A Lightweight Architecture for Secure Two-Party Mobile Payment

Y. Zhu and J. E. Rice
Dept. of Math & Computer Science
University of Lethbridge
4401 University Dr. W., Lethbridge, AB, Canada
{yunpu.zhu, j.rice}@uleth.ca

Abstract

The evolution of wireless networks and mobile devices has resulted in increased concerns about performance and security of mobile payment systems. In this paper we propose SA2pMP, a lightweight secured architecture for two-party mobile payments. SA2pMP employs a lightweight cryptography scheme that combines public key and symmetric key cryptography systems (ECDSA and AES), as well as a multi-factor authentication mechanism. These are coupled with a transaction log strategy to satisfy the properties of confidentiality, authentication, integrity and non-repudiation. We simulate SA2pMP in a context of money transfer banking transaction, on three different emulators: Sun Java Wireless Toolkit 2.5.2 for CLDC emulator, Sony Ericsson SDK 2.5.0.3 Z800 emulator, and Nokia S60 3rd Edition emulator. We also compare SA2pMP to some existing mobile payment platforms. The result of simulation and comparison proves that SA2pMP is a lightweight secured mechanism that is feasible and suitable for two-party mobile payment transactions, e.g. mobile banking, over Java ME enabled, resource-limited mobile devices.

1. Introduction

Wireless networks and mobile devices have been used widely in more areas than ever. As an important wireless network application in financial field mobile payment was predicted to possess a bright future in becoming a successful mobile service [18]. However, due to security problems, it has not become a major medium for payment. In this paper we propose a lightweight secured architecture for two party involved mobile payments (SA2pMP). Our architecture makes use of a lightweight cryptography scheme, a multi-factor authentication mechanism along with a transaction log strategy to ensure all security requirements are fulfilled. The simulations on Sun Java Wireless Toolkit 2.5.2 for CLDC emulator, Sony Ericsson SDK 2.5.0.3 Z800 emulator, and Nokia S60 3rd Edition emulator prove that SA2pMP can be feasibly implemented in Java ME enabled mobile devices. Compared to various existing platforms such as *JASA* [9], *LSM* [12], *SET* [28] and *iKP* [1], SA2pMP is lightweight and better suited to two-party mobile payment transactions over resource-limited mobile devices. This work continues the proposal suggested in [20].

The rest of paper is organized as follows: Background information will be introduced in section 2. In section 3, SA2pMP's network module, security mechanism and key management will be described. Then in section 4, the implementation will be introduced in a context of mobile banking. The simulation will be described in section 5. Section 6 will analyze the result of simulation, and make evaluations on time delay and code size. In section 3, SA2pMP will be compared to some other existing mobile payment platforms. We conclude this paper in section 8, and raise issues for future work.

2. Background

Background information in this area is benefit in understanding the following sections. In our research, a mobile device is defined as a handheld device with internet browsing capability and other basic computational capabilities. A mobile device can be viewed as an identifier for a particular individual as each individual generally has one's own mobile device which is not usually shared with others. Currently, numerous mobile devices accept the Java ME standard. A Java ME standard at least includes both a configuration like the Connected Limited Device Configuration (CLDC) [23] and a profile like the Mobile Information Device Profile (MIDP) [24]. Optional packages provide additionally capability in specific areas of functionality.

Mobile payment can be defined as any payment transaction involving a mobile device [4]. We focus on account-based payment systems which can be mobile phone-based, smart card or credit card-based [7]. The parties involved in two-party mobile payment transactions are assumed to be a user and a financial service provider, normally a bank.

978-0-7695-3823-5/09 $26.00 © 2009 IEEE
DOI 10.1109/CSE.2009.364

IEEE computer society

We also assume that transactions take place over the mobile phone network [25].

The security architecture implemented on the application layer is independent of the lower layers' security protocols, and it does not require modifications to the current wireless network's infrastructure and protocols. Our architecture is proposed on the application layer.

To design a secured mobile application system, four security requirements should be satisfied [8]. These four requirements include:

- *Confidentiality*: The confidential information must be secured from an unauthorized party.
- *Authentication*: Authentication ensures two parties with right accessing to a system.
- *Integrity*: Systems must be guaranteed to have not been corrupted by outside parties.
- *Non-repudiation*: The user must not deny the performed transaction and must provide proof in case that this situation occurs.

3. Proposed Architecture

We propose a new lightweight secured architecture for two-party mobile payment namely SA2pMP. A mobile bank is employed as the context to describe our architecture. Adopted from [14], table 1 summarizes the requirements resulting from the security concerns, and technologies recommended to address them. The third column of Table 1 describes the specific solutions proposed in this paper for addressing each of the concerns.

Security	Technology	Solution
Authentication	Possession	mobile device
	Knowledge	PIN
	Property	userid/password
	Digital Signature	ECDSA
Integrity	Digital Signature	
Non-repudiation	Digital Signature	
	Log	Biz-Transaction Log
Confidentiality	Encryto/Decrypto	AES

Table 1. Security requirements for mobile transactions, along with the technologies recommended for these requirements and solutions to address them.

3.1. Network Module

In a banking transaction there are two parties involved: the user and the bank. The user taking part in our banking transaction is also the mobile device's holder and owner. As illustrated in Figure 1, the mobile device connects with a network gateway through mobile networks provided by a mobile network operator. Wired networks connect the banking system and the network gateway. Except for constraints in network bandwidth and mobile devices, this physical network architecture is transparent to the mobile banking platform.

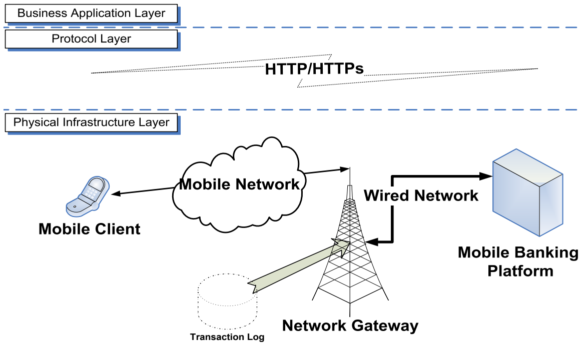


Figure 1. Network module for SA2pMP.

3.2. Security Mechanism

The security mechanism of SA2pMP mainly includes three parts: a lightweight cryptography scheme, a multiple-factor authentication strategy and a distributed transaction log strategy.

Lightweight Cryptography Scheme SA2pMP employs the Advanced Encryption Standard (AES) [2] to perform data's encryption and decryption. The AES with 256-bit key size is used. SA2pMP utilizes the Elliptic Curve Digital Signature Algorithm (ECDSA) [27] to mainly protect non-repudiation because of its low computational cost and short key size, both of which reduce the overhead in a wireless and resource constraint environment [30]. Specifically, elliptic curves are over the prime field F_p which is best for software applications [6]; the key size is 192 bits which has an equal security level to the Digital Signature Algorithm (DSA) with the key size of 1024 bits [13]. We use *ECDSAprime192* to name the ECDSA's implementation way. SA2pMP also makes use of SHA-1 [17] to calculate a *message digest* for producing the digital signature.

Figure 2 illustrates how cryptography algorithms processed in SA2pMP. The communication layer represents a public wireless network environment. The transaction message (*msg*) must be protected from third party eavesdropping in the communication layer, so we use the signature layer and the encryption layer to process *msg*. The plaintext message is signed before being encrypted. This method is named as "Sign-and-Encrypt" [3]. The digital signature layer is not intended to be independent of the encryption layer, therefore, we utilize an *added identifier* to cross these two layers. We conjunct the Subscriber Identifier Module number (SIM), the PHone IDentifier (PHID), and the user's bank ACcount number (ACID) to the *added identifier*: ID_C.

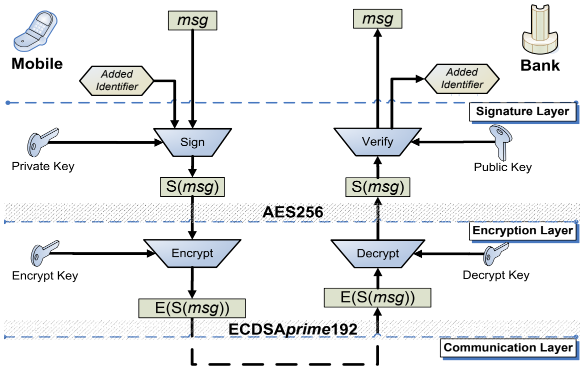


Figure 2. lightweight cryptography scheme for SA2pMP.

Multi-factor Authentication Strategy Authentication concerns that a communicating entity is the one it claims to be [22]. To meet recommendations in [5], SA2pMP provides a strong authentication. To satisfy this we chose these factors from [15]:

- *Something you have*: A mobile device, as well as a Personal Identification Number (PIN) offered by mobile network operators, is possessed by the user.
- *Something you know*: A password for transactions offered by banks is the knowledge of the user.
- *Something the user is or does*: Digital signature can be looked as a behavior of the user.

Transaction Log Strategy Along with the digital signature a distributed transaction log strategy is used to ensure non-repudiation. The transaction log server is a security mechanism to protect a bank from a false repudiation. If a user refuses to admit to participating in a mobile transaction, the transaction log server can provide the transaction records as proof. In Figure 1, the transaction log is distributively located in the network gateway which physically belongs to the wireless network operator. The wireless network operator is not the same business unit with the bank, in that it objectively takes the role as the third trusted auditor.

3.3. Key Management

Secure methods of key management are important to a secured mobile payment system. In practice most attacks on public key systems will be aimed at the key management [21]. In our architecture the two key pairs required are used both for digital signature, and encryption.

Digital Signature Key Management To keep the private key secret we propose to generate the key pair in the mobile device. A Key Generation function is called to generate a key pair when no valid key pair exists in the system. The keys must then be distributed and stored. The private key is stored in the mobile device, either using File-Store-in-JAR or Record-Store-in-RMS [20]. The public key is transferred to the authentication server in the bank side and then stored in a public key depository. This process can run off-line, not competing with other transaction or communication processing for computational resources. A renewal of a key pair must initiated by the banking server after the current key pair expires. Once the server detects that renewal is needed, a notice (such as SMS) must be sent to the mobile device to generate a new key pair.

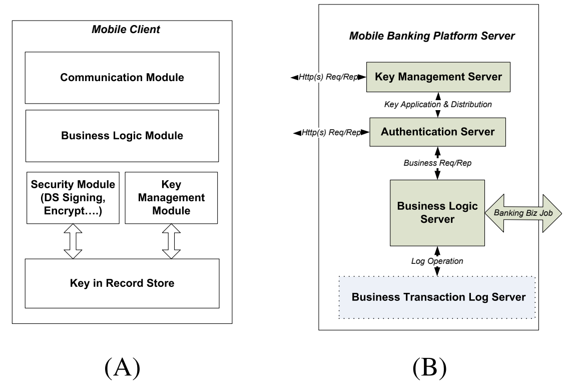
Encryption Key Management Generation of the encryption key takes place on the bank server. Encryption and decryption share the same secret key, which clearly cannot be transferred over the open wireless network due to the risk of interception. The encryption key is proposed to be stored in the program application jar package; then users would be issued the key along with the application package when they register for mobile banking services. The bank side's security measures are assumed to be sufficient.

4. Implementation

For space reasons, full implementation details cannot be included in this paper. The more detailed explanation of the bank modules, mobile client modules and server components can be referred in [20]. The architecture's implementation is divided into a mobile client and a banking server. Figures 3(A) and 3(B) illustrate the proposed designs for the client architecture and the server architecture for SA2pMP.

SA2pMP Mobile Client Architecture The client portion of the proposed SA2pMP is built on Java ME enabled mobile devices. As Figure 3(A) depicts, the mobile client system is composed of the four modules described below.

- Business Logic Module: The BLM is in charge of all particular business functions.
- Security Module: The SM is to be responsible for security issues.
- Communication Module: The CM is the module in charge of the network link.



(A) (B)

Figure 3. (A) Mobile client architecture for SA2pMP. (B) Mobile banking server architecture for SA2pMP.

- Key Management Module: The KMM is in charge of key management as described in Section 3.3.

SA2pMP Server Architecture Figure 3B) illustrates the proposed architecture for the MBP server, which consists of the components described below.

- Key Management Server: KMS deals with maintaining the digital signature's public key and the encryption key, as well as initiating messages notifying the user the renew the key pair currently in use.
- Authentication Server: The AS is in charge of authentication service.
- Business Logic Server: The BLS handles all legal business requests.
- Transaction Log Server: The TLS maintains the log files for banking transactions.

5. Simulation

Our simulation operated on an IBM IntelliStation M Pro PC, with Pentium 4 CPU 2.80GHz and 2GB RAM. The operating system is Windows XP Professional SP3. The implementation of cryptographical algorithms makes use of a third party cryptography API provider namely *Bouncy Castle*. SA2pMP is evaluated on three different mobile device emulator platforms: Sun Java Wireless Toolkit 2.5.2 for CLDC emulator, Nokia S60 3rd Edition emulator and Sony Ericsson SDK 2.5.0.3 Z800 emulator. All these three platforms support CLDC-1.1 and MIDP-2.0.

Transaction Process The simulation is designed for a mobile banking context with a money transfer transaction. In the following equations, *Client* communicates with (*Bank*). PWD denotes the password for accessing *Bank*.

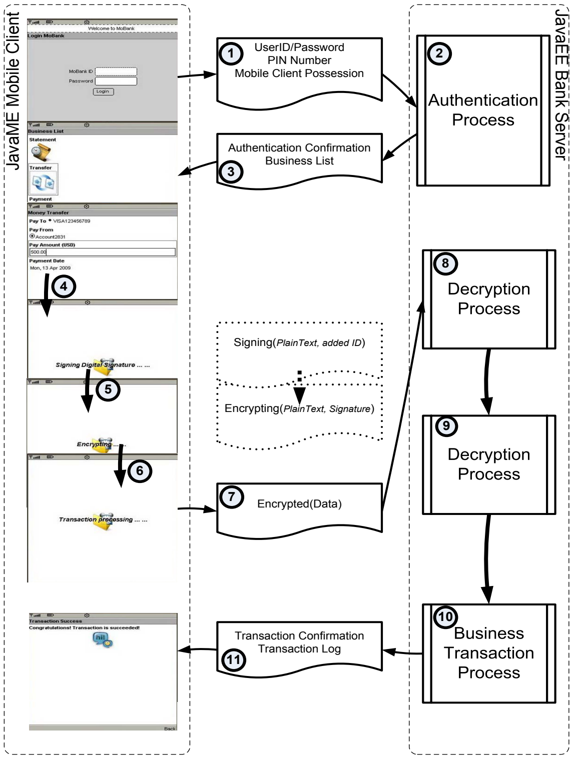


Figure 4. Secured transaction process.

$Encrypt$ and $Decrypt$ denote the encryption and decryption operations. $Verify$ denotes the process of authentication verification, while $DVerify$ denotes verifying the digital signature. *AuthConfirm* denotes an authentication confirmation, while $BizConfirm$ denotes a business confirmation. $BizTransaction$ denotes a business transaction process. *Bizlist* denotes the authorized business transaction list. *TD* denotes transaction data. *DS* denotes a digital signature. RK_S and PK_S denotes the private key and the public key for generating the digital signature, while K_E denotes the encryption key. The *h* is a hash function, and $crypto\_msg$ is an encrypted message. $TLog$ is an activity recording transaction log on the network $Gateway$. As Figure 4 illustrates, a secured transaction is processed as follows.

1. $Client$ sends an authentication request to $Bank$.

$$Client- > Bank : (ACID, PWD, ID_C) \tag{1}$$

2. $Bank$ verifies the authentication request.

$$Bank : Verify(ACID, PWD, ID_C) \tag{2}$$

3. $Bank$ responds a confirmation to $Client$.

$$Bank- > Client : (AuthConfirm, Bizlist) \tag{3}$$

4. $Client$ generates a transaction message.

$$Client : msg < -TD \tag{4}$$

5. $Client$ computes a digital signature.

$$Client : DS = DSign(ID_C, h(msg), RK_S) \tag{5}$$

6. $Client$ encrypts the signed message.

$$Client : crypto\_msg = Encrypt(DS + msg, K_E) \tag{6}$$

7. $Client$ sends an encrypted message to $Bank$.

$$Client- > Bank : crypto\_msg \tag{7}$$

8. $Bank$ decrypts encrypted message.

$$Bank : DS + msg = Decrypt(crypto\_msg, K_E) \tag{8}$$

9. $Bank$ verifies the digital signature.

$$Bank : Yes/No = DVerify(DS, h(msg), PK_S) \tag{9}$$

10. If the step responds true, the business transaction is started.

$$Bank : BizTransaction \tag{10}$$

11. $Bank$ responds a transaction confirmation to $Client$, and records a transaction log.

$$Bank- > Client : BizConfirm \tag{11}$$

$$Gateway : TLog \tag{12}$$

Data Transformation In a money transfer transaction, some information needs to be recorded. $FAcID$ refers the bank account number from which the money will be transferred. $TAcID$ denotes the bank account number to which the money is transferred. Amt denotes the money amount. $Time$ denotes the time when money transfer transaction is initiated. After the user submits the business record information, a string *RawText* connects four parameters together by a "&"sign.

$$RawText = FAcID + TAcID + Amt + Time \tag{13}$$

Then *RawText* is added with ID_C. In the simulated money transfer transaction, *PText* has the length of 312 bits. The digital signature is computed from *PText*. The signed message *TempText* will be followed by a symmetric encryption operation. *EnText* is ready to be sent to the bank server.

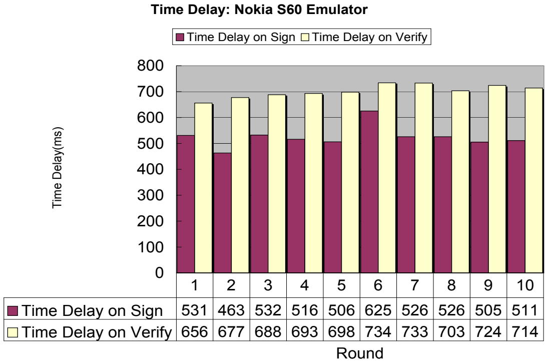


Figure 5. Time delay on Nokia Emulator (ms).

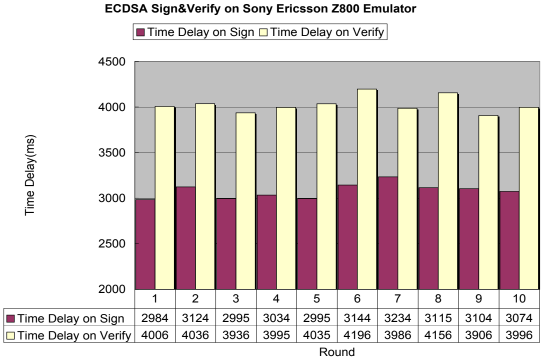


Figure 6. Time delay on Sony Ericsson Emulator (ms).

$$PText = RawText + ID_C \tag{14}$$

$$TempText = PText + DSign(h(PText), RK_S) \tag{15}$$

$$EnText = Encrypt(TempText) \tag{16}$$

We run the same MIDlet suite in different emulators to measure the time delay caused by cryptographical operation for the goal of security. Figure 5, 6, 7 illustrate the time cost on these three different emulator platforms.

6. Evaluation

To evaluate the feasibility of SA2pMp, we consider the processing time delay for the cryptographical operations and the JAR-file size of the MIDlet suite in the mobile device.

6.1. Time Delay Evaluation

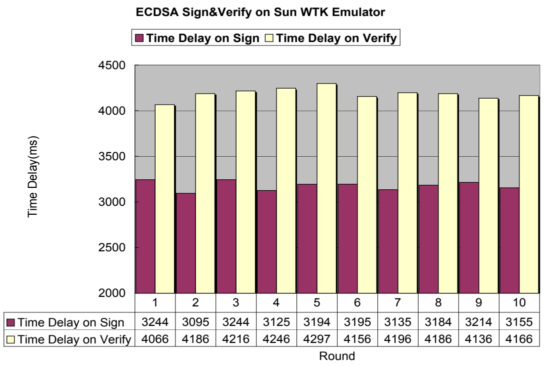


Figure 7. Time delay on Sun WTK Emulator (ms).

Time Delay Criteria To make an evaluation on time delay, we need to discuss the time delay criteria. Although the response should be as quick as possible, the basic advice regarding response time delay was categorized in [16].

- 100 milliseconds is the time limit for users to feel that the system is reacting instantaneously.
- 1,000 milliseconds is the time limit for users who thought the system stayed uninterrupted.
- 10,000 milliseconds is the time limit for keeping the user's attention focused.

Time Delay Evaluation Method Considering the business system employing our secured architecture, the total time delay *T* can be evaluated as:

$$T = T_{CBiz} + T_{CSec} + T_{Com} + T_{SSec} + T_{SBiz} \qquad (17)$$

T_{CBiz} denotes the time delay for the business computation on a mobile client. T_{CSec} denotes the time delay for the goal of security. Almost all of the security time delay is cost by cryptographical computation, including digital signature sign and encryption. T_{Com} denotes the time delay on network communication. T_{SSec} denotes the time delay cost by security operation on the server side. T_{SBiz} denotes the time delay the business transaction. As the bank server is built on a rich-resource computer server platform, the computational speed is quite fast, and the transaction log is written simultaneously, so we consider the operation on bank server side to be instantaneous. As T_{CBiz} and T_{Com} are not spent in the security module, we do not consider them in evaluating extra time delay for security. To consider the extra time delay for the goal of security, we use the parameter of T_{CSec}. Referred by equation 15 and equation 16, T_{CSec} can be illustrated as:

$$\begin{aligned} T_{CSec} &= T_{encrypt(DSign(msg)+msg)} \\ &= T_{encrypt} + T_{DSign} + T_{+} \end{aligned} \qquad (18)$$

In our simulation, the time delay for AES256 encryption or decryption is less than 1 millisecond, so we consider encryption to operate instantaneously, $T_{encrypt} \approx 0$. The time delay T_{+} for string connecting operation is considered to be instantaneous as well. The digital signature's operation is largest time cost for security, and so we use T_{DSign} to evaluate T_{CSec}.

Time Delay Evaluation Result Table 2 calculates the average time delay on ECDSA sign and verification in milliseconds on the three different emulator platforms. The simulation on Nokia S60 is most positive. The average time delay of signing on Nokia S60 is 524.1 ms, which is much better than the data on Sony Ericsson Z800 (3080.3 ms) and on Sun WTK QwertyDevice (3178.5 ms). The same situation applies to verification. Verification costs more time than signing. Following the time delay criteria proposed in [16], the time delay on Nokia S60 is less than 1,000 milliseconds, then the system appears to stay uninterrupted to users. The time delay on Sony Ericsson Z800 or WTK QwertyDevice is between 1,000 milliseconds and 10,000 milliseconds, therefore, the system still keeps users' attention focused although they notice a delay.

	Nokia	Sony Ericsson	Sun WTK
Sign	524.1	3080.3	3178.5
Verify	702	4024.8	4185.1

Table 2. Average time delay (ms).

Though our software application is running on JVM, some mobile devices have more efficient implementations. Seeing from our experiment, the Symbian based Nokia S60 has more efficient implementation than Sony Ericsson Z800. Neglecting the difference in platform, we roughly compared our simulations to other ECDSA's Java ME implementations [26] [29] [11]. The results from [26] [29] suggests that ECDSA could be performed faster, while the emulation data of [11] is less efficient than our data on Nokia S60 emulator.

By evaluating extra time delay for security, SA2pMP is feasible for implementing on mobile devices.

6.2. Code Size Evaluation

A MIDlet suite with an API library normally has a large JAR-file. However, mobile devices usually have their own size limitation for implementing MIDlet suites. For example, CLDC 1.0 specifies that the minimum base memory available for the Java platform is 160KB [10]. Furthermore, the large JAR size may cause program be initiated or the application be installed over-the-air (OTA) slowly. As implementing SA2pMP only uses a small part of API, the JAR-file size will be shrink if the unused code of Bouncy Castle API library is not contained in MIDlet suites. Obfuscation can be used to effectively reduce the size of Java class [19].

In our simulation, the mobile client application, *MoBankClient.jar* has a size of 130 KB, which is feasible for a CLDC 1.0 platform. In this JAR-file the security package is only a small part (the rest part of JAR-file is business package, the size of which is depended on the complexity of business transaction). That means our API library meets the requirement for code size and is feasible to be implemented on CLDC 1.0 mobile devices.

7. Comparison

We provide a comparison with some existing security mobile payment architectures. *JASA* provides end-to-end client authentication, data confidentiality and integrity; however, it can not guarantee non-repudiation. *SET* and *iKP* are two credit-card payment protocols which were designed originally for electronic commerce. Although they are implemented successfully for e-commerce on wired network, they are too heavy-loaded to operate in resource-limited environments. *LSM* utilizes a wireless protocol gateway to meet the four security requirements of mobile commerce. However, the security gap lying on the wireless protocol gateway harms that the cryptography key pair need to be protected from any third party interference. Table 3 illustrates our comparison between *JASA*, *SET* and *iKP*, *LSM* and SA2pMP.

JASA utilizes AES, which only fulfills the security requirements for authentication, integrity and confidentiality, but not non-repudiation. *LSM*, *SET* and *iKP* employ RSA, which is too heavyweight in computation for a resource-limited mobile device. SA2pMP designed a light scheme combining both AES and ECDSA.

Regarding with the authentication strategy, *JASA*, *LSM*, *SET* and *iKP* utilize single-factor authentication, which is not enough to secure a financial application. SA2pMP implements a multi-factor authentication strategy, which protects a stronger security in authentication.

From the perspective of computational requirements, *JASA* has the advantage in computational requirements as it only implements AES. *LSM* utilizes the wireless network gateway as the agent of the mobile device. On mobile devices portion of *LSM* computational requirement is lightweight. *SET* and *iKP* are not originally designed for mobile payment, thus for mobile devices the computational requirement is heavyweight. SA2pMP balances the security and the computational complexity. The light cryptography scheme is lightweight in computational requirement without sacrificing security protection.

	JASA	*SET & iKP*	*LSM*	SA2pMP
M-Payment	√	No	√	√
2-Party	√	(1)KP	√	√
Crypto- Algorithm	AES	RSA	RSA	ECDSA AES
Authent- -ication	Single -Factor	Single -Factor	Single -Factor	Multi -Factor
Non- Repudiation	No	√	√	√
Java ME	√	-	-	√
Computation Requirement	Light weight	Heavy weight	Light weight	Light weight
3rd-Party involved	Low	High	High	Medium

Table 3. Comparison.

Non-repudiation is a key requirement for a comprehensive security. In all architectures listed in Table 3, only *JASA* did not suggest approaches to solve the false repudiation problem. Besides a digital signatures employment, SA2pMP utilizes a distributed transaction log strategy to provide a defensive approach to keep non-repudiation.

Although inter-enterprise collaboration is prevalent, no enterprise wants the third party be involved overmuch. *LSM* stored its cryptography key pair in the network protocol gateway, which leads an excessive involvement of third party. *SET* and *iKP* require a high third-party involvement, as they rely on the trusted third party to offer the key pairs. As *JASA* does not employ any third party in its architecture, it had a low third-party involvement. SA2pMP utilizes the network gateway in its distributed transaction log strategy. The approach avoiding the third partys excessive involvement; meanwhile, it contributes the non-repudiations protecting.

Moreover, both *JASA* and SA2pMP target Java ME platform, while *LSM*, *SET* and (iKP) are not targeted to any platform. All of them can be used for mobile payment and for two-party mobile payment transaction.

8. Conclusion and future work

We have proposed a lightweight secured architecture for two-party mobile payments. While other architectures and protocols have been proposed, they either are not well suited for resource-limited mobile devices or they do not satisfy all of the parties' concerns regarding security on mobile transactions. Our architecture, named as SA2pMP, is implemented in a Java ME enabled mobile client, with a mobile banking server supporting it, likely implemented in Java EE. The proposed architecture employs a lightweight cryptography mechanism combining symmetric and public key algorithms (AES and ECDSA), a multi-factor authentication strategy utilizing mobile devices' natural benefits, and a distributed transaction log strategy to meet the security requirements of integrity, authentication, confidentiality and non-repudiation. Through the simulations based on three different brands of mobile devices' emulators, we show SA2pMP is feasible to be implemented in some mobile devices without unacceptable delay. Compared to some other existing mobile payment platforms, SA2pMP is feasible and well-suited to protect security for two-party payment transaction over resource-limited mobile devices.

The future work will be focused on promoting operation speed of cryptographical algorithms. The simulations will be carried out on real mobile devices.

References

[1] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, E. V. Herreweghen, and M. Waidner. Design, implementation, and deployment of the iKP secure electronic payment system. *Selected Areas in Communications, IEEE Journal*, 18:611–627, April 2000.

[2] J. Daemen and V. Rijmen. The block cipher rijndael. *Smart Card. Research and Applications*, LNCS1820:277–284, December 2006.

[3] D. Davis. Defective sign & encrypt in S/MIME, PKCS#7,MOSS, PEM, PGP, and XML. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 65–78, June 2001.

[4] P. C. Deans. *E-Commerce and M-Commerce Technologies*. IRM Press, 2004.

[5] Federal Financial Institutions Examination Council. *Authentication in an Internet Banking Environment*.

[6] A. Fernandes. Elliptic curve cryptography. *Dr. Dobb's Journal*, December 1999.

[7] J. Gao, J. Cai, K. Patel, and S. Shim. A wireless payment system. In *Embedded Software and Systems, 2005. Second International Conference on*, volume 16-18, page 8, 2005.

[8] W. C. Hu, C.-W. Lee, and W. Kou. *Advances in Security and Payment Method in Mobile Commerce*. Idea Group *Inc.*, 2004.

[9] W. Itani and A. Kayssi. J2ME application-layer end-to-end security for m-commerce. *Journal of Network and Computer Applications*, 27:13–32, 2004.

[10] Java Community Process. JSR 139: Connected limited device configuration 1.1. http://www.jcp.org/en/jsr/detail?id=139.

[11] M. Kilas. Digital signatures on nfc tags. Master's thesis, KTH Royal Institute of Technology, Sweden, March 2009.

[12] K.-Y. Lam, S.-L. Chung, M. Gu, and J.-G. Sun. Lightweight security for mobile commerce transactions. *Computer Communications*, 26:2052–2060, 2003.

[13] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Lecture Notes in Computer Science*, 1751:446–465, 1999.

[14] K. Linck, K. Pousttchi, and D. Wiedemann. Security issues in mobile payment from the customer viewpoint. In *Proceedings of the 14th European Conference on Information Systems(ECIS 2006)*, Ljungberg, 2006.

[15] E. Maiwarld. *Fundamentals of Network Security*. McGraw-Hill Professional, 2004.

[16] J. Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[17] NIST. FIPS PUB 180-3: Federal Information Processing Standards Publication, Secure Hash Standard (SHS). Technical report, National Institute of Standards and Technology, October 2008.

[18] J. Ondrus and Y. Pigneur. A disruption analysis in the mobile payment market. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 3 p. 84c*, September 2005.

[19] C. E. Ortiz. Obfuscating your midlet suite. http://developers.sun.com/mobility/midp/ttips/midletsize/, October 2009.

[20] J. E. Rice and Y. Zhu. A proposed architecture for secure two-party mobile payment. In *2009 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, August 2009.

[21] RSA Laboratories. *RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1*. RSA Security Inc., 2000.

[22] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson Prentice Hall, third edition edition, 2006.

[23] Sun Microsystems. Connected Limited Device Configuration (CLDC): JSR 30, JSR 139 Overview. http://java.sun.com/products/cldc/overview.html.

[24] Sun Microsystems. Mobile Information Device Profile (MIDP): JSR 37, JSR 118 Overview. http://java.sun.com/products/midp/overview.html.

[25] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.

[26] S. Tillich and J. Großschadl. A survey of public-key cryptography on J2ME-enabled mobile devices. *Lecture notes in computer science*, pages 935–944.

[27] S. Vanstone. Responses to NISTs Proposal. *Communications of the ACM*, 35:50–52, July 1992.

[28] VISA & Mastercard. *SET Secure Electronic Transaction Specification*, 1997.

[29] J. Zheng, Z. Shao, S. Huang, and T. Yu. Security of two signature schemes based on two hard problems. In *Communication Technology, ICCT 2008. 11th IEEE International Conference on*, pages 745 – 748, Nov 2008.

[30] X. Zhong, D. Guanzhong, and Y. Deming. An efficient ECDSA-based signature scheme for wireless networks. *Wuhan University Journal of Natural Sciences*, 11(6):1707–1710, November 2006.