Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 4– No.5, February 2016 – www.caeaccess.org

Security Research of a Social Payment App and Suggested Improvement

Swapna Khandekar
Department of Electrical Engineering and Computer Science
Cleveland State University, OH, USA

Jingyuan Liang
Department of Electrical Engineering and Computer Science
Cleveland State University, OH, USA

Abdul Razaque
Department of Electrical Engineering and Computer Science
Cleveland State University, OH, USA

Fathi Amsaad
Department of Electrical Engineering and Computer Science
University of Toledo, OH, USA

Musbah Abdulgader
Department of Electrical Engineering and Computer Science
University of Toledo, OH, USA

ABSTRACT
Electronic commerce has become integral part of business operation and individual person's life. It is easy, fast and reliable way of money transfer. However with new technology security related issues have increased drastically. In this research, study of a payment application "Square Cash" has been done from different aspects like checking security related issues and suggestions have been given for improvement.

To measure performance of used payment procedure in square cash, one should understand flaws in depth in payment application. This research analyzes the security related issues of application "Square Cash" and highlights flaws in existing application. Author has used different methods to inspect this application which includes doing reverse engineering, observing and finding risks related to social engineering attacks. Also they have proposed secured payment protocol using self-certified key generation method. Introduction of this cryptographic system will keep transactions more reliable and secured. This research provides developers guidelines to build secure and usable online payment applications. This will result in a better payment application which will gain customers trust and will increase e-commerce business.

General Terms
Mobile Payment app, Security, Payment Protocol, Social engineering attacks, Private Key, Public Key

Keywords
Square Cash, Cryptography, Key Generation, Payment Gateway, Encryption.

1. INTRODUCTION
Payment developments grow as commodity exchange grows. People are always seeking some fast, secure, and reliable way to make payments. With the development of computer systems, more and more payments are done within computer systems [1]. Nowadays the usage rate of smart phones is increasing rapidly and undoubtedly people are eagerly switching to mobile platforms to make payments [2]. With the help of all kind of computer systems including smart phones, payments can be done much faster than before, but it is not necessarily more secure than before [3].

Many authentication and authorization methods have been developed for traditional desktop computer systems for a secure payment but users always find it too heavy and not agile enough, especially when it comes to mobile platforms where people expects that taking out a phone and making a payment to a friend can be as easy as taking out some banknotes and give them to a friend [4]. Furthermore mobile platforms have their unique attributes, such as non-transparent use of mobile devices, risk of new technologies introduced every year such as near-field communication (NFC) and quick response (QR) code, and more [5]. At the same time due to a relatively shorter history of mobile development and many related technologies, many security risks have not been realized by people, which further worsen the security situation on mobile phones [1].

Security on a mobile payment app can be affected from many aspects, which can be something challenging. There are always new things in this field, for example, a recent security incident called XcodeGhost popped up in iOS development community in China [6]. Although the exact method used by its author is somewhat creative, the fundamental concept is not new [4]. Traditional security issues seen on desktop environments may also apply [7] and users may even be switching between them [8]. Besides the part in frontend and user interface, "security in payment app" must also include security consideration of the backend, which is often tightly linked to the whole e-payment system, whose security is also a serious topic [1]. Apart from pure technical considerations and related vulnerabilities, social engineering attacks are more and more important these days [9].

People can send money to other people, using social payment applications on their own devices such as computers or smart phones, with help of some information that can be known easily by their friends, such as telephone numbers or e-mail addresses, instead of boring bank information [10]. Square Cash is one of these payment applications, and it is used as an example to do some security analysis in this research. Every user account in square Cash is associated with unique email address and phone number. However, multiple accounts can be linked to same debit card. It can be used for personal as well as business use. So security becomes integral part of online fund transfer [11].

The rest of the paper is organized as follows: Section 2 reviews problem identification and significance. Section 3 provides the description of similar research and some other related work. Section 4 describes the analysis of Square Cash application. Section 5 presents suggestions and finally section 6 gives conclusion.

2. PROBLEM IDENTIFICATION AND SIGNIFICANCE

Authors have conducted some security analysis for Square Cash, a social payment app which has been chosen as a target. Literature suggests some potential security vulnerabilities of mobile apps and after testing those items accordingly, they have tried to give suggestions to fix them. [5].

Some potential issues include improper local sensitive data processing and storage, handling of insecure and untrusted input from user and operating system. Communication between servers at another place. There is need to check for weakness, such as insufficient transport-layer protection, poor or missing authorization and authentication and weak server-side control [5], [12].

Electronic commerce has become integral part of business operations. In fact, every individual prefer to do online fund transfer instead of doing cash payment. So network security becomes important factor for gaining customer's trust and ultimately increasing electronic commerce [1].

To deal with issues related to security, one should understand potential vulnerabilities, so as to find resolution for it. Here, in our research paper, they have scrutinized techniques used in online payment for target payment application "Square Cash" and audit report will help customers to understand reliability of this application [3].

This research will be useful for users in handling issues related to unauthorized access, online scam / theft, fraud prevention. Also it will improve security-integrity, non-repudiation, authenticity, confidentiality, privacy and availability [7].

3. RELATED WORK

3.1 Security Research of another Social Payment app- VENMO

In [10], another application Venmo is used for security research. In their research, they searched for technical and social vulnerabilities and found several issues in the application. They also gave some suggestions for Venmo team to improve their application.

The research group downloaded the APK file of Android version of Venmo, then used dex2jar utility to decompile the file, and tried to figure out how the application works. They also looked at the web application of Venmo payment system, and found a private API endpoint providing some API calls which are not publicly documents.

These APIs are used by both the web application and the Android application. By checking the API and the whole process about how official Venmo applications work, including the Android application and its corresponding web application, they found several security issues.

For the security problems identified, the research group proposed their solutions, including increasing the length and setting rate limits for the authorization text message, fixing known security bugs by complying with security policies throughout the whole system and not exposing secret values accidentally, and improving web design to allow users to identify other users easier.

For target application, Square Cash, there are some reviews and comments available but no public security analysis report is available.

3.2 Potential Security Issues in Mobile Applications

E-commerce is considered as the buying and selling of products or services over internet, however transaction completely done online can also be considered as e-commerce. It offers great opportunities to banking industries, but at the same time, it gives rise to new security threats. So few researchers have studied security issues in e-commerce and as per their analysis security-Integrity, Non-repudiation, Authenticity, Confidentiality, Privacy, Availability are principle areas which needs to considered [3], [7].

Author [3] has mentioned 3 types of security threats – denial of service, unauthorized access, theft and fraud. Spamming and viruses are two major factors causing denial of service. Due to illegal access to system, application and data security threats increase. Stolen data can lead to fraud in business. To deal with this, researchers have suggested few security tools like firewalls, public key infrastructure, encryption software, digital certificates, digital signatures, biometrics, passwords [3], [10].

Some people did research on various security attacks for Mobile Ad hoc networks. The mobile nodes communicate with each other by wireless radio links. Due to possible security attacks issues like leaking secret information, message contamination and node impersonation can happen. So they have created new routing protocol called Cryptographic Hybrid Key Management for secure routing in MANETs [12]. In Cryptographic Hybrid Key Management protocol, sender's AES symmetric key is generated for message and receiver's RSA public key is used to encrypt the AES secret key. Also the message is encrypted with the sender's RSA private key so as to maintain high security levels. It helped to maintain and improve confidentiality, availability, integrity, authenticity and non-repudiation [12].

3.3 Payment Protocol for Vehicular Ad Hoc Network

Vehicular ad-hoc network (VANET) - This type of network consists of vehicles and roadside infrastructure. Value added application in VANET improves passengers comfort and at the same time it offers great value business opportunities. Online fund transfer is easy and fast, however it is risky at the same time and it becomes more risky in Vehicular ad-hoc Network. In one of the research paper, Author has suggested an efficient and secured payment protocol for VANET [13].

This protocol uses self-certified key agreement to establish symmetric keys and it provides the advantages of both public-key cryptography and symmetric-key cryptography. This protocol guarantees robust security protection. This key agreement process can be integrated with the payment phase. Due to self-certified public keys, the protocol is avoiding large use of public keys.

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 4– No.5, February 2016 – www.caeaccess.org

4. ANALYSIS OF SQUARE CASH

Based on information, we have applied following analysis to Square Cash payment application in order to scrutinize reliability and security.

```
18 @Module(complete=false, library=true)
19 public final class ApiModule
20 {
21   public static final String PRODUCTION_API_URL = "https://api.squareup.com";
22   public static final String PRODUCTION_BLE_PUBLIC_KEY =
   "30820122300d06092a864886f70d0101010500038201f003082010a0282010100f591126eac
23   public static final String PRODUCTION_CHARACTERISTIC_UUID = "70363bb1-a432-
24   public static final String PRODUCTION_SERVICE_UUID = "3a6d75e1-f747-4e79-a3
25   private static final String USER_AGENT_FORMAT = "%s/%s (Android %s; %s %s %
26
27   static AppService createAppService(RestAdapter paramRestAdapter)
28   {
29     return (AppService)paramRestAdapter.create(AppService.class);
30   }
```

Fig 1: SQC de-compilation attempt

4.1 Square Cash Android Application Analysis

Android application are normally distributed in the form of Android application package (APK) files, where all program code and some or all resources are packaged into a single file with an extension of .apk. When an application is installed from Google Play, user doesn't have direct access to the APK file. However it's possible to get the file from Google Play using some non-standard methods or from a rooted phone. An APK file is basically a ZIP archive, and there's a file classes. dex inside containing executable code. This file can be converted into a JAR file, which is another ZIP archive and contains Java byte code files. It can be either manually inspected directly in a form similar to assembly code, or reconstructed into Java source code [14] using some decompiling tool as shown in figure 1.

In test, first they have retrieved the APK file of application from a cell phone with this application installed. Then, they unpacked the APK file, converted the file classes.dex, which contains executable code of this application into JAR file with dex2jar [15], for easier reverse engineering. The JAR file is then decompiled with Java de-compiler [16] into Java source code. By reading the source code, they found their API endpoint and it is possible to analyze further to document the whole API. The developers did not actively prevent this by obscuring source code before compiling [17]. The explanation about Square Cash application saying that the application utilizes some other Email components on the same system [18] is not true anymore. Instead it communicates with Square server directly to complete the whole payment process.

4.2 Security of Network Communication in Square Cash

Traditionally, data were transferred over the internet in unencrypted plain text form, more or less for human readability reasons, especially in protocols developed in early ages. This would make it possible to observe, block or modify data traffic between two ends of connection, and make it completely transparent to both sides. A network administrator or some evil person who sets up an access point can do this easily. To avoid this, there are already several protocols which have been designed to provide communication security, among which Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL) is the most notable and widely used ones. By encrypting data traffic over the Internet with symmetric session keys negotiated using asymmetric cryptography included in X.509 certificates, they provide an encrypted data connection for upper layer protocols to make it impossible to sniffer or modify, and ensure the other side is the genuine host that is expected to communicate and eliminate Man-in-the-middle (MITM) attack [19].

Particular implementations might still be vulnerable even with TLS or SSL used. For example, in some case, some of them are not configured to validate certificates properly so it's possible to set up a fake server which gets accepted by the client and talk with two side simultaneously, consequently all data passing through can be intercepted (MITM attack) [20]. In some other cases, client or server can be instructed to switch to some lower versions of TLS / SSL protocol which are less secure, which can also be done with MITM in some situations [21].

By observing network traffic, they found that communication data are already sent over TLS. they set up a Domain Name System (DNS) server locally using dnsmasq [22] and changed system DNS configuration to point to the server we set up. On that server, we added a fake record to have api.squareup.com pointed to the local loopback address 127.0.0.1. Configured a web server using Apache [23] with self-signed certificate, and

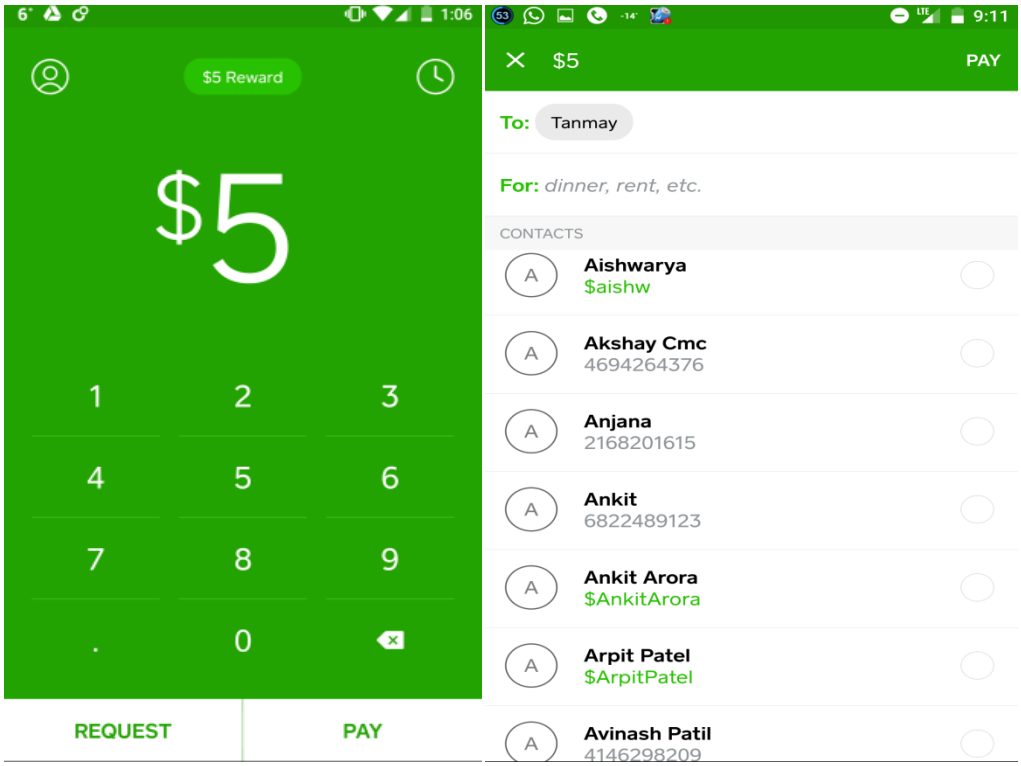

Fig 2 : Money transfer using square cash application

Checked Apache log files afterwards, and no requests were made. With behavior described above authors could conclude that Square Cash can properly guard this by implementing TLS and checking server certificates correctly, and it's not possible to attack Square Cash in this way. They could also try to intercept and change the beginning of TLS / SSL traffic, trying to force it to downgrade to a weaker encryption, and check if the application risks this kind of attacks.

4.3 Locally stored data by Square Cash

Android provides several storage APIs allowing applications to store data on the device. Internal storage is generally considered secure, which doesn't allow access from other applications and also user interface as well by default [24]. Many application developers think it's already secure enough to store data there and use data stored there previously without any validation in the future. However since the application is running on a device owned by users completely, they have the ability to get everything screwed up. For example, by "rooting" the system, they get full access to the whole file system [25], which allows them to modify data there as they want. This is often fine where users are fully responsible for results, following the "garbage in, garbage out" logic, but it's not true when it comes to network applications especially financial transactions; we do not want users to tamper with any data used to initiate a secure transaction.

Google introduced an API called SafetyNet allowing developers to check if a device is compatible [26], which also checks the possibility of any local data modification. Google's payment product Android Pay even refuses to run on rooted or otherwise modified Android phones, although it is considered too aggressive by many users. Square Cash can run on such "insecure" phones. In the analysis against it, one can check what kind of data is stored locally, and determine if any sensitive data is stored. For each interesting item, one can try to edit it and run the application, and see whether it accepts it dumbly with unexpected result, or filters and rejects it. If unexpected scenario happens, everyone can determine if it can be used in some malicious way.

4.4 Social engineering attack possibility in Square Cash

Nowadays people are moving more towards online social activities; here everyone can communicate with their friends. They can send you a request for money and you are happy to accept it; devils can too, and they pretend to be your friend. It may just looks like a scam but most of time it ties to some technical feature [9]. Money transfer can be done using square cash as shown in figure. 2

Although there are some attempts to detect social engineering attacks [27], it is still difficult to distinguish them from normal transaction since in both scenarios people are sending money themselves. It is always a better approach to allow people to protect themselves. For example, sometimes the system just displays a name or picture which can be mimicked by someone else and we can avoid this by allowing users to identify other people clearly, or do more to check if it's the first time to interact with this particular person and give users a notice. Researchers have tried to use every feature in the application especially where a user interacts with another person, and check whether the application has put a reasonable effort to prevent users from being a victim of social engineering attacks.

5. PROPOSED PAYMENT PROTOCOL

Square Cash is currently working on smart phones using the standard networking facilities. Network availability is limited by the availability of access to the Internet in hosting devices, and limitations of the traditional networking model. It's possible to switch to some more efficient protocols specially

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 4– No.5, February 2016 – www.caeaccess.org

designed for mobile payment to achieve a better performance. Considering this factor, authors have proposed efficient payment protocol to achieve more security and reliability in our payment process.

Proposed model is **Self Certified Key Generation Payment Protocol**, in which they have considered following entities playing crucial role [28] as shown in figure 3.

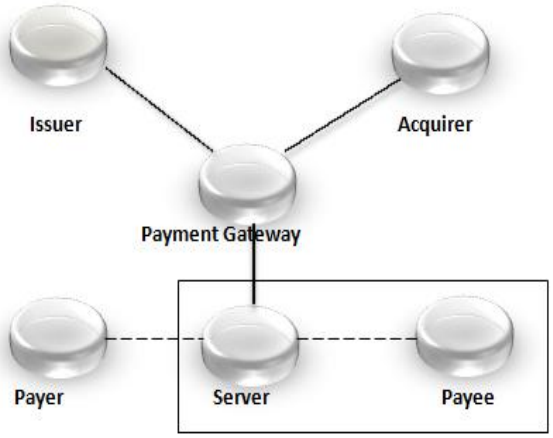

Fig 3 : Operational model for proposed protocol

Payer:
A payer is the one who wish to transfer money to payee. In proposed protocol, the payer is an entity equipped with an Application Unit. In this case Application Unit is Mobile phone in which Square Cash application is installed.

Server:
Server is an entity to whom payer will contact to make fund transfer. This entity could be a computational one such as a normal web server. In this case, it is square cash web server.

Payee:
A payee is the one who will receive money from payer. The payee is also equipped with an Application Unit. Here, Application Unit is Mobile phone in which Square Cash application is installed. Here, payee and server is considered as one unit.

Acquirer:
It is payee's financial institution. It verifies the validity of the deposited payment instrument and manages fund transfer from payee's end.

Issuer:
It is the payer's financial institution. It provides electronic payment instruments to the client to use in a payment and manages fund transfer on payer's end.

Payment gateway:
It is an additional entity that acts as a medium between acquirer/issuer at banking private network side

5.1 Self-certified key generation

In proposed protocol, some cryptographic techniques have been used to provide payment security and enhance efficiency. A self-certified public key provides benefits of certificate-based and identity-based public key cryptosystems.

As per Diffie–Hellman assumption, Internal mechanism in user's end device generates a non-singular elliptic curve E defined over a finite field Fq of characteristic p such a that Fq is used with base point generator P of prime order n. It chooses a key pair (S,Pk), where **Pk = SP**. The related parameter P is public while S is kept secret [13].

5.2 Private key generation

Payer chooses a random number N1 and computes N such that

N= H1 (ID, N1) P (1)
ID, N1 are send to SA over secure channel.

Payee chooses a random number after receiving this message called N2 and calculates R as a witness payer.

R = N + N2P (2)
It computes partial private key X as follows

X = H2 (ID,R)S + N2 (3)
Finally Server returns R,X to user over secure channel using which Secret Key Sk can be calculated

Sk = X + H1(ID,N1) (4)

5.3 Public key extraction from available parameters

Sk = X + H1(ID,N1)

Considered equation number (4)

SkP = XP + H1(ID,N1)P

Multiplied by P both sides

SkP = H2(ID,R)SP + N2P + H1(ID,N1)P

As per equation number (3), put value of X

SkP = H2(ID, R)SP + N2P +N

As per equation number (1), N= H1(ID,N1)P
SkP = H2(ID, R)SP + R

As per equation number (2), N+ N2P = R
SkP = H2(ID, R) Pk + R

As Pk = SP

Pk = (SkP –R)/ H2 (ID, R)

So user can compute the corresponding public key using above Equation, once they receive the value of witness R and Private key Sk. Refer Table 1 containing notations used in proposed protocol.

Table 1: Notations used in proposed protocol

Notation	Description
ID	Identity of payer
Pk	public key
R	Witness
Sk	Private key
X	Partial private key
Hi(,)	One-way hash function
N1	Random number entered by payer
N2	Random number entered by payee

5.4 The proposed mobile payment protocol

Encrypted messages are exchanged among the entities along with the symmetric keys. These keys are generated by self-certified key agreement protocol. After registration, all these entities own a pair of public/private key which is updated at the beginning of every payment transaction to ensure freshness of the keys [13].

In this section, they have described the execution of the proposed electronic payment protocol using self-certified public key by which payer transfers funds to payee. During the payment transaction, payer does not communicate directly with payee. Thus, Server and payment gateway plays the role of proxy to transmit authentication messages from payer to payee.

Flow of messages within entities

1) Payer → Server
2) Server → Payer
3) Payer → Server
4) Server → Payment Gateway
5) Payment Gateway → Issuer
6) Payment Gateway → Acquirer
7) Acquirer, Issuer → Payment Gateway
8) Payment Gateway → Server
9) Server → Payer

- Payer exchanges necessary information with server like payer ID, details about payee and witness R, parameter P (generated using Diffie–Hellman assumption) and request for payee's ID to initiate transaction. Here, transaction request has been created by payer. To initiate this request payer has to choose random number which will be used to compute payer ID.
- Server checks if the payer and payee is authorized user or not and accordingly responds to transaction request made by payer with required details like payee ID. It also calculates witness R using random number selected by payee. Here, server and payee are considered as a single unit, so communication between payee and server is not explained in detail.
- Once payer receives witness R and other required details in response of transaction request, it calculates private key. To get authentication from financial institutes involved in this transaction, payer sends authentication request to server.
- As server does not have right to authenticate transaction request on behalf of financial institutes, it forwards this encrypted authorization request to payment gateway.
- Payment gateway further contacts Issuer and acquirer bank for deduction and transfer of amount from respective accounts. Payment gateway has all required parameters to calculate private and public keys, so as to decrypt messages.
- On response to these authentication requests, payment gateway sends encrypted response which is recovered by server and later by payer. This can be seen in figure 4.

5.5 Advantages of Proposed Protocol

5.5.1. Confidentiality

Payer's and Payee's real identity is only revealed to server, so user anonymity is maintained throughout transaction.

5.5.2. Impartial Transaction

Introduction of payment gateway and server makes sure transaction takes place secure and fair way possible.

5.5.3. Safety from Pseudo Message Interference

In this protocol, safety from insertion of fake messages has been maintained, as this protocol is safe with encrypted messages.

5.5.4. Safety from Impersonation Attack

If anyone tries to impersonate payee and cheat payment gateway, private and public key generated directly from user's input. Also fresh keys are generated before every transaction to make sure security.

5.5.5. Preventing from Overspending

In proposed protocol with participation of issuer and acquirer bank, Payment gateway can make fund transfer very conveniently without deducting or depositing amount more than once in given single transaction.

6. CONCLUSION

In this paper, we have done security analysis of social payment application square cash on various factors. It has been observed that Square Cash is secured and reliable application. However, to increase level of security, we have introduced new payment protocol – Self certified key generation payment protocol. Using this self certified key generation protocol, user can send or receive money without any involvement of third party for key generation which reduces various types of security attacks and also reduces communication cost. It provides user anonymity, fair exchange, payment security. So there will be highly safe environment during fund transfer which will result in increased number of transactions done using this application. This research paper will be surely useful in future developments in secured mobile payment applications. So there will be highly safe environment during fund transfer which will result in increased number of transactions done using this application. This research paper will be surely useful in future developments in secured mobile payment applications.

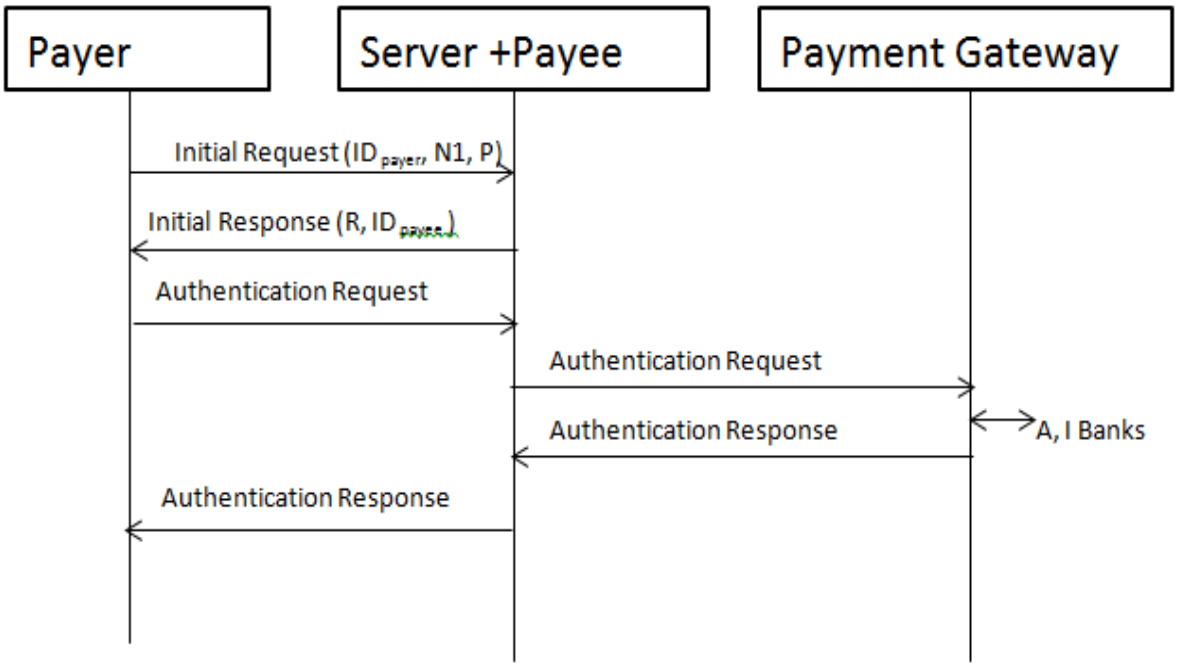

Fig 4: Message flow in proposed payment protocol

7. ACKNOWLEDGMENTS

This work is supported by Cleveland State University, Ohio of United States of America.

8. REFERENCES

[1] C. Kim, K. Changsu, T. Wang, S. Namchul, and K. Ki-Soo, "An empirical study of customers' perceptions of security and trust in e-payment systems," *Electron. Commer. Res. Appl.*, vol. 9, no. 1, pp. 84–95, 2010.

[2] Razaque, Abdul, and Khaled Elleithy. "Detection of Attacks for Restoring Privacy of Users to Improve Mobile Collaborative Learning (MCL) Over Heterogeneous Network." Networked Digital Technologies. Springer Berlin Heidelberg, 2012. 201-216.

[3] M. Niranjanamurthy and D. Chahar, "The study of e-commerce security issues and solutions," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 7, 2013.

[4] Razaque, Abdul, and Khaled Elleithy. "Interactive Prototypes to Foster Pedagogical Activities for Mobile Collaborative Learning Environment (MCLE)." International Journal of Interactive Mobile Technologies (iJIM) 6.1 (2012): 16-24.

[5] A. K. Jain and S. Devendra, "Addressing Security and Privacy Risks in Mobile Applications," *IT Prof.*, vol. 14, no. 5, pp. 28–33, 2012.

[6] "有关 XcodeGhost 的问题和解答," *Apple (中国)*. [Online].Available:http://www.apple.com/cn/xcodeghost/

[7] A. K. Ghosh and T. M. Swaminatha, "Software security and privacy risks in mobile e-commerce," *Commun. ACM*, vol. 44, no. 2, pp. 51–57, 2001.

[8] R. A. Botha, S. M. Furnell, and N. L. Clarke, "From desktop to mobile: Examining the security experience," *Comput. Secur.*, vol. 28, no. 3–4, pp. 130–137, 2009.

[9] P. S. Maan and M. Sharma, "Social Engineering: A Partial Technical Attack," *International Journal of Computer Science*, vol. 9, no. 2, 2012.

[10] B. Kraft, E. Mannes, and J. Moldow, "Security Research of a Social Payment App." 2014.

[11] Razaque, Abdul, and Khaled Elleithy. "Multi-frame Signature-cum Anomaly-based Intrusion Detection Systems (MSAIDS) to Protect Privacy of Users over Mobile Collaborative Learning (MCL)." arXiv preprint arXiv:1210.2030 (2012).

[12] K. Sahadevaiah, S. Kuncha, and P. R. P.V.G.D., "Impact of Security Attacks on a New Security Protocol for Mobile Ad Hoc Networks," Network Protocols and Algorithms, vol. 3, no. 4, 2011.

[13] W. Li, L. Wenmin, W. Qiaoyan, S. Qi, and J. Zhengping, "An efficient and secure mobile payment protocol for restricted connectivity scenarios in vehicular ad hoc network," Comput. Commun., vol. 35, no. 2, pp. 188–195, 2012.

[14] V. Rastogi, Y. Chen, and X. Jiang, "Droidchameleon: evaluating android anti-malware against transformation attacks," Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 329–334, 2013.

[15] pxb, "pxb1988/dex2jar," GitHub. [Online]. Available: https://github.com/pxb1988/dex2jar. [Accessed: 13-Oct-2015].

[16] E. Dupuy, "Java Decompiler." [Online]. Available: http://jd.benow.ca/. [Accessed: 13-Oct-2015].

[17] D. Low and L. Douglas, "Protecting Java code via code

obfuscation," Crossroads, vol. 4, no. 3, pp. 21–23, 1998.

[18] "How does Square Cash work technically? - Quora." [Online]. Available: https://www.quora.com/How-does-Square-Cash-work-technically. [Accessed: 15-Oct-2015].

[19] E. Rescorla, SSL and TLS: Designing and Building Secure Systems. Addison-Wesley Professional, 2001.

[20] F. Callegati, C. Franco, C. Walter, and R. Marco, "Man-in-the-Middle Attack to the HTTPS Protocol," IEEE Security & Privacy Magazine, vol. 7, no. 1, pp. 78–81, 2009.

[21] Praetorian, "Man-in-the-Middle TLS Protocol Downgrade Attack," Praetorian. [Online]. Available: https://www.praetorian.com/blog/man-in-the-middle-tls-ssl-protocol-downgrade-attack. [Accessed: 16-Oct-2015].

[22] "Dnsmasq - network services for small networks." [Online].Available:http://www.thekelleys.org.uk/dnsmasq/doc.html. [Accessed: 15-Oct-2015].

[23] Documentation Group, "Welcome! - The Apache HTTP Server Project." [Online]. Available: https://httpd.apache.org/. [Accessed: 15-Oct-2015].

[24] "Security Tips | Android Developers." [Online]. Available: http://developer.android.com/training/articles/security-tips.html. [Accessed: 16-Oct-2015].

[25] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, "Google Android: A Comprehensive Security Assessment," IEEE Secur. Priv., no. 2, 2010.

[26] "Checking Device Compatibility with SafetyNet | Android Developers." [Online]. Available: https://developer.android.com/training/safetynet/index.html. [Accessed: 16-Oct-2015].

[27] M. Bezuidenhout, B. Monique, M. Francois, and H. S. Venter, "Social engineering attack detection model: SEADM," in 2010 Information Security for South Africa, 2010.

[28] Isaac, Jesus Tellez, Jose Sierra Camara, Sherali Zeadally, and Joaquin Torres Marquez. "A secure vehicle-to-roadside communication payment protocol in vehicular ad hoc networks." Computer Communications 31, no. 10(2008):2478-2484.