphone, a user's terminal, and a server.

The user is an ordinary human, limited by human's shortcomings, including limited capabilities of per- forming complex computations or remembering sophisticated cryptographic credentials, such as cryptographically strong keys. With a user's terminal such as a desktop computer or a laptop, the user can log in a server of a financial institution (bank) for financial transactions. Also, the user has a smartphone, the third system entity, which is equipped with a camera and stores a public key certificate of the server for digital signature verification. Finally, the server is the last system entity, which belongs to the financial institution and performs back-end operations by interacting with the user (terminal or smartphone) on behalf of the bank.

Assuming a smartphone entity in our system is not a far- fetched assumption, since most cell phones nowadays qualify (in terms of processing and imaging capabilities) to be the device used.In our work we assume that thereisno direct channelbetween the serva smartphone does not use the communication channel— er and the smartphone. Also, security and high usability. We showwe note that in most of the protocols proposed in this paper, areunless otherwise is explicitly stated—so a smartphone canbe replaced by any device with a camera and some properprocessing power such as a digital camera, a portable musicplayer with camera (iPod touch, or mobile gadget with theaforementioned capabilities) or a smart watch/glasses

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

B. Trust and Attacker Models

For the trusted entities in our system, we assume the following: First, we assume that the channel between the server and the user's terminal is secured with an SSL connection, which is in fact a very realistic assumption in most electronic banking systems. Second, we assume that the server is secured by every means and is immune to every attack by the attacker; hence the attacker's concern is not breaking into the server but attacking the user. Finally, with respect to the keylogger attack, we assume that the keylogger always resides on the terminal. As for the attacker model, we assume a malicious attacker with high incentives of breaking the security of the system.The attacker is capable of doing any of the following:

- The attacker has a full control over the terminal. Thus,
  - While residing in a user's terminal, the attacker
  can capture user's credentials such as a password, a private key, and OTP (one time password) token string.
  - The attacker can deceive a user by showing agenuine-looking page that actually transfers money to the attacker's account with the captured credentials that she obtained from the compromised terminal.
  - Or, just after a user successfully gets authenticatedwith a valid credential, the attacker can hijack the authenticated session.
- The attacker is capable of creating a fake server to launch phishing or pharming attacks.

For the smartphone in Protocol 1, we assume that it is always trusted and immune to compromise, which means no malware can be installed on it. Notice that this assumption is in line with other assumptions made on the smartphone's trust- worthiness when used in similar protocols to those presented in this paper .We, however, note that relaxing this assumption still could provide a certain level of security with Protocol 2. Protocol 2 uses two factors (password and the smartphone), and thus, the assumption can be relaxed so that not only the terminal but also smartphone could be com- promised (one of them at a time but "not both together"). The non-simultaneous compromise assumption obviously excludes the shoulder-surfing attacker.

In our protocols, we also assume several cryptographic primitives. For example, in all protocols, we assume that a user has a pair of public/private keys used for message signing and verification. In Protocol 1, we assume that the server has the capability of generating one time pads, used for authentication. In Protocol 2, we assume users have passwords used for their authentication. Notice that these assumptions are not far-fetched as well, since most banking services use such cryptographic credentials. For example, with most banking services, the use of digital certificates issued by the bank is very common. Furthermore, the use of such cryptographic credentials and maintaining them on a smartphone does not require any technical background at the user side, and is suited for wide variety of users. Further details on these credentials and their use are explained along with the specific protocol where they are used in this paper.

C. Linear and Matrix Barcodes

A barcode is an optical machine-readable representation of data, and it is widely used in our daily life since it is attached to all types of products for identification. In a nutshell, barcodes are mainly two types: linear barcodes and matrix (or two dimensional, also known as 2D) barcodes. While linear barcodes—shown in Figure 1(a)—have a limited capacity, which depends on the coding technique used that can range from 10 to 22 characters, 2D barcodes—shown in Figure 1(b) and Figure 1(c)—have higher capacity, which can be more than 7000 characters. For example, the QR code— a widely used 2D barcode—can hold 7,089 numeric, 4,296 alphanumeric, or 2,953 binary characters [4], making it a very good high-capacity candidate for storing plain and encrypted contents alike.

Both linear and matrix barcodes are popular and have been widely used in many industries including, but not limited to, automotive industries, manufacturing of electronic components, and bottling industries, among many others. Thanks to their greater capacity, matrix barcodes are even proactively used for advertisement so that a user who has a smartphone can easily scan them to get some detailed information about advertised products. This model of advertisement—and other venues of using these barcodes in areas that are in touch with users—created the need for barcode's scanners developed specifically for smartphones. Accordingly, this led to the creation of many popular commercial and free barcode scanners that are available for smartphones such as iPhone and Andriodphones alike.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

Fig. 1.Three different barcodes encoding the statement "Virtual reality".(a)is a linear barcode (code 128), and (b) and (c) are matrix barcodes (of the QR code standard). While (b) encodes the plain text, (c) encodes an encrypted version using the AES-256 encryption algorithm in the cipher-block chaining (CBC) mode (note this last code requires a password for decryption).

III.PROPOSED CONCEPT

Our concept is to introduce an intermediate device that bridges a human user and a terminal.Instead of user directly invoking the regular authentication protocol,user invokes user-friendly protocol via the intermediate helping device. Every interaction between the user and an intermediate helping device is visualized using a Quick Response (QR) code.In our protocols, a user does not need to memorize extra information except a security token such as password or PIN.our approach visualizes the security process of authentication using a smartphone-aided augmented reality. The visual involvement of users in a security protocol enhance the security of the protocol and offers protection against attacks such as malware and keyloggingattack.To securely implement visual security protocols, a smartphone with a camera is used. Instead of executing the entire security protocol on the personal computer,part of security protocol is moved to the smartphone.

QR Code

Quick Response Code is the trademark for a type of matrix barcode (or two-dimensional barcode) first designed for the automotive industry.A QR code uses four standardized encoding modes(numeric,alphanumeric,byte/binary) to efficiently store data; Extensions may also be used.A QR code consists of black modules (square dots) arranged in a square grid on a white background, which can be read by an imaging device(such as a camera) and processed using Reed–Solomon error correction until the image can be appropriately interpreted. The required data are then extracted from patterns present in both horizontal and vertical components of the image.

c

One Time Password

The one-time passwords in user authentication protocol oPass are generated by a secure one-way hash function.With a given input, the set of one time passwords is established by a hash chain through multiple hashing. Assuming we wish to prepare N one-time passwords, the first of these passwords is produced by performing N hashes on input **c.**

$$\delta_0 = H^{n}(c) \quad (1)$$

The next one-time password is obtained by performing hashes

$$\delta_1 = H^{n-1}(c) \quad (2)$$

Hence, the general formula is given as follows:

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

$$\delta_i = H^{n-i}(c) \quad (3)$$

Create New Account

User creates a new account in our banking application.user have to input all the valid informations including his mobile IME number in the new account registration.After enters the user details, the form is submit to the corporate.Our bank manager validate and accept the user registration form then creates account number for the user.Now user receives an account number and user can able to access all the services in our bank.

Apply Net Banking

User receives an account number then user have to enters a retail login and applies a net banking as services.This service enables the account holder to view his profile and account details.Now user can able to transfer the fund to another account.ourbank provides secure net banking because fund transfer /money transfer is challenging task for the user in untrusted PCs devices.

An Authentication Protocol with Password and Randomized Onscreen Keyboard

Before enters the password,Account holder have to enter the account number and amount going to transfer.Our banking application uses a two visual authentication protocols (OTP) One-Time-Password and Password-based authentication.These two protocols are used to transfer the fund/amount to another account.

Protocol 1-One-Time-Password Protocol(OTP): Each and every time the randomized OTP 0-9 number is generated.It is then encrypted in 4x4 grid in the form of QR code.The QR code is display in the left-hand side of the terminal and randomized 4x4 grid plain keyboard is display in the right-hand side.Account holder using his android phone scans the QR code.If the IME registered during account registration and the user's mobile IME are matched then the QR code is decrypted using user's private key,then the OTP is appear on the user's mobile.The OTP contains randomized (0 to 9) number placed in different places in the 4x4 grid.

Protocol 2-Password-based authentication Protocol: Using android mobile user views a randomized (0 to 9) number placed in the 4x4 grid.Then user clicks his transaction password in the right hand side randomized plain keyboard using the mouse with the help of OTP viewed in his mobile.If the password is match then the fund/amount is transferred to another account.

IV. PROPOSED ALGORITHM

Visual Authentication Protocols To Resist Keylogging

In this section, we describe two protocols for user authen- tication with visualization. Before getting into the details of these protocols, we review the notations for algorithms used in our protocols as building blocks. Our system utilizes the following algorithm

Encr$_k$(.): an encryption algorithm which takes a key *k* and a message*m*from set M and outputs a ciphertext*c*in the set C.

Decr$_k$(.): a decryption algorithm which takes a ciphertext

c in C and a key *k*, and outputs a plaintext (or message)

m in the set M.

Sign(.): a signature generation algorithm which takes a private key *SK* and a message *m* from the set M, and outputs a signature σ.

Verf(.): a signature verification algorithm which takes a public key *PK* and a signed message (M, σ), and returns valid or invalid.

QREnc(.): a QR encoding algorithm which takes a string *s* in S and outputs a QR code.

QRDec(.): a QR decoding algorithm which takes a QR code and returns a string*s* in S.

Any public key encryption scheme with IND-CCA2 (Indistinguishability against Adaptive Chosen Ciphertext Attacker) security would be good for our application. A public key encryption scheme with IND-CCA2 adds random padding to a plaintext, which makes the ciphertext different whenever encrypted, even though the plaintext is the same. This re- striction on the type of the used public key encryption scheme will prevent an attacker from checking whether his guess for the random layout is right or not. Thus, the security of the scheme is not dependent on the number of possible layouts but the used encryption scheme. If no such encryption is used, the adversary will be able to figure out the layouts used because he will be able to verify a brute-force attack by matching all possible plaintexts to the corresponding ciphertext. On the other hand, when such encryption is used, the 1-1 mapping of plaintext to cipher text does not hold anymore and launching the attack will not be possible at the first place. Also, any signature scheme with EUF-CMA (existential-unforgeabilityagainst adaptive chosen-message attacker) can be used to serve the purpose of our system.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

A. Authentication With Random Strings

In this section, we introduce an authentication protocol with aone time password (OTP). The following protocol (referred to as Protocol 1 in the the rest of the paper) relies on a strong assumption; it makes use of a random string for authentication.
The protocol works as follows:

1) The user connects to the server and sends her ID.

2) The server checks the ID to retrieve the user's publickey (PK_{ID}) from the database. The server then picks a fresh random string OTP and encrypts it with the public key to obtain E_{OTP} = Encr$_{PK\text{ID}}$ (OTP).

3) In the terminal, a QR code QR_EOTP is displayed prompting the user to type in the string.

4) The user decodes the QR code with E_{OTP} = QRDec(QR_EOTP).Because the random string is en-crypted with user's public key (PK_{ID}), the user can read the OTP string only through her smartphone by OTP = Decr$_k$(E_{OTP}) and type in the OTPin the terminal with a physical keyboard.

5) The server checks the result and if it matches what the server has sent earlier, the user is authenticated. Otherwise, the user is denied.

In this protocol, OTP is any combination of alphabets or numbers whose length is 4 or more depending on the security level required.

B. An Authentication Protocol with Password and Randomized Onscreen Keyboard

Our second protocol, which is referred to as Protocol 2 in the rest of this paper, uses a password shared between the server and the user, and a randomized keyboard.The protocol works as follows:

1) The user connects to the server and sends her ID.

2) The server checks the received ID to retrieve the user'spublic key (PK_{ID}) from the database. The server preparesπ, a random permutation of a keyboard arrangement, and encrypts it with the public key to obtain E_{KBD} = Encr$_{PK\text{ID}}$ (π). Then, it encodes the ciphertext with QR encoder to obtain QR_EKBD = QREnc($E_{k_{\text{ID}}}$ (π)). The server sends the result with a blank keyboard.

3) In the user's terminal, a QR code (QR_EKBD) is displayed together with a blank keyboard. Because the onscreen keyboard does not have any alphabet on it, the user cannot input her password. Now, the user executes her smartphone application which first decodes the QR code by applying QRDec(QR_EKBD) to get the ciphertext (E_{KBD}). The ciphertext is then decrypted by the smartphone application with the private key of the user to display the result (π= Decr$_{SKID}$ (E_{KBD})) on the smartphone's screen.

4) When the user sees the blank keyboard with the QR code through an application on the smartphone that has a private key, alphanumerics appear on the blank keyboard and the user can click the proper button for the password. The user types in her password on the terminal's screen while seeing the keyboard layout through the smartphone. The terminal does not know what the password is but only knows which buttons are clicked. Identities of the buttons clicked by the user are sent to the server by the terminal.

5) The server checks whether the password is correct or not by confirming if the correct buttons have been clicked.

V.ILLUSTRATION

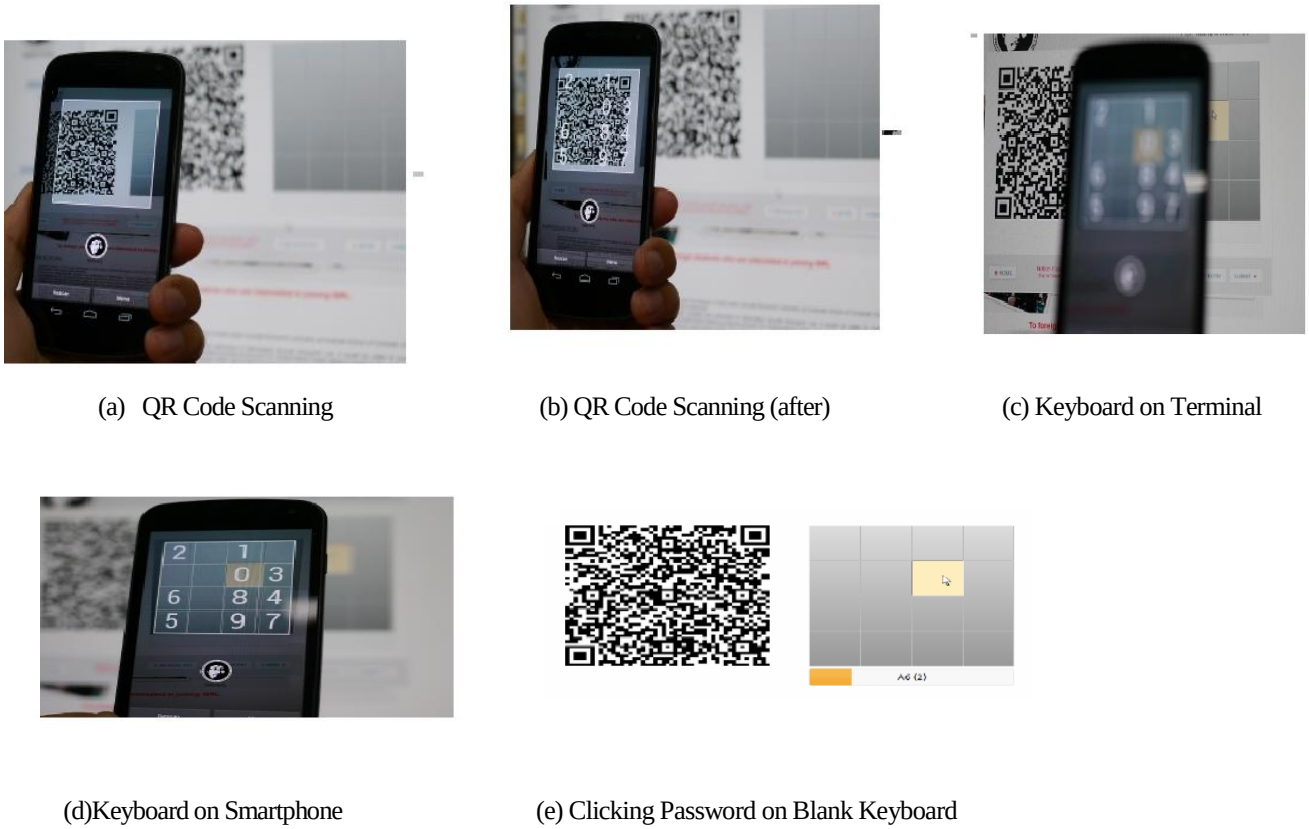
Some of the technical issues in the two protocols that we have introduced in the previous sections call for further discussion and clarification. In this section, we elaborate on how to handle several issues related to our protocols, such as session hijacking, transaction verification, and securing transactions.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |



(a) QR Code Scanning (b) QR Code Scanning (after) (c) Keyboard on Terminal

(d)Keyboard on Smartphone (e) Clicking Password on Blank Keyboard

Fig.1. Photographs of the prototype we have developed to demonstrate our authentication protocols. (a) and (b) show the moments of a QR code scanning of a keyboard layout. (c) shows the blank keyboard shown at the terminal (on LCD screen). (d) shows the decoded randomized layout of the keyboard obtained from the QR code after decryption as viewed on smartphone. Note that the yellow square on which the mouse cursor is hovering in the terminal is shown through the smartphone to assist user's input. (e) shows that a user is clicking the password on the blank keyboard while seeing numbers through the smartphone.

High-level description of an authentication protocol with password and a randomized onscreen blank keyboard.

user::user.send(server, id)

server::__upon_id_arrival:

if(**server**.verify(id) == true):

pkid =**server**.db.find(id)

pi =**server**.generate_random_kb()

ekbd = server.encrypt(pkid, pi)

qrekbd = server.qrencode(ekbd)

server.send(user, qrekbd)

terminal::__upon_qrekbd_arrival:

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

```
    terminal.view(qrekbd)
    terminal.view_blank_kb(pi)
smartphone::__upon_qrekbd_view:
    qrekbd = smartphone.capture(qrekbd)
    ekbd = smartphone.qrdecode(qrekbd)
    pi = smartphone.decrypt(skid, ekbd)
    smartphone.view(pi)
user::__upon_pi_view:
    pw = user.inputpassword(terminal)
terminal::upon_pw_input:
    terminal.send(server, pw)
server::__upon_pw_arrival:
    if(server.verify(id, pw) == true):
        server.authenticate(user)
    else:
        server.deny(user)
```

A. Password Hashing

Passwords are usually stored in a hashed form with a salt to prevent server attacks, instead of being stored in plaintext on the server. In Protocol 2, we can easily support this password hashing by making the server compare the password hash computed from the stored salt value and the transferred password after decrypting it with the stored password hash value.

B. Message Signing

For the generality of the purpose of this protocol and the following protocols, and to prevent the terminal from misrepresenting the contents generated by the server, one can establish the authenticity of the server and the contents generated by it by adding the following verification process. When the server sends the random permutation to the user, it signs the permutation using the server's private key and the resulting signature is encoded in a QR code. Before decrypting the contents, the user establishes the authenticity of the contents verifying the signature against the server's public key. Both steps are performed using the Sign and Verfalgorithms. Verification is performed by the smartphone to avoid any man-in-the-middle attack by the terminal.

C. Prevention of Session Hijacking with Visual Signature Validation

Even with secure authentication, an attacker controlling entities in the system—the terminal in particular—via a malware can hijack the authentication session when a user tries to request some transactions such as money transfer. Though usually money transfer action prompts a user to input the password, the malware can easily hijack it and alter the transfer information with the attacker's information.To prevent the session hijacking in Protocol 1, we can make a QR code to include additional information on the user's transaction request as follows:

1) A user requests via terminal to the server money transfer denoted as T that describes sender name/account,recipient name/account, a timestamp, and amount of money to transfer.

2) The server checks the ID to retrieve the user's public key (PK_{ID}) from the database. Then, it picks a fresh OTP to prepare $QR = \text{QREnc}(E_{OTP}, T, \sigma = \text{Sign}(PrK, T))$, where PrK is a signing key of the server. Then, it sends QR to the user to authorize thetransaction.

3) On the terminal, a QR code QR is displayed prompting the user to type in the OTP string.

4) The user decodes the QR code to get $(E_{OTP} = \text{QRDec}(QR_{EOTP}), T, \sigma)$ with her smartphone application. Here the application verifies the time stamp and the signature by $\text{Verf}(PubK, T, \sigma)$ to show the result (Valid/Invalid) on the screen with the decrypted OTP and T. If the application fails to validate the signature, it does not show neither the decrypted OTPnor$T$, but displays an error message to

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

alert the user. When the user is confirmed with the signature verification result and with*T* , she inputs the *OT P* to the terminal, which is sent back to the server.

5) The server checks the result and if it matches with the *OT P* that the server has sent earlier, the user is authenticated. Otherwise, the user is denied.

This expansion of the protocol enables a user to confirm that her critical transaction request has not been altered, and thus, the session hijacking attack is prevented. For this additional security functionality, a user's involvement is minimized in the protocol, because a user only sees the transaction information on the phone when it is valid, or an alert message when invalid. We note that the expansion is also applicable to Protocol 2, but not applicable to legacy OTP/password authentications.

D.Backward Visual Channel from PC to Smartphone

It is quite natural to think of the backward channel from PC to smartphone when PC has a camera. The idea of using QR code on the phone's screen as an upload channel (from phone to terminal) can be used, as previously done in other schemes. Instead of using cellular network for a phone to send the confirmation, we can use this QR code-based visual channel from phone to a terminal.

The use of the visual channel to input encrypted credentials from the smartphone to the terminal has an interesting security implication. As is the case with using e-banking on untrusted terminals, imagine that such terminal is infected with a virus, or has a malware, which could be a keylogger. If the user is to use the authentication credentials directly on the terminal, it is obvious that these credentials will be compromised. On the other hand, if these credentials are keyed in on the smartphone and to be transferred using
the visual channel between the smartphone and the terminal in an encrypted form for theserver, the keylogging attacker will be prevented from logging these credentials on the terminal.

E. The Smart Glasses

The application context of our visual authentication protocols can be easily expanded by applying the visual authenti- cation protocols to the smartphone and the glasses instead of the terminal and the smartphone. That is, an encrypted OTP for Protocol 1 or a blank keyboard for Protocol 2 appears on a smartphone instead of a terminal, and a user sees the secret information through the glasses instead of a smartphone to authenticate securely by entering the OTP or by tapping the password on the blank keyboard on the smartphone. Accordingly, the threat model must be redefined, and the usability study must be conducted again. The application- context change using the newly-arisen smart wearable devices such as smart glasses and smart watches together with a smartphone may lead to new security services supported by the visualization concept. We leave this for future work.

F. Time-based OTP Devices

While one may consider independent OTP device such as RSA's SecureID and Google's Authenticator [2], [1], the way both systems work prevent various advantages of our design if they are used in conjunction. Both of the SecurIDand Google Authenticator are variations of the same idea, and they indeed implement the broader type of time-based authentication using RFC "6238: TOTP: Time-Based One- Time Password Algorithm". SecurID uses a tokenizer(at the client side; hardware or software) which has its own clock and a symmetric key, used as the seed for computing the token. At the other side, the server has a database with all legit "smart-cards" (tokenizers), a real clock, and keys. The server uses the key and the real clock value to re-compute the token generated by the user and validate the PIN sent by the user with a 60 seconds time-window. While SecurID is a proprietary software (closed source), Google authenticator is an open source, and software-based. It implements the token as the first few digits (6) of the HMAC-SHA1 of the current clock value on the device using the app. The same thing is done for verification as in SecurID. SecurIDtokenizer refreshes tokens every 60 seconds and Google authenticator uses 30 seconds as a default.

While we use the concept of OTP, our use of tokens is in the form of challenge/response, differing from the time- based authentication schemes (represented by the two schemes above). Unlike both schemes, the OTP in our case is generated by the server, not the user, and is not timed (although the server may reject the once after a time-out by keep a counter of that OTP). For that reason, our system brings two advantages: (1) our system provides better mitigations to the replay attack. Everytime the user connects to the server, she's given a fresh token, whereas tokens in the time-based systems are refreshed every certain number of seconds. That window can be used by the attacker for launching a replay attack. (2) While mitigating the replay attack, our use of the challenge/response instead of time-based OTP is user friendly. It removes any stress on the users of having to type in the token within a short period of time.

G. Replacing Visual Channels with Bluetooth

The visual channel in Protocol 1 (that uses OTP) is used to transfer the encrypted OTP from PC to the smartphone, and the user plays a role of another channel from the smartphone to PC by entering the decrypted OTP into PC. Here, both channels (from PC to the smartphone and vice versa) can be replaced with other channels such as Bluetooth, and the whole authen- tication procedure can be automated. This will significantly enhance the usability of the authentication protocol. However, in another aspect, not all PCs are equipped with the Bluetooth module. Also, even though PC has the Bluetooth module, it might be an annoying job to execute the pairing whenever the user uses a device that she has never paired before. In that sense, Protocol 1 with visual channel and user's entering PIN are easier to be deployed in the current

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

environment.

VI. IMPLEMENTATION

In this section, we describe the details of the prototype implementations, and show the results of the user study for Protocol 2 using a numeric keyboard and using an alphanu- meric keyboard. The numeric keyboard study was to know the speed and the error of the PIN entry, and the alphanumeric keyboard study was for the password entry.

A.Numeric Keyboard with Blank Space

We implemented Protocol 2 to see its usability for PIN, which is widely accepted for authenticating a person during banking transactions. QR code in the protocol, which includes 164 characters at low error correction level, contains a JSON object [11] that consists of an encrypted keyboard layout, hashed user ID, and current time. The keyboard layout is encrypted by using AES-128 encryption algorithm with CBC mode and PKCS#7 padding. Base64 was used for encoding the ciphertextand initial vector. We used ZBar android SDK 0.2 [8], an open source library for reading barcodes and QR codes. To speed up the reading QR codes, before running ZBar library, we let the application re-sample a capture image to a small image of which width is 500-pixel using nearest neighbor image scaling. After reading QR code, the smartphone displays a numeric keyboard on screen. The size of numeric keyboard is 4□4 and the numeric keyboard contains 10 numbers (0 to 9) in random positions. The rest of the positions remain in blank.

B. Alphanumeric Keyboard

Alphanumeric passwords are widely used to sign into various types of servers, so we also developed a prototype of Protocol 2 with an alphanumeric keyboard as an Android application to see its usability. The application can run on any smartphone with Android OS

[17] (version 2.2 or later). AES-192 encryption algorithm (in the counter mode) for contents encryption, Base64 encoding for byte-to-character encoding of encrypted contents, and uses ZXing [5], an open source implementation provided by Google for reading several standards of the 1D and 2D barcodes**.**

VII. USABILITY &DEPLOYABILITY

Besides the security of an authentication protocol, both usability and deployability are equally important and critical for the acceptance of any protocol in modern computing settings. Bonneau et al. have developed 25 different metrics for evaluating such aspects in an authentication scheme to compete with the existing password-based authentication that is well-accepted in practice [7]. Furthermore, while those metrics are ideal, and the best authentication scheme in the literature does not address many of them, they are fairly generic to benchmark different designs and to compare them based on their merits. For that, the authors provided an extensive comparison and study of 38 schemes based on those metrics. Here, we benefit from this study in understanding our protocols in the context of the related works. We outline some of the merits based on the common features our schemes share with other works, and some others based on the prior user studies and security analyses we discussed in this paper.

The reader is referred to [7] for further details on the definitions those metrics, and how they apply to the various authentication mechanisms in the literature.In the following, we summarize how our proto- cols perform on those metrics, and thus how they compare to other protocols in the literature. We limit our attention to the baseline, the password-based authentication, and a few phone- based authentication protocols as shown in Table V. We notice that our first protocol meets the first metric by not requiring a password, meets the fifth, sixth, seventh, and eighth as shown in our user studies. Our design is security-rich, and its security features are discussed earlier to support the marked merits. Finally, for deployability, our system relies on an intensive user study that provides an obvious merit of its use against those metrics. The mapping of the metrics for the second protocols are also concluded from the prior discussion—details are omitted for the lack of space.

For the coloring part in the comparison, we use the same coloring system - with slightly different legend and keys - as utilized by Bonneau in his work, and the rationale of using those colors is explained therein. Our choice of PhoolProofandCronto is not arbitrary: They share the same platform and some of the design aspects with our work, and thus it was easy to extrapolate many of the (subjective) coloring made by Bonneau to our work. This particularly explains much of the matching in deployability and security colorings for ourprotocol. The mismatch in "resilience to leaks from other verifiers" does not apply to our work, since it assumes a single verifier. The first protocol is more memorywise efficient, efficient to use, and infrequent errors than password based on the fact that it relies on OTP (not memorized by the user.

VIII .SECURITY ANALYSIS

Trust in our protocols can be seen shifted from PC to smartphone to make authentication protocols secure against malware in a PC. However, considering that it is not easy to protect user's credentials when a malware resides in a PC without sacrificing usability [1], and that a user sometimes has to use an untrusted PC such as a public PC or a kiosk, our approach to move trust to the smartphone that is at least more

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

trustworthy than public PCs is plausible. Also, in Protocol 2 that uses a password for authentication, a smartphone is not required to be trusted because a password is another factor for any successful authentication.

In this section, we analyze the security of our scheme under several attack scenarios and show how these attacks are defended against.

A. Key Space and Brute-Force Attacker

In our protocols, several stages include encryption of sensitive information such as credentials, which are of interest to the attacker (including the user ID, password, and nonce generated by the server). In our prototype, and system recommendations for wide use of our protocols, as well as the description provided above for the different protocols, we consider public key cryptography. Furthermore, we suggest a key length that provides good security guarantees. This includes the use of RSA-2048, which is infeasible to attack using the most efficient brute-force attack. This applies to both encryption and signature algorithms used in the protocols. Notice that all public key cryptography in our protocols (except for signing and verification) can be replaced by sym- metric key cryptography, which is far more efficient (despite that computation overhead in our protocols is marginal). Fur- thermore, such replacement of cryptographic techniques will
It is very hard to detect rootkits with software when they reside in the OS's kernel, so nowadays dedicated hardware devices are being considered to detect rootkits. Also, looking back at research on shoulder-surfing prevention, the shoulder-surfer resisting techniques are necessarily accompanied by degradation of usability somehow.

B. Keyloggers

Keyloggers are popular and widely reported in many contexts. In our protocols, input is expected by the user, and in every protocol one or another type of input is required. Our protocols—while designed with the limitations and shortcoming of users in mind, and aim at easing the authentication process by means of visualization— are aimed explicitly at defending against the keylogger attacks. Here, we further elaborate on the potential of using keyloggersas an attack, and the way they impact each of the two protocols.

Protocol 1. Authentication in this protocol is solely based on a random string generated by the server. The random string is encrypted by the public key of the user, and verified against her private key. The main objective of using OTP is that it is for one time use. Accordingly, if the keylogger is installed on the terminal, the attacker obviously will be able to know the OTP but will not be able to reuse it for future authentication. Alternatively, a keylogger installed on the smartphone will not be able to log any credentials, since no credentials are input on the smartphone. It is worth noting that the attacker may try to block users from being authenticated and reuse the OTP immediately. In this case, mitigations explained in section V-C can be used to remedy the (session hijacking) attack.

Protocol 2. In the second protocol, a blank keyboard is posted on the terminal whereas a randomized keyboard with the alphanumerics on it is posted on the smartphone. Because the protocol does not require the user to do any keyboard input on the smartphone side, the protocol is immune against the keylogger attack. The user just checks the keyboard layout on the phone and there is no input from a user. Obviously, the terminal might be compromised, but the keylogger will be able to only capture what keystrokes are used on the blank keyboard. Thus, the keylogger will not be able to know which alphanumeric characters are being clicked.

C. Malicious Software (malware)

The term malware is generic, and is technically used to describe any type of code with malicious intentions including keyloggers. It is obvious that an attacker who successfully compromised a smartphone that has a private key that is a whole credential required to break the system in Protocol 1 will be always successful to break the systems except Protocol 2 that requires both password and private key.

D. Theft of Smartphone

In case the smartphone that has a key to recover OTP in Protocol 1 and to decipher the keyboard layout in Protocol 2 is stolen, obviously the security degrades. In Protocol 1, theft of smartphone means that the attacker has total control over user's account if the attacker knows the user's ID. Protocol 1 can be regarded as an authentication protocol requiring only one security token (a smaprtphone) and focusing on user .

Finally, while our design is not intended in its current form to defend against it, we note that the smart glasses such as the Google glass will easily frustrate the shoulder surfing attacker. In essence, this is because the keyboard layout will be shown only to the user wearing the glasses.

E. Shoulder-Surfing Attacks

As already mentioned in the introduction, shoulder-surfing resistance is not within our scope. However, in this section, we investigate the possibility and the effectiveness of shoulder- surfing attacks. The shoulder surfing is a powerful attack in the context of password-based authentication and human identification. In this attack, the attacker tries to know credentials, such as passwords or PINs (personal identification numbers) by stealthily looking over the shoulder of a user inputting these credentials into the systems.

In Protocol 1, OTP tokens that have high entropy and are human-unfriendly making them hard to remember and recall are one-time used. Accordingly, a shoulder surfer would not benefit from launching an attack by trying to observe what the user at the terminal is inputting. The attack is not applicable to this protocol.

In Protocol 2, observing the terminal or the smartphone keyboard layout (on the smartphone screen) alone would not reveal the credentials of the user. Observing both at the same time in a shoulder surfing attack, and mapping stroked keys on the terminal to those on the smartphone screen would reveal the credentials of the user. Being able to successfully launch this attack is a non-trivial task, and requires the attacker to be in very near proximity to the user, which would raise the user's suspicions about the intentions of the attacker. However, because the attacker who successfully conducts all this necessary steps will get a password in Protocol 2, the protocol cannot be

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 4, April 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0904132 |

said to be secure against the shoulder- surfing attack.

F. Comparison

To sum up, we compare the two protocols and the way they perform against several attacks. We consider the scenarios where the attacker has control over either the terminal or smartphone but not both of them at the same time.

IX .CONCLUSION& FUTURE DIRECTIONS

In this paper, we proposed and analyzed the use of userdriven visualization to improve security and user-friendliness of authentication protocols. Moreover, we have shown two realizations of protocols that not only improve the user experi- ence but also resist challenging attacks, such as the keylogger and malware attacks. Our protocols utilize simple technologies available in most out-of-the-box smartphone devices. We de- veloped Android application of a prototype of our protocol and demonstrate its feasibility and potential in real-world deployment and operational settings for user authentication. Our work indeed opens the door for several other directions that we would like to investigate as a future work. First of all, our plan is to implement our protocol on the smart glasses such as the google glass, and conduct the user study.

REFERENCES

[1] —.Google authenticator.http://code.google.com/p/ Proc. of ESORICS, pages 1-18,2009.
[2] —. Rsasecurid. http://www.emc.com/security/rsa-securid.htm.
[3] Cronto. http://www.cronto.com/.
[4] —. BS ISO/IEC 18004:2006. information technology. automaticiden-tification and data capture techniques. ISO/IEC, 2006.
[5] —. ZXing. http://code.google.com/p/zxing/, 2011.
[6] D. Boneh and X. Boyen. Short signatures without random oracles. In Proc. of EUROCRYPT, pages 56-73, 2004.
[7] 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 Security and Privacy (SP), 2012 IEEE Symposium on, pages 553-567. IEEE, 2012.
[8] J. Brown. Zbar bar code reader, zbar android sdk 0.2. http://zbar. sourceforge.net/, April 2012.
[9] C.-H. O. Chen, C.-W. Chen, C. Kuo, Y.-H.Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y.Yang, and T.-C.Wu. Gangs: gather, authenticate 'n group securely. In J. J. Garcia-Luna-Aceves, R. Sivakumar, and P. Steenkiste, editors, MOBICOM, pages 92-103. ACM, 2008.
[10] S. Chiasson, P. van Oorschot, and R. Biddle. Graphical password authentication using cued click points. In Proc. of ESORICS, 2008.
[11] D. Crockford. The application/json media type for javascript.http://www.ietf.org/rfc/rfc4627.txt?number=4627, July 2006.
[12] D. Davis, F. Monrose, and M. Reiter. On user choice in graphical password schemes.In Proc. of USENIX Security, 2004.
[13] N. Doraswamy and D. Harkins.IPSec: the new security standard for the Internet, intranets, and virtual private networks. Prentice Hall, 2003.
[14] M. Farb, M. Burman, G. Chandok, J. McCune, and A. Perrig. Safes- linger: An easy-to-use and secure approach for human trust establish- ment. Technical report, CMU, 2011.
[15] H. Gao, X. Guo, X. Chen, L. Wang, and X. Liu. Yagp: Yet another graphical password strategy. In Proc. of ACM ACSAC, pages 121-129, 2008.
[16] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal, 1988.
[17]Google.Android.http://www.android.com,
[18] E. Hayashi, R. Dhamija, N. Christin, and A. Perrig. Use your illusion: secure authentication usable anywhere. In Proc. of ACM SOUPS, 2008.

INNO SPACE

SJIF Scientific Journal Impact Factor

Impact Factor:
7.488

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details