Entity Authentication in a Mobile-Cloud Environment

David Schwab, Li Yang
University of Tennessee at Chattanooga
615 McCallie Avenue
Chattanooga, TN 37403
zys887@mocs.utc.edu

ABSTRACT
Mobile devices are seeing an increase in usage in recent years and mobile security becomes important in part due to the shift of computing landscape towards mobile devices. Security and assurance of mobile computing is vital to the normal functioning in people's lives and our social, economic and political systems. In this paper, we propose and implement a novel system that authenticates users, devices, and a remote server in a mobile computing environment based on fuzzy vault, digital signature and zero-knowledge authentication. Our protocol is robust against the following attacks: 1) sniffing attack; 2) man-in-the-middle; 3) data modification; 4) impersonation; and 5) loss of device. Additionally, our protocol provides usability by using a fuzzy picture password. The strength of our protocol security is enhanced by using sensor data from the mobile device in the process of key generation. Our protocol was implemented and evaluated using Android and Amazon EC2.

Categories and Subject Descriptors
K.6.5 [**Management of Computing and Information Systems**]: Security and Protection – *authentication, unauthorized access.*

General Terms
Security, Human Factors

Keywords
Mobile Security, Picture Authentication, Fuzzy Password

1. INTRODUCTION
The computing landscape is shifting rapidly towards mobile platforms, and PCs are no longer the dominant form of computing. Worldwide unit sales of mobile devices are expected to increases from 300 million in 2010 to 650 million in 2012 [15]. Mobile applications are growing explosively as evidenced by over 425,000 apps available for iOS devices and over 200,000 apps in Android Market [13]. In June 2011, for the first time ever mobile-device users on average spent more time using mobile applications (81 minutes) than browsing the web (74 minutes) [14]. Indeed, mobile devices are becoming general-purpose computing platforms, and often store tremendous amounts of personal, financial, and commercial data. As such, mobile devices attract both targeted and mass-scale attacks. Recently over 250,000 Android users were compromised in an unprecedented mobile attack when they downloaded malicious software disguised as legitimate applications from the Android Market [16]. With the unstoppable "Bring Your Own Device" (BYOD) movement in enterprises, and the growing amount of data that consumers and business are now generating, the need to secure data is more urgent than ever.

Authentication is the key of security services including confidentiality and integrity. Mobile devices suffer from a variety of security threats such as sniffing of wireless communication, impersonation, loss of devices, and man-in-the-middle attacks. This paper proposes a novel protocol which is robust against the above threats by authenticating both user and device identities before communication, which creates a secure channel between mobile devices and remote cloud. Our protocol also adopts a visual password which provides usability in smart mobile devices with a touch screen. The visual password system proposed for this system has the property of allowing the user to specify part of the correct sequence of images to prove his or her identity. The user of the device will only be allowed access to the cloud resources after the user, device and server in the cloud are authenticated.

CSIIRW '12, Oct 30 - Nov 01 2012, Oak Ridge, TN, USA
ACM 978-1-4503-1687-3/12/10.

2. BACKGROUND
We introduce several key techniques that are integrated in our novel protocol, which are fuzzy vault, picture authentication, and zero-knowledge authentication.

2.1 Fuzzy Vault
A fuzzy vault is a cryptographic construct that locks a secret k using a key A, and can unlock the secret using any key B that is substantially similar to A [5]. To lock a value (k) in the vault, the secret k is represented as a polynomial p, where the information of the secret is encoded as coefficients of p. Next, a key A = [a_1, a_2, …] is chosen, which is an array of values. The polynomial p is then evaluated at the values in A, creating a a new set R, where R = [(a_1, $p(a_1)$), (a_2 , $p(a_2)$), …]. At this time extra points are chosen to be evaluated over the polynomial p, and these extra points are added to the set R, which serves to hide the value of A. It is also a good idea to order the values in R such that the order in which they were chosen is not obvious.

To unlock the secret stored in the fuzzy vault, a user must submit a set of values, B, that is significantly similar to the values in A used to lock the vault. Initially, the algorithm iterates over the values in B, searching for matches in the x-values of R. As matches are found, each matching 2-tuple from R is added to a new set, Q. At this stage the set Q, and the degree of p are used as input for a special Reed-Solomon decoder. The decoder either

outputs the original polynomial p if the unlock procedure was successful, or else Null [5].

2.2 Picture Authentication

There are several areas of research that are being presented that explore the use of pictures and images to authenticate users or encrypt information [1, 4, 6, 8]. One approach is to replace textual passwords with image selection in order to authenticate a user [1, 4]. Textual passwords that are random enough and of sufficient length to protect against a brute-force attack are often difficult to remember, and are prone to entry errors. Typically text based passwords need to be changed after a period of time such that an attacker does not have an unlimited amount of time to defeat the password. It is suggested in [4] that picture passwords could use the same set of images for a password, and simply generate a new value to be associated with the picture password. Applying this method could be done automatically without the need for user intervention, and thus providing additional security. In [6, 8] a different use of visual cryptography is presented. Here, the authors propose that an image can be split into n separate transparencies. A user can only obtain the original image if he has $k < n$ of the transparencies. An additional feature of this method is that having k-1 transparencies gives a user no information about the other transparencies, or about the hidden image.

2.3 Zero-Knowledge Authentication

The area of zero-knowledge authentication is an effort to create a system where an entity can prove it knows a secret, without revealing the secret, or any information about the secret that would allow an observer to gain knowledge about what the secret is. One such protocol is the Feige-Fiat-Shamir protocol which is outlined in [2, 10]. In this protocol, two large prime numbers p and q are multiplied to obtain $n = p \times q$. The value n is a public value. In this protocol there is a vector $S = [s_1, s_2, \ldots, s_k]$ of private keys, a vector $V = [v_1, v_2, \ldots, v_k]$ of public keys, and a boolean challenge vector $C = [c_1, c_2, \ldots, c_k]$. The values in S are randomly chosen such that each s_i is relatively prime to n. The public keys are created where $v_i = (s_i^2)^{-1}$. In order for the claimant to prove knowledge of the secret key, a random number r is chosen, and $x = r^2 \bmod n$ is sent to the verifier. The verifier then responds by sending a challenge C to the claimant, where each c_i is a 0 or 1. The claimant responds by calculating and sending $y = (r s_1^{c1} s_2^{c2} \ldots s_k^{ck}) \bmod n$ to the verifier. Finally, the verifier calculates $(y^2 v_1^{c1} v_2^{c2} \ldots v_k^{ck}) \bmod n$ and compares this value with the x received previously. If the two values are equal, then it is probable that the claimant knows the secret. Otherwise, it is improbable that the claimant knows the secret. The probability for knowing the secret based on the calculations above depends on the value of k chosen.

3. SECURITY NEEDS

For secure communications over the internet it is important that entities can authenticate the identity of each other in a secure manner. When an individual uses their mobile device to log on to an online banking website, it is critical that the identity of the website can be established. Likewise, if a private corporation uses mobile devices to access secure resources on their network, it is imperative that both the identity of the mobile device and the company servers are authenticated. Often times entity authentication is accomplished through the use of certificates and digital signatures. Using these methodologies the identity of the mobile device and the server can be verified. However, because of the risk of mobile device loss or theft, it is also imperative that the mobile device's user is also authenticated.

Typically the identity of users over a network is done using a user name and password pair that can be verified by the server. It would not make much sense to allow the mobile device to authenticate the user because we assume that the mobile device could be in the hands of a malicious user, and therefore subject to tampering. The problem with normal passwords is that they are often times lost or stolen, because for the password to be secure it must contain a certain number of characters. To help eliminate this problem we propose a fuzzy password system, which is tied into a visual authentication system. A user can set his or her password by choosing a sequence of images which make up the password. When a user wishes to log in, the fuzzy password system allows some errors in the sequence such that a user only needs to select most of the correct images in order to be authenticated.

4. OUR SECURE SYSTEM PROPOSAL

4.1 Secure Key Exchange

In order to create a secure channel for communications between the mobile device and the server, the Diffie-Hellman key exchange protocol was chosen [11, 12]. This protocol provides a straightforward method of creating a symmetric session key between two parties to be used for message encryption. However, the Diffie-Hellman protocol by itself is vulnerable to a man-in-the-middle (MITM) attack. A malicious entity could pretend to be the server while communicating with the mobile device, and pretend to be the mobile device while communicating with the server. Thus, the malicious entity would create two session keys, one between itself and the server, and another between itself and the mobile device.

In order to provide entity authentication for the mobile device and the server, asymmetric RSA keys [9] were chosen to digitally sign the messages exchanged during the Diffie-Hellman key exchange. The choice to use RSA keys allows the mobile device to verify the identity of the server, and allows the server to verify the identity of the mobile device. Once the identities of the mobile device and server have been verified, secure encrypted communications are initiated in order to verify the identity of the user of the mobile device using an AES key [7] created during the Diffie-Hellman exchange. A diagram of the protocol can be seen in Figure 1 below. It is important to note that the client knows the RSA public key of the server, and the server knows the RSA public key of the client.

4.2 Fuzzy Password Authentication

Once a secure encrypted channel has been established, the user can submit a user name and a confidential password by selecting images on the mobile device. The first time the user logs into the system the server creates an association between the user name and the submitted password. The server also creates error correcting codes to add to the password. A password consists of 7 numbers, where each number is an integer between 1 and 255. Once the server receives the password, the digits are input into a Reed-Solomon encoder which for our method adds 5 error correcting codewords (ECC) to the initial password. (The choice of 7 password terms plus 5 ECC terms allows a user to submit a 'correct' password by choosing 5 of 7 images correctly). These 12 digits are then treated as coefficients for a polynomial, which is evaluated at several values. The original 7 coefficients submitted

to the server are discarded, while the user name, error correction code words, and evaluation points are stored on the server.

When a user wishes to log into the system, a secure communication channel is again created as before. (A new AES session key will be generated, while the RSA keys used for digital signatures are reused). Once the secure encrypted channel is created, the user will again submit a user name and 7 image selections, where each selection will map to an integer between 1 and 255. The server looks up the 5 ECC's associated with the submitted user name, and appends these ECC's to the submitted password. These values are then input to a Reed-Solomon decoder, which corrects up to two incorrect image selections from the user. The new sequence of numbers is treated as a polynomial p, and p is evaluated at the same points used during the enrollment process. If the values of the polynomial during this log-in attempt match the values created during the enrollment phase, then the user is authenticated and may begin to use resources on the server. Otherwise, the user is rejected.

4.3 Our Secure Protocol

This protocol is designed to provide entity authentication for the user of the mobile device, the mobile device, and the server (executing in the cloud). Authentication of the server and the client is accomplished using asymmetric RSA key pairs and message signing. The authentication of the mobile device's user is achieved through the use of a fuzzy picture password system. Finally, if all entities pass authentication, a secure encrypted channel is created between the cloud and the mobile device. The protocol contains 6 steps: 1) Server calculates and transmits Diffie-Hellman (DH) public value; 2) Client calculates Diffie-Hellman (DH) public value, and session key; 3) Client signs then encrypts its DH public value and sends this to the server, also transmits the DH public value in plain text; 4) Server calculates session key, verifies the client's digital signature; 5) The server signs then encrypts it's DH public value, and transmits the value to the client; and 6) the client verifies the server's digital signature as shown in Figure 1 below.

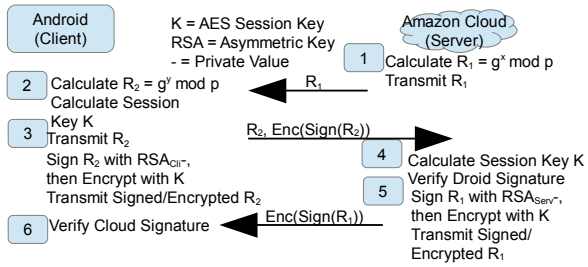

Figure 1. Secure Channel Creation, Entity Authentication

In the first step of the protocol, the server chooses a random secret x, and calculates $R_1 = g^x$ mod p, where g and p are public values. Additionally, the value R_1 does not need to be kept secret. The value R_1 is then transmitted to the client. In the second step of the protocol, the client chooses a random secret y, and calculates $R_2 = g^y$ mod p. Again, R_2 does not need to be kept secret. The random number y is seeded with random sensor data from the mobile device. Using R_1 and R_2, the client creates a session key K. The value created by the Diffie-Hellman exchange is used to create a SHA-256 message digest, which is used to create a 256 bit AES session key (K). This session key provides confidentiality for communications during later steps. The client then signs R_2 using the client's RSA private key, which provides authentication for the mobile device. (This assumes that the RSA private key has not been stolen). In the third step of the protocol, the client transmits both R_2 and the signed and encrypted copy of R_2. In the fourth step of the protocol, the server calculates the session key K in a similar manner as the client. The server then authenticates the identity of the mobile device by checking that the signed and encrypted copy of R_2 matches the plain text value of R_2. This provides authentication for the mobile device, as well as validating the session key for encrypted communications. In step 5, the server signs R_1 using the server's RSA private key, and encrypts the signed copy of R_1 using the session key K. This value is then transmitted to the client. Finally, in step 6, the client decrypts the signature sent by the server, and uses the server's public RSA key to verify that the signed R_1 matches the R_1 sent in step 1 of the protocol. This provides identity authentication for the server, and again validates the session key K. Once the secure channel is created, the user of the mobile device is authenticated by choosing 7 images which constitute the user's password. The images selected are sent as an array $P[p_1, p_2, \ldots, p_7]$, where $0 < p_i < 256$.

5. SECURITY ANALYSIS

Our protocol first provides key management between the client and server by generating a shared key on the fly using the Diffie-Hellman key exchange protocol. The security strength of the Diffie-Hellman key exchange protocol comes from the difficulty of the discrete logarithm problem [2, 11]. An attacker can relatively easily sniff and capture the public values used for the key exchange, $R_1 = g^x$ mod p, and $R_2 = g^y$ mod p. However, given R_1 and R_2, it is computationally difficult to find x and y. (If the attacker were to find x and y, then the key could be computed by $K = g^{xy}$ mod p). The Diffie-Hellman protocol has been in existence for over 20 years and there is still no efficient algorithm to solve the discrete logarithm problem. It is recommended to make Diffie-Hellman secure, the prime p must have more than 300 decimal digits, and the values of x and y cannot be reused. Finally, the prime p must be chosen such that p-1 has at least one prime factor greater than 60 digits. The Diffie-Hellman protocol is susceptible to a man-in-the-middle (MITM) attack, but this can be avoided by using RSA keys for digital signatures. Diffie-Hellman key exchange is secure because it is bulit upon the RSA cryptosystem which has no known strong attacks.

Our protocol then provides encryption services using the Advanced Encryption Standard (AES), which was developed after the Data Encryption Standard (DES) had been in existence for quite some time. The known attacks on DES were tested against AES and none of them were able to break the security of AES. Additionally, AES specifies the use of large key sizes (128, 192, or 256 bit), which helps prevent against a brute force attack.

Authentication service is provided by using a fuzzy password system for user authentication. The fuzzy password provides both authentication and usability especially when users have a difficult time remembering a password required for sufficient length and randomness to be secure. However, this system must be designed in such a way that it does not weaken the security of the password. If a mobile device is stolen, the thief could gain access to the server resources if the password is not of sufficient length. For this implementation we choose a password to be 7 numbers between 1 and 255. (Number selection is done by choosing an image, the user has no knowledge of the numbers making up his or her password). This leads to $255^7 \approx 7.011 \times 10^{16}$ possible

password combinations. However, a user only needs to select 5 out of 7 images correctly in order to access the server resources. After allowing for two missed entries, there are only 1,365,525 ($\approx 1.4 \times 10^6$) password entries that will allow access to the system. An attacker would only have a probability of 1.95×10^{-11} of guessing a correct password. Another way to view this is that for every 1 correct sequence of image selections, there are approximately 5.13×10^9 incorrect selections. The password system can also be set to only allow a small number of password attempts per minute to prevent against a brute-force attack.

The protocol detailed in this paper is resistant to *man-in-the-middle* attacks because the client and the server use digital signatures to authenticate each other. The protocol is strong against replay attacks because the public values used in the Diffie-Hellman key exchange are randomly generated each time a secure channel is created. The protocol is resistant to sniffing attacks because the data exchanged between the client and the server is encrypted using a temporary session key. The secure channel is resistant to *data modification attacks* because it is unlikely that an attacker is able to modify the encrypted channel in a meaningful way. Finally, the protocol is resistant to impersonation and loss of the device because the user of the device is authenticated through the use of a visual password.

6. IMPLEMENTATION

The system described in this paper has been implemented as a proof of concept experiment to demonstrate the protocol in action. The Android platform is used for the mobile the device. For this experiment the Android is emulated on a Windows PC using the Eclipse SDK. The Android mobile platform is a good choice for this implementation because it uses java, which can easily be run on any device that supports the java virtual machine (JVM). Additionally, through the use of the Bouncy Castle provider, java supports multiple cryptographic algorithms that are needed for encryption and decryption.

For the server side programming, the Amazon Web Services (AWS) are chosen to host the server. Specifically, the Elastic Compute Cloud (EC2) is used to host an Ubuntu Linux virtual machine which can execute the server side code. Java is chosen as the programming language for server code development again for the cryptographic support. (It is interesting to note that all the .class files created on the Windows machine were run without modification on the Linux server, as it should be).

One of the steps in the Diffie-Hellman key exchange protocol requires the use of a random number. If this random number is not chosen carefully, and an attacker gains knowledge of how this number is chosen, it may give an attacker a chance to learn the session key created between two entities. Java's SecureRandom object is passed as a parameter to the Diffie-Hellman key generator in order to provide random numbers for key generation. In this implementation we choose to use sensor data provided by the Android in order to seed the SecureRandom object, thus creating a truly random number. During the authentication stage, the user is asked to shake and twist the device. This is an approach that typically is not possible on a desktop client, but can be integrated onto almost any mobile device that has some type of sensor that provides orientation information.

7. CONCLUSION

In this paper we present a protocol for mobile device security that provides entity authentication for the user, the mobile device, and the server. Additionally, this protocol creates a secure encrypted channel between the mobile device and the server. The Diffie-Hellman key exchange protocol is used to create a shared AES session key. In our approach the client's random number generator is seeded by random sensor data in order to present a truly random number. In order to protect the Diffie-Hellman key exchange against a MITM attack we use RSA key pairs to digitally sign the public values of the key exchange. Finally, a visual authentication system is presented which maps images to a fuzzy password scheme, which helps users avoid some of the pitfalls of traditional password systems.

8. REFERENCES

[1] Bock, James T. (1996). Visual Authentication. http://websrv.cs.fsu.edu/academics/grad/cnsa/projects/bock.pdf .

[2] Forouzan, Behrouz A. Cryptography and Network Security. New York: McGraw-Hill, 2008.

[3] Hook, David. Beginning Cryptography with Java. Indianapolis: Wiley Publishing, 2005.

[4] Jansen, W., Gavrila, S., Korolev, V., Ayers, R. and Swanstrom, R. (2003). Picture Password: A Visual Login Technique for Mobile Devices. National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistir/nistir-7030.pdf .

[5] Juels A., and Sudan, M. 2002. A Fuzzy Vault Scheme. In *International Symposium on Information Theory*.

[6] Naor, M. and Shamir, A. 1995. Visual Cryptography.

[7] Advanced Encryption Standard (AES). November 2001. Federal Information Processing Standards Publication 197.

[8] Naor, M. and Pinkas, B. 1997. Visual Authentication and Identification. In *Lecture Notes in Computer Science*.

[9] Rivest R. L., Shamir A. and Adleman, L. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In *Communications of the ACM.*

[10] Feige, U., Fiat, A., and Shamir, A. 1988. Zero-Knowledge Proofs of Identity. In *Journal of Cryptology*. (77-94).

[11] Diffie, W., and Hellman, M. E. 1976. New Directions in Cryptography. In *IEEE Transactions on Information Theory.* IT-22, 6 (November 1976).

[12] Diffie, W. The First Ten Years of Public-Key Cryptography. 1988. In *Proceedings of the IEEE*. Vol 76, 5. (May 1988).

[13] Erica Ogg, "HP: Number of mobile apps doesn't matter," CNET News, June 29, 2011

[14] Flurry (June 2011), Mobile Apps Put the Web in Their Rear-view Mirror: http://blog.flurry.com/bid/63907/Mobile-Apps-Put-the-Web-in-Their-Rear-view-Mirror

[15] 2011 Mobile Threat Report, URL: https://www.mylookout.com/mobile-threat-report, retrieved April, 2012.

[16] Lookout Mobile Security Blog (March 2011), Update: Security Alert: DroidDream Malware Found in Official Android Market: http://blog.mylookout.com/2011/03/security-alert-malware-found-in-official-android-market-droiddream/