International Journal of Security and Its Applications
Vol. 11, No. 8 (2017), pp.77-88
http://dx.doi.org/10.14257/ijsia.2017.11.8.06

A Provably Secure Android-Based Mobile Banking Protocol

Hisham S. Elganzoury[1], Ahmed A. Abdel Hafez[2] and Abdelfattah A. Hegazy[3]
[1]*Engineering Faculty, Al-Azhar University, Cairo, Egypt*
[2]*Communication Dept., Military Technical College, Cairo, Egypt*
[3]*College of Computing and Information Technology, AAST, Cairo, Egypt*
Hisham.ganzoury@gmail.com, aabdelhafez@gmail.com, ahegazy@aast.edu

Abstract

The rising vogue of smart phones and tablets has led users to complete their daily works (such as M-Banking) with these devices. Therefore, mobile banking needs to become more proper, reliable, effective; and secure. Security is the most crucial requirement in mobile banking, since all the communications are via unsecure networks such as the Internet. Providing main security services; Confidentiality, Integrity, and Authentication (CIA) between any two communicating parties must be ensured and guaranteed. Many vulnerabilities may make Users' confidential information vulnerable to risks. These vulnerabilities can take different shapes, such as fixed values-based security techniques, one factor authentication, separate hard token-based authentication, hardware thievery, and Android OS based attacks. This paper proposes a new secure scheme for mobile banking applications to overcome these risks. Then, the proposed scheme is analyzed, and compared to the most powered approaches. Finally, performance key identifiers are assessed and validated.

Keywords*: Android OS Security, M-Banking, Variable security primitives, One-Time-Password, Multi factor Authentication.*

1. Introduction

Smartphones provide most of computer-based services and facilities. For that and other attractive features such as its light weight, limited size, availability of both voice and data communications through one device, high speed core processors (up to 16 core processors), existence of GPS, and other sensors and peripherals; users prefer to replace computers with smart phones. Its software includes modern OS (iOS /Android/Win8) with integrated support for all hardware, and many standard built-in components, such as, web browsers, social media applications, and search engines [1]. Mobile-based services grow from day to day with very large rates, and pervade all life needs, which mark the presence of confidential information and vulnerability to cybercrimes [2]. Our work, as described in this paper, demonstrates that, it is possible to keep classified information on a modern Android-Based device, to securely process data using the device, and communicate classified information with a remote entity in a competitive security level with that provided in wired communication. I.e. keeping data secure in its three possible states, at rest, under processing, and in transient. All this security features can be achieved, despite the many privilege escalation problems that plague the Android platform. In this paper, a new proposal scheme will be presented, and then the proposed scheme is implemented using a real Android-Based mobile set using Java-based Android. Finally, the security of the proposed scheme is analyzed using Schyther verification tool [3]. The following subsections give some brief notes on technologies used in the research.

1.1. Wireless Public Key Infrastructure (W-PKI)

keeping security level of wireless internet at the same level as in wired internet is the main purpose of wireless security studies. Ordinary PKI (Public Key infrastructure) which is used for securing e-commerce in wired internet is not appropriate to mobile communication because of mobile devices platform limitations such as limited memory, limited physical size, and limited computation capabilities. The PKI Components are [4]

1) Certification Authority (CA)
2) Registration Authority (RA)
3) Certificate Distribution system or Repository
4) Certificate Revocation List (CRL)
5) X.509 Public key Certificate.

Some efforts were exerted to instantiate a new version of PKI with wireless features (W-PKI), those efforts will be discussed in part 2 in a very brief fashion while more details are discussed in [1].

1.2. Android Operating System

Android is an open-source, Linux-based operating system that was developed under the leadership of the Open Handset Alliance (OHA) and Google. Android dominates all other Mobile phone operating systems due to its open source nature, ease of application development and publishing process, and competitive price. According to IDC report for 2017Q1, Android OS takes over about 85.0% of the market share among other operating systems [5]. However, security risks and threats have increased and continue to increase more than other mobile platforms, such as Apple's iOS. To achieve the goal of security, Android OS supplies the following security features:

1) Powerful security mechanism on the Linux Kernel Level.
2) Mandatory application isolation (sandboxing) for all the applications.
3) Secure inter-process communication
4) Application signing
5) User approved and application specific permissions.

The basic structure of an Android application consists of the following:

1) Activities
2) Android Manifest File
3) Broadcast Receivers
4) Services

As a default, each application has access to limited system resources. The permission mechanism handles access issue to those resources and checks whether they properly access the resources and do not maliciously behave. Constraints are developed by using different techniques. In certain cases, storage isolation is selected for protection; in other cases, constraints are created based on the Permission List mechanism that restricts access to sensitive APIs (Application programming interfaces). A few of these protected APIs include Camera, Location (GPS), Bluetooth, Phone, SMS/MMS, and Network/Data (3G, 4G and Wi-Fi). These APIs can only be used via the operating system [6]. Each API call in the Android operating system corresponds to permission in the manifest file (AndroidManifest.xml) that contains the list of permissions. When a user installs an application, the list of permissions is appeared to the user. When the user clicks “Accept” which grants all presented permissions, API calls become active. However, users can only allow or reject all the permissions and do not have the power to select certain permissions [7]. Some of recent related works that tried to alleviate the Android OS-related risks are well summarized in [8, 9].

1.3. Schyther Test Tool

Schyther is a protocol security analysis tool under the condition that, all cryptographic functions are perfect. Schyther takes the input protocol, which is represented in a specific semantic language, with its security properties, and then it accepts a list of design security claims, such as the key that is supposed to be secret and synchronized. The tool can characterize the roles that are involved in the communication process, test protocol security with specific designer chosen security claims, and test protocol security with automatic suggestion of security claims. Protocol description is loosely based on C/JAVA-like syntax. The protocol is defined as a set of roles (communicating parties). Roles in its turn have some events which describe the behaviour of those roles. Schyther tool allows the user to define variables, functions, new user types, and claims [10].

The remainder of the paper is structured in the following sequence: Section 2 introduces the related works, while in Section 3, the proposed scheme is briefly described and M-Banking key performance considerations and identifiers are listed. After that, Section 4 shows the practical results comparison to a related work. Finally, Section 5 concludes the proposed scheme and suggests some future points to work on.

2. Related Work

2.1. Android OS Based Solutions

This approach of security solutions relies on modifying the operating system itself. This type of modification targets only users with technical awareness as he/she needs to select among permissions. In [11] authors developed a system called (APEX) that allows selective permissions. Users have the ability to accept or reject any of the services that are listed on the interface during the application installation phase. This system achieves its goal by making changes on the operating system code. This approach does not contribute to the security or privacy of users who have no technical knowledge. This also keeps those devices which use the modified operating system versions outdated from current Android corrections and updates (fractioning problem).

2.2. Permission-Based Solutions

The Android OS model does not support dynamic permission assignment. Researchers, in this approach, usually provide statistical results based on experimental analysis to reveal a complete picture about how permissions are used, misused, excessively used, or incompletely used as introduced in VetDroid application [1], [12]. Other works rebuilds the applications according to analysis performed as introduced in Dr. Android. And Mr. Hide [1], [13]. However, Mr. Hide module adds about 10-50% overhead on the system, and the time required to rebuild applications takes an average of 60 seconds. Running in the background as a service overloads the whole system. Also, rebuilt applications run slower than original versions. Finally, the proposed application was applied on a small set of applications, about 19 applications, which doesn't give sufficient impression about its performance [8].

2.3. Byte Code-Based Solutions

Studies under this category provide solutions by processing the byte-code of applications. In [14] authors proposed APPGUARD application that extracts the byte-code files and inspects policies, then, rebuilds the byte-code files of the applications according to generated restrictions. Without any doubt, this overloads the processor as applications built on the mobile set itself. Time used to rebuild applications cannot be neglected, for instance, the time elapsed to rebuild the popular game of Angry Birds is 45 seconds, while Instagram requires 66 seconds and WhatsApp Messenger requires 58 seconds. Furthermore, the overload on function calls that is caused by applying the security policies is 5% [14].

2.4. Mobile Platform Related Works

When turning the discussion to the Mobile phone platform, it provides better security services at different levels, such as storage and messages encryption and authentication algorithms with stronger combinations than those used in android applications. Different approaches were introduced under this category, such as in [15], in which authors proposed an application that can connect many banks with two levels of security, first is user level security which lies in TPA (Third Party Agent) that provides secret keys (passwords) of 6 digits for user by sending it to the authorized email, while second level is network level security which lies in using HTTPS that is used to connect the mobile data base to the server. While in [16] authors introduced the MHA (Mobile Home Agent) which performs all cryptographic operations in favour of mobile phone such as requesting and obtaining the digital certificate, and generating ECC-based public-private key pair. In [17] a one-time-password and a personal biometric have been combined with personal identification and password for verification while M-Banking. As the user asks for a transaction request, the server side will then generate an OTP and transfer to the default receiving equipment. If the input one-time password (OTP) (by the user) is correct, the user will be asked for taking a fresh biometric data and uploading it to the server side for comparison with the stored data. In [18] authors deployed the PDE (plausible Deniable Encryption) in Mobile communication after using in the desktop environment [18].

3. Proposed Scheme

3.1. Key Performance Considerations and Identifiers

From part 2 and after careful extrapolation, it is set in mind that, Mobile Banking scheme should possess the following considerations: low key size for all running algorithms, low processing time, and low display requirements. Higher data encryption level, good digital signature level with lowest key size, and multi factor authentication are highly required. Ease of use (i.e. it shouldn't require any user to have a technical background), end-to-end security, resistance to attempted thievery or loss, high level of dynamicity and randomness, and resistance to most common attacks such as replay, birthday, Distributed Denial of Service Attack (DDoS) and dictionary attacks have to be granted as well.

3.2. Proposed Scheme Description

The preliminary design of the below work was published in [1] by the author, while in this paper the final design, implementation, practical results, comparison to related works, and validation of assessment criterion (mentioned in Part 3.1) are presented. In brief, the proposed scheme consists of three elements, Android-Based Mobile device with attached SD Card, Bank Server with local Database, and Bank cloud storage and processor. Android App installation, second level authentication password registration, and message exchange are the main three phases occupied during the proposed scheme operation. Upon completion of bank account issuance procedures, the following are concluded, user's digital certificate, user's first level authentication password, and encrypted SD card that is loaded with user's digital certificate, initialization vector set, all cryptographic algorithms, bank digital certificate information such as public key, CA name… etc. and user's bank account information. To guarantee first level password security, user is required to log in to bank mobile web site to install the bank special android application within limited period. This application will permit the user to choose his second level password. The idea behind is to keep some of user's credentials away of bank admins reach. Figure 1 shows the detailed procedure of Android App installation phase. The installation process will be continued the below criteria has been met

User's digital certificate existence =Passed
User's device IMEI correspondence=Passed
User's device malware scan=Passed

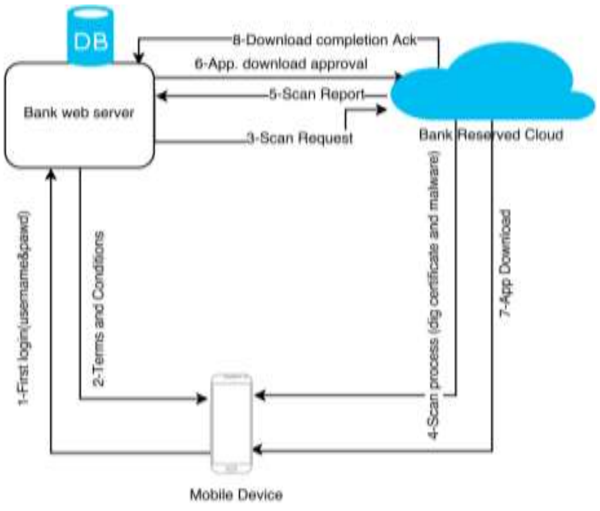


Figure 1. Bank Special Application Installation

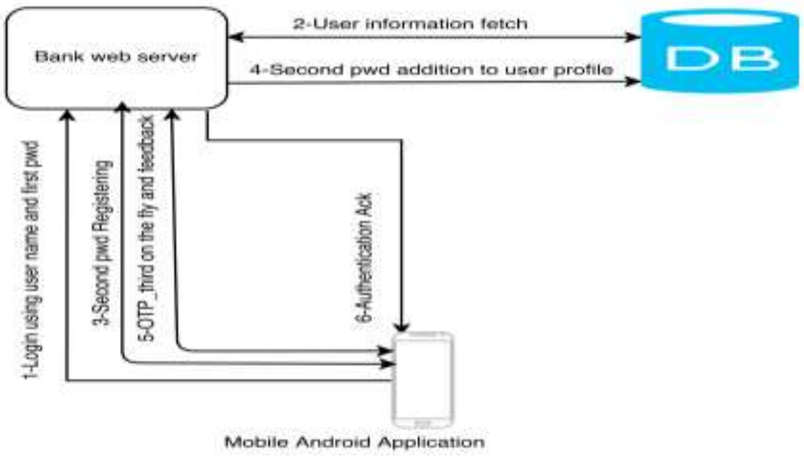

Bank web server
2-User information fetch
4-Second pwd addition to user profile
DB
1-Login using user name and first pwd
3-Second pwd Registering
5-OTP_third on the fly and feedback
6-Authentication Ack
Mobile Android Application

Figure 2. Mobile-Bank Authentication

Then, user will be moved to the next phase, where user is required to choose his/her strong second level password as in figure 2. Second level password is enforced to be not less than 8 characters, to include at least one special character, one number, and any upper-case letter. Second level password negotiation is performed in encrypted fashion, and credentials such as second level password is sent in a hashed fashion. Second level password will partially open outer of the SD card as shown in figure 3. So far, the application cannot read, write, or use any SD resident information. The third level authentication is performed as follows, after registering the second level authentication password, and after user decides he/she will proceed in a transaction, bank will generate an OTP called OTP_third as in figure 5 which will be put (on the fly) to a specific directory, then user will brows it. Now, application has a full privileged access to the SD card AES-XTS part.

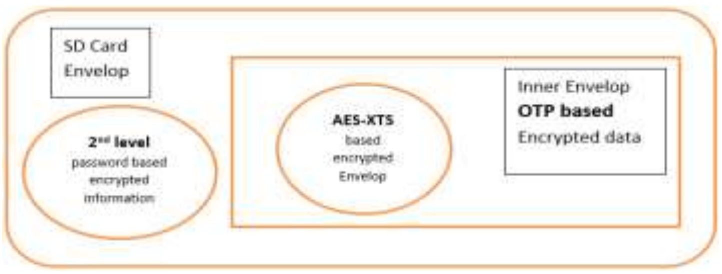

SD Card Envelop
2nd level password based encrypted information
AES-XTS based encrypted Envelop
Inner Envelop OTP based Encrypted data

Figure 3. SD Card Internal Layout

Figure 4 explains the message exchange protocol diagram between user's mobile device and bank server until the transaction will be completed. The message composition is as follows:

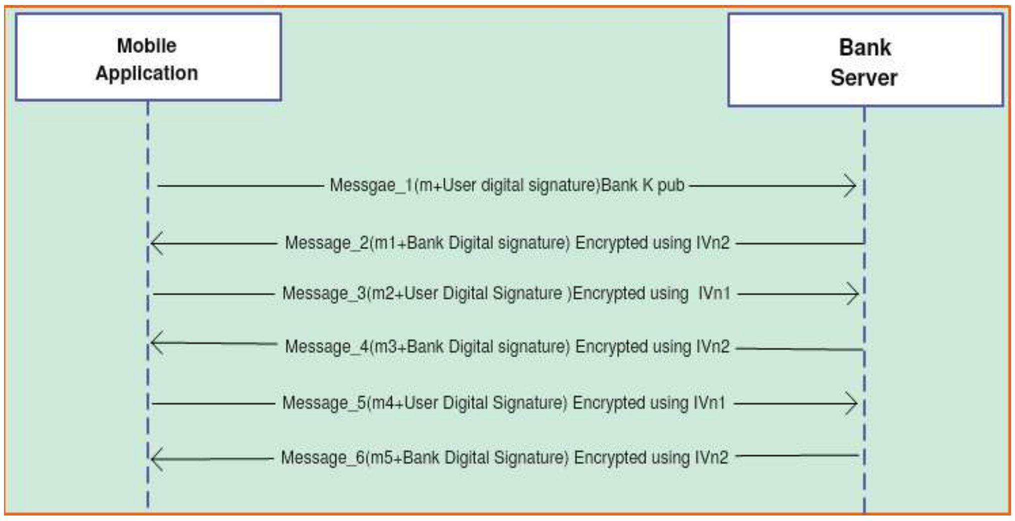

Mobile Application
Bank Server
Messgae_1(m+User digital signature)Bank K pub
Message_2(m1+Bank Digital signature) Encrypted using IVn2
Message_3(m2+User Digital Signature)Encrypted using IVn1
Message_4(m3+Bank Digital signature) Encrypted using IVn2
Message_5(m4+User Digital Signature) Encrypted using IVn1
Message_6(m5+Bank Digital Signature) Encrypted using IVn2

Figure 4. Message Exchange Protocol Diagram

Message_1 is initiated at user's mobile device and looks like a transaction initiation request messages. Message_1 comprises only information required to fetch the user's tables from bank database. User name, user digital certificate, encryption related primitives that is required for upcoming message exchange such as the index of IV among the initialization vectors set, and additional authentication objects such as OTP, and Digital signature are fetched in Message_1. As the bank server receives Message_1, bank decrypts the whole message using its private key, fetches user information from bank database tables, constructs the message digest, and compares it with the sent digest and the OTP. If it succeeds, the sent IV_n will be used to encrypt all the upcoming messages in the fashion that will be shown soon in this section. Then bank validates the user's digital certificate. After that, Message_2 is initiated at bank server which comprises account issuance information to authenticate itself to the user, transaction sequence number to overcome replay attacks, and bank digital signature. All this information are encrypted using AES-CBC-256 with the IV_n (which is generated using the algorithm published in [19]) sent by the user. Here is, a symmetric encryption algorithm is used for authentication overcoming its trusted third party support problem [20]. After that, Message_3 is initiated at mobile device side on reception of Message_2. Mobile decrypts the incoming message using the agreed IV_n; this is known by user and bank server only. After exchanging Message_3, Message_4, Message_5 and Message_6 successfully as in figure 4, a complete account transaction log is stored in the SD card to be used in the next transactions and a comprehensive encryption is applied back to the SD card.

3.3. Security Controls and Algorithms

1) **Encryption Process**: encryption process is performed using AES-256 in two different modes, first mode is AES-CBC for message encryption, and second mode is AES-XTS [21] to encrypt the SD card. One modification has been made here over that was published before in [1] is that, the initialization vectors used are 256-bits, the first 128-bits are used to encrypt messages from mobile device to bank servers, and the second 128-bits are used to encrypt the messages in the opposite direction. This mitigates play back attacks that may be utilized on the minor party (mobile device) of the communication system to break the major party (bank server) as they would use the same IV. Also, using 256 bits' key put the protocol far enough from opponents, as it needs computational complexity about 2251.92, data complexity 2120, and memory complexity 28 when compared to AES-128 which needs computational complexity about 2125.34, data complexity of 288, and memory complexity of 28 [22].

2) **Digital Signature Process**: EC-DSA (Elliptic curve digital signature algorithm) is used for digital signing purposes, as it shows attractive results as a public-key cryptosystem for wireless environments. ECC achieves a competitive security level with a key size of 160-bit when compared to a 1024-bit key RSA [23], [24].

3) **OTP Generators**: The OTP generator in figure 5 is used at both sides (bank server and mobile device) to generate OTPs. The OTP generator is designed to meet diffusion, complexity, and nonlinearity properties. It uses randomly selected digits from the second level password to conceal any statistical properties of inputs, last transaction information and IV_n to destroy the static nature of some OTP generators, and hash functions (SHA-256) to guarantee a fixed length, and irreversibility.

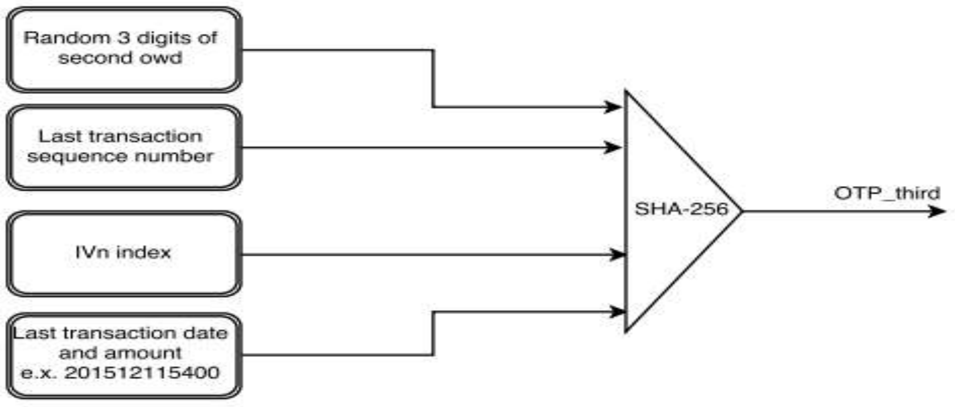
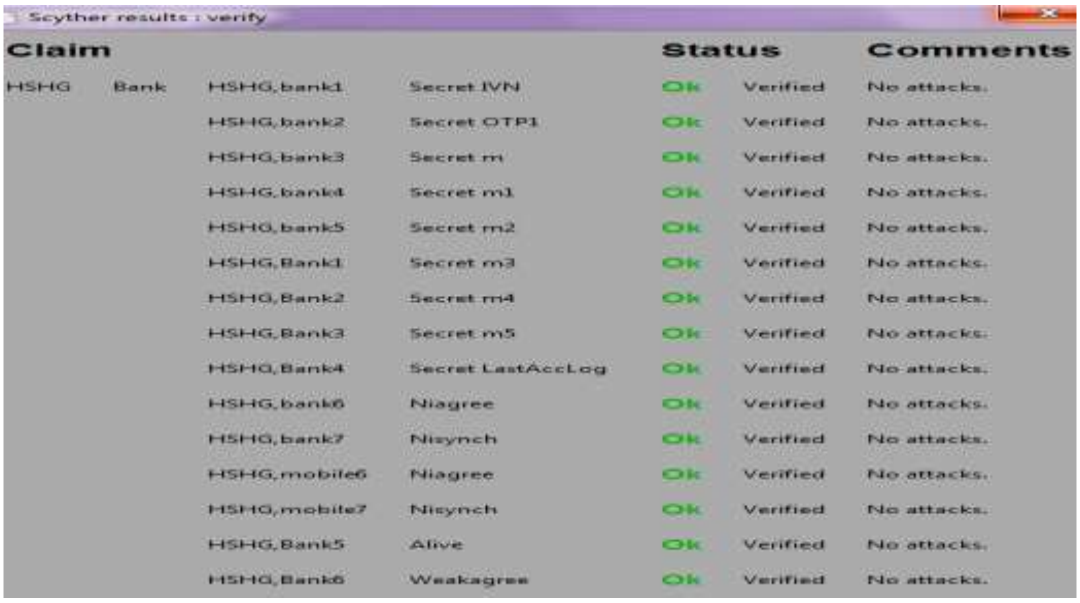

Random 3 digits of second owd
Last transaction sequence number
IVn index
Last transaction date and amount e.x. 201512115400
SHA-256
OTP_third

Figure 5. OTP_third Generator

4. Practical Results and Comparison to Related Work

4.1. Scyther Analysis Results

Schyther analysis tool is a formal protocol analysis tool used to check the protocol security of the message exchange phase to guarantee nonexistence of threats. Secrecy, aliveness, agreement, and authentication, are security properties that are analysed and verified. Aliveness claim means that, if an agent A executes a role of the protocol, thinking she/he ran it with B, and then B has indeed performed an action (alive). While weak agreement Aliveness, where additionally B assumes that he is communicating with A, and hence they both "agree" on the agents involved in the protocol. Agreement claim implies weak agreement and additionally, B are indeed performing the role that A assumes. Finally, A and B agree on the values in {s}, e.g., they agree on the computed session key. Secrecy claim is essentially the statement that the attacker never knows anything. Nisynch claims, the notion of synchronization which requires that corresponding send and receive messages must be executed in the expected order.

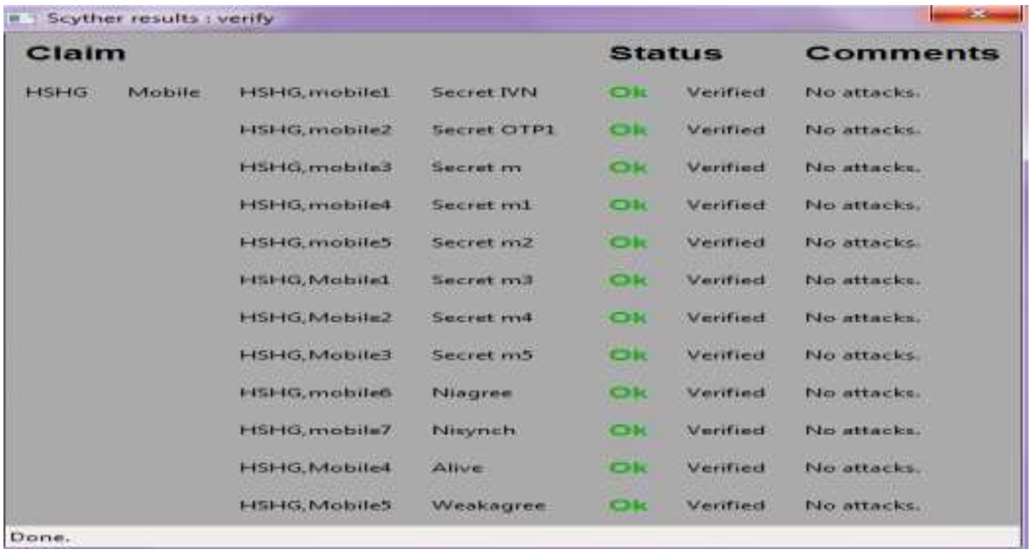


Scyther results : verify

Claim				Status		Comments
HSHG	Bank	HSHG,bank1	Secret IVN	Ok	Verified	No attacks.
		HSHG,bank2	Secret OTP1	Ok	Verified	No attacks.
		HSHG,bank3	Secret m	Ok	Verified	No attacks.
		HSHG,bank4	Secret m1	Ok	Verified	No attacks.
		HSHG,bank5	Secret m2	Ok	Verified	No attacks.
		HSHG,Bank1	Secret m3	Ok	Verified	No attacks.
		HSHG,Bank2	Secret m4	Ok	Verified	No attacks.
		HSHG,Bank3	Secret m5	Ok	Verified	No attacks.
		HSHG,Bank4	Secret LastAccLog	Ok	Verified	No attacks.
		HSHG,bank6	Niagree	Ok	Verified	No attacks.
		HSHG,bank7	Nisynch	Ok	Verified	No attacks.
		HSHG,mobile6	Niagree	Ok	Verified	No attacks.
		HSHG,mobile7	Nisynch	Ok	Verified	No attacks.
		HSHG,Bank5	Alive	Ok	Verified	No attacks.
		HSHG,Bank6	Weakagree	Ok	Verified	No attacks.

Figure 6. Bank Role Verification with Claims

Figures 6, and 7 show the output results from Schyther analysis tool. Figures show that, the protocol (protocol is named HSHG) passed the secrecy claim as its secrete information is sent in an unclear (encrypted form) such as IV_n, OTP1 =OTP_third using IV_n, m = user name||index of IV_n||OTP1||IMEI, m_1=bank digital certificate info||sequence number||Account issuance information, m_2= source account number||destination account number||transaction type, m_3 =current account status||expected account status, m_4=confirmation, m_5=account log||Acknowledgment. Also, both roles passed the “Alive” claim as they both interact at each message. After that both roles passed “Nisynch “claim as each message sent from the Mobile device application is received as it’s and in the same order by the bank side. The “Niagree” is passed as well, as all communication events causally preceding the claim have occurred before the claim



Scyther results : verify

Claim				Status		Comments
HSHG	Mobile	HSHG,mobile1	Secret IVN	Ok	Verified	No attacks.
		HSHG,mobile2	Secret OTP1	Ok	Verified	No attacks.
		HSHG,mobile3	Secret m	Ok	Verified	No attacks.
		HSHG,mobile4	Secret m1	Ok	Verified	No attacks.
		HSHG,mobile5	Secret m2	Ok	Verified	No attacks.
		HSHG,Mobile1	Secret m3	Ok	Verified	No attacks.
		HSHG,Mobile2	Secret m4	Ok	Verified	No attacks.
		HSHG,Mobile3	Secret m5	Ok	Verified	No attacks.
		HSHG,mobile6	Niagree	Ok	Verified	No attacks.
		HSHG,mobile7	Nisynch	Ok	Verified	No attacks.
		HSHG,Mobile4	Alive	Ok	Verified	No attacks.
		HSHG,Mobile5	Weakagree	Ok	Verified	No attacks.

Done.

Figure 7. Mobile Role Verification with Claims

4.2. Numerical Results

A sample range of about 30 trials were conducted to record the time consumed during each transaction, and the throughput which will be produced and stored in the SD card per each transaction. It is very important to reach a constant time consumption not to affect the customer experience, and as short time as possible not to make user vulnerable to attacks, and to save limited resources of mobile devices such as battery and processing resources. Regarding amount of data generated per each transaction, it is very important to be kept as small as possible not to end up the SD card capabilities. Figure 8 gives a picture of time consumed along with throughput in each trial and how it reached almost a fixed value. As it can be seen, time values (in milliseconds) are almost constant and kept at minimum level. Message number 1 composition and validation takes most of the elapsed time as Bank server fetches all information available about the user. Bank server loaded the IV-Key arrays as well. While other messages took no time as all information are available now. Other time elements are estimated with a mean of time elapsed in 30 trials such as user interaction with the application, for example to read tips, enter user name and passwords, getting out bank card, thinking about the amount, etc. Also, wireless network traffic congestion is not taken in consideration, but keeping processing time and output data as small as possible contributes in reducing its effect

4.3. Comparison to Related Work

In [24] authors mentioned elapsed times during only encryption/decryption processes for one message. Comparing the whole transaction time of the proposed scheme to the public key infrastructure scheme proposed by [24] does show the enhancement. Figure 9 shows elapsed time values (in milliseconds) for each step during the whole encryption/decryption phase of one message. RSA authentication operation takes 1047ms, while AES encryption operation takes 722ms and 3-DES takes 925ms. Referring to figure 9, it can be found that about 4332ms (i.e. 722ms achieved in [24] for each message of current proposal) would be elapsed by the whole operation, while the proposed scheme offers a total time of about 3500ms which is a good enhancement to existing solutions with respect to time.

5. Conclusion and Future Work

The proposed protocol aims to enhance Android application security. The previous section showed a successful security tests performed using Schyther verification tool under good security requirements such as synchronization, and aliveness. It showed a good enhancement to existing protocols with respect to elapsed times as well. Most of key performance considerations mentioned in 3.1 are achieved such as:

- Low key size by choosing ECC with lower key size than RSA,
- Low processing times as shown in figure 8 compared to figure 9,
- Higher data encryption levels by using AES-256,
- Using 3 factor authentication before exchanging messages and mutual authentication during message exchange,
- Ease of use as users don’t need to have any technical background but just to keep two passwords in mind,
- end to end security as there is no white areas through the transaction,
- Mitigation of malware effects by applying periodic scan to user’s mobile device
- Banning any other applications from using the SD card,
- Randomness and dynamicity by using provable random input keys and by changing inputs to OTP generator as in figure 5.
- Withstand against replay attack by using sequence numbers, dictionary attack by using long keys and by concealing user chosen passwords, and birthday attack by using SHA-2 instead of SHA-1 have been met as well.

In future, further enhancements will be tried to reduce elapsed time and added throughput. Real attacks will be conducted to guarantee scheme robustness as well. Disaster recovery and business continuity plan will be taken in consideration. Distributed Denial of service attack caused service outage will be put at an acceptable level.

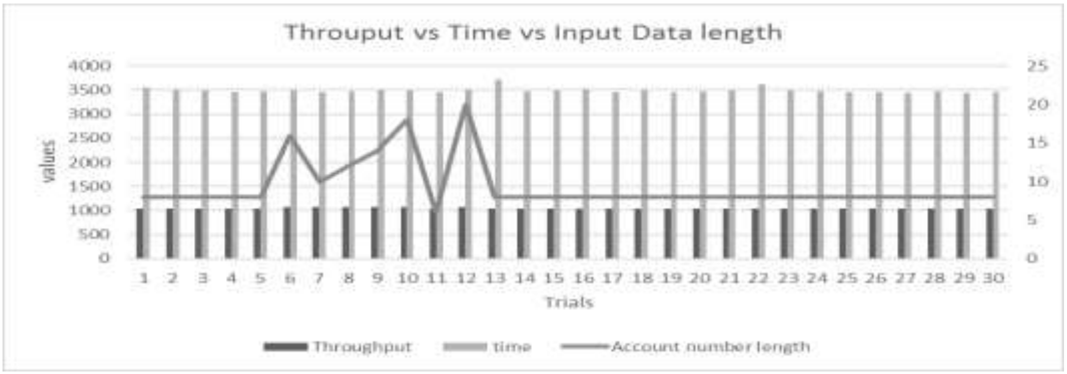


Figure 8. Time Consumption and Throughput

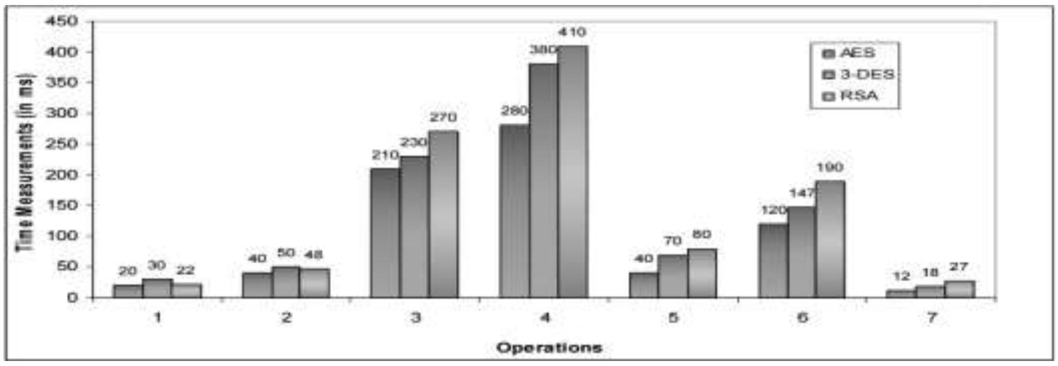


Figure 9. Time Consumption

References

[1] H. Sarhan, A. A. Hafez, A. Safwat, and A.A. Hegazy., "Secure Android-Based Mobile Banking Scheme", International Journal of Computer Applications, vol. 118, no.12, **(2015)**.
[2] Android security overview, https://source.android.com/devices/tech/security/index.html
[3] C.J.F. Cremers, Scyther: Unbounded Verification of Security Protocols". Information Security, ETH Z¨urich, IFW C Haldeneggsteig 4 CH-8092, Technical Report No. 572, Z¨urich, Switzerland. **(2014)**.
[4] Cooper, Santesson, Farrell, Boeyen, Housley, Polk Internet. X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" RFC 5280, **(2008)**.
[5] http://www.idc.com/prodserv/smartphone-os-market-share.jsp, last visited 19-Aug-**2017**
[6] Miriam, Ben-Av, and, Gerdov, StoreDroid: Sensor-Based Data Protection Framework for Android. Wireless Communications and Mobile Computing Conference (IWCMC), **(2014)**, pp. 511 – 517 .
[7] Google website [Online] Android security overview. https://source.android.com/devices/tech/security
[8] Yuksel, Zaim, and, Aydin, "A Comprehensive Analysis of Android Security and Proposed Solutions" I.J. Computer Network and Information Security, **(2014)**.
[9] Cas Cremers. Scyther User Manual., February 18, **(2014)**
[10] M. Nauman, S. Khan, and X. Zhang, "Apex: Extending android permission model and enforcement with user-defined runtime constraints", Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS '10, pages 328–332, New York, NY, USA, **2010**. ACM. doi:10.1145/1755688.1755732.
[11] Y. Zhang, M. Yang, Z. Yang, G. Gu, P. Ning, and B. Zang. Analysis for Vetting Undesirable, IEEE Transactions on Information Forensics and Security, Vol. 9, No. 11, Pages 1828-1842, November **2014**.
[12] Jinseong, Micinski, Jeffrey, Nikhilesh, Foster, Fogel, and Millstein, Dr. Android and Mr. Hide: Fine-grained Permissions in Android Applications, SPSM'12, October 19, **2012**, Raleigh, North Carolina, USA.
[13] Backes, Gerling Hammer, and Styp-Rekowsky. AppGuard - Enforcing User Requirements on Android Apps. In Proceedings of DPM/SETOP, **(2013)**, pp. 213-231.
[14] Amol Bhatnagar, Shekhar Tanwar, and R.Manjula ,Secure Multiple Bank Transaction Log" Inter. Journal of Research in Eng. And Technology, **(2014)**.
[15] Sangram Ray and G.P. Biswas, Design of Mobile Public Key Infrastructure (M-PKI) using Elliptic Curve Cryptography, Int. Journal on Crypt. And information security (IJCIS), vol.3, no.1, **(2013)**.
[16] Chang-Lung Tsai Chun-Jung Chen and Deng-Jie Zhuang, Secure OTP and Biometric Verification Scheme for Mobile Banking, Third FTRA international conference on mobile, Ubiquitous and intelligent computing
[17] A. Skillen and M. Mannan. (**2014**). Mobiflage: Deniable Storage Encryption for Mobile Devices., IEEE, Transaction on dependable and secure computing, Vol11, No.3 May-June **2014**
[18] Hisham S Elganzoury, Talaat A El-Garf, A A Hafez and Ahmed Safwat. Enhanced Stream Cipher Algorithm using Consecutive Nonlinear Functions. International Journal of Computer Applications 123(10):33-37, **(2015)**.
[19] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid N, Recommendation for Key Management – Part 1: General (Revision 3). NIST Special Publication 800-57, **(2012)**.
[20] Morris Dworkin, Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices. NIST Special Publication 800-38E January, **(2010)**.
[21] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger, Biclique Cryptanalysis of the Full AES. K.U. Leuven, Belgium; Microsoft Research Redmond, USA; ENS Paris and Chaire France Telecom, France.
[22] Kamlesh, and Sanjay, ECC over RSA for Asymmetric Encryption: A Review. IJCSI International Journal of Computer Science Issues, vol. 8, issue 3, no. 2, **(2011)**.
[23] Maryland University Website [Online] W. Chou. Elliptic Curve Cryptography and Its Application to Mobile Devices. http://undergrad.cs.umd.edu/departmental-honors.
[24] C. Narendiran, Albert Rabara and N. Rajendran. Public Key Infrastructure for Mobile Banking Security, 978-1-4244-2829- 8/08/25. IEEE, **(2008)**.