International Journal of Internet, Broadcasting and Communication Vol.8 No.1 7-18 (2016)
http://dx.doi.org/10.7236/IJIBC2016.8.1.7

IJIBC 16-1-2

Per-transaction Shared Key Scheme to Improve Security on Smart Payment System

Fawad Ahmad[1], Younchan Jung[2+]

School of Information, Communications and Electronics Engineering,
The Catholic University of Korea
fawadahmad324@gmail.com[1], ycjung@catholic.ac.kr[2+]

Abstract

Several authentication methods have been developed to make use of tokens in the mobile networks and smart payment systems. Token used in smart payment system is genearated in place of Primary Account Number. The use of token in each payment transaction is advantageous because the token authentication prevents enemy from intercepting credit card number over the network. Existing token authentication methods work together with the cryptogram, which is computed using the shared key that is provisioned by the token service provider. Long lifetime and repeated use of shared key cause potential brawback related to its vulnerability against the brute-force attack. This paper proposes a per-transaction shared key mechanism, where the per-transaction key is agreed between the mobile device and token service provider for each smart payment transaction. From server viewpoint, per-transaction key list is easy to handle because the per-transaction key has short lifetime below a couple of seconds and the server does not need to maintain the state for the mobile device. We analyze the optimum size of the per-transaction shared key which satisfy the requirements for transaction latency and security strength for secure payment transactions.

Keywords: *token authentication; smart payment system; per-transaction key; mobile security; secure transaction.*

1. Introduction

Subscriber Identity Module (SIM) has evolved to Universal Subscriber Identity Module (USIM), which are responsible for stable and secure solutions making mobile phones and quality vendors reliable and protected. The main reason that GSM uses SIM authentication is to securely verify the user identity before allowing it to access the network. One of the potential vulnerabilities of GSM is the fact that GSM Authentication is limited to user authentication. UMTS removes this potential flaw by introducing user and network authentication together. In contrast with GSM, the new parameters in UMTS are the Authentication Token (AUTN) to support mutual authentication, management and prevent replay attacks, as well as the key for integrity protection. Also, token authentication has been applied to smart payment systems through mobile phones. Smart payment brings cashless and card-less payment which replaces the physical wallet with a mobile wallet. It allows paying through Near Field Communication (NFC) enabled mobile device at

Manuscript Received: Nov. 12, 2015 / Revised: Nov. 24, 2015 / Accepted: Dec. 21, 2015
Corresponding Author: ycjung@catholic.ac.kr
Tel: +82-2-2164-4364, Fax: +82-2-2164-4777
School of Information, Communications and Electronics Engineering, The Catholic University of Korea

contactless Point-of-Sale (POS) terminals. The mobile device and the POS terminal communicate with each other through NFC [1]. Smart payment uses dynamic-data transactions where the mobile device generates a cryptogram, which authenticates a valid payment token.

The current smart payment systems use the tokenization techniques to ensure the privacy and security of the important card information. Tokenization refers to the substitution of sensitive card data with a less sensitive random number called tokenized PAN [2]. Token service provider, which is issuer or payment network, generates token and maintains mapping information of token and PAN in the token vault. The token is then provisioned to mobile device. The issuer also provisions the shared key, which is used to make the cryptogram, to the mobile device. The token mapped to the credit card number avoids the credit card number to be stored in the merchant side and transmitted over the payment network. Therefore tokenization techniques prevent cross-channel fraud (credit card number hacked at NFC channel and used in static-data transactions) [2]. Even though the fraudsters hack the token, it is useless because they confront with the difficulty to decrypt the relating cryptogram which is a transaction-unique value and is computed by encrypting the nonce and token with the shared key [3-5].

As shown in Fig.1, when it comes to making a payment, the merchant side sends the nonce to mobile device in order that the nonce may be used as transaction identity. Based on the nonce the mobile device generates a cryptogram with a shared key, which is sent along with the token to the tokenization server via the payment network. After the token service provider verifies the token using the shared key, it can authorize the payment transaction. This paper suggests the solutions against the possible attacks in the smart payment system, which includes intrusion and brute-force attacks against the shared key, which resides in the mobile device and token service provider for a long time and is used in every payment transaction. The key shared between the mobile device and token service provider is used in a valid cryptogram generation and token authentication process. Payment transaction is authorized after token has been authenticated by the token service provider through the cryptogram associated with it. The existing smart payment systems use the same shared key in every transaction for cryptogram generation. The long lifetime and frequent use of the shared key makes it vulnerable against the intrusion attack as well as easy to be compromised against the brute-force attack.

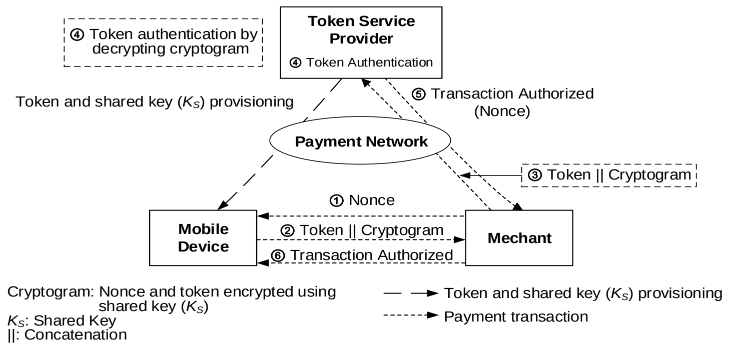
④ Token authentication by decrypting cryptogram
Token Service Provider
④ Token Authentication
Token and shared key (K_S) provisioning
⑤ Transaction Authorized (Nonce)
Payment Network
③ Token || Cryptogram
① Nonce
Mobile Device
② Token || Cryptogram
Mechant
⑥ Transaction Authorized
Cryptogram: Nonce and token encrypted using shared key (K_S)
K_S: Shared Key
||: Concatenation
Token and shared key (K_S) provisioning
Payment transaction

Figure 1. Use of token, shared key and nonce in smart payment system

In this paper, we propose a per-transaction shared key scheme, where a new shared key is generated for each smart payment transaction by both the mobile device and the server in the token service provider. The one-time use of the shared key results in a very short lifespan of the key to the degree of below a couple of seconds. We introduce the use of per-transaction key list which consists of two fields, i.e. random nonce N_R and per-transaction shared key K_{DH}. The N_R, which is generated by the server in token service provider, is used as an index to identify the appropriate per-transaction shared key. The per-transaction key list has a very short lifetime that corresponds to the duration required for one payment transaction. This approach greatly reduces the vulnerability of the shared key at the key storage location as well as against brute-force attack due to its very short transaction based lifetime. This paper uses Diffie-Hellman key exchange algorithm to obtain shared key securely between the mobile device and the token service provider. The optimum size of the shared key is derived to satisfy the requirement for computational load and latency issues for the mobile phones, and the security strength for the smart payment transactions.

The rest of the paper is organized as follows. Section 2 presents an overview of the existing token authentication techniques in mobile networks and smart payment system. In section 3, we present our proposed per-transaction shared key scheme. The performance analysis for our proposed system is carried out in section 4. Finally, section 5 concludes the paper.

2. Token Authentication in Mobile Networks and Smart Payment System

2.1 User and Network Authentication in UMTS Mobile Networks

The Authentication and Key Agreement (AKA) procedures in GSM include only user authentication, not the network authentication. The goal of the AKA in GSM is to authenticate the SIM and to establish a cipher key that can be used to protect the user data exchanged between the user and the base station [6]. The GSM authentication process starts when the user sends user International Mobile Subscriber Identity (IMSI) to the Mobile Switching Center/Visitor Location Register (MSC/VLR). IMSI uniquely identifies the user on the network. MSC/VLR requests the authentication data from the Home Location Register/Authentication Center (HLR/AuC). HLR/AuC calculates the Authentication Vector (AV) comprising of random challenge (RAND), expected response (XRES) and cipher key *Kc* and forwards it to MSC/VLR. MSC/VLR sends only the RAND to user. Then the SIM calculates subscriber response (SRES) and cipher key *Kc* based on secret key *Ki* and RAND, and sends SRES to MSC/VLR. Finally, MSC/VLR authenticates the user by performing a verification check. The cipher key *Kc* is used in safeguarding the confidentiality of the data exchanged between the user and the Base Transceiver Station (BTS). As mentioned before, the GSM AKA only authenticates the user but not the network, which can lead to a man-in-the-middle attack. The user cannot verify the network, and as a consequence can be easily exploited by a false base station.

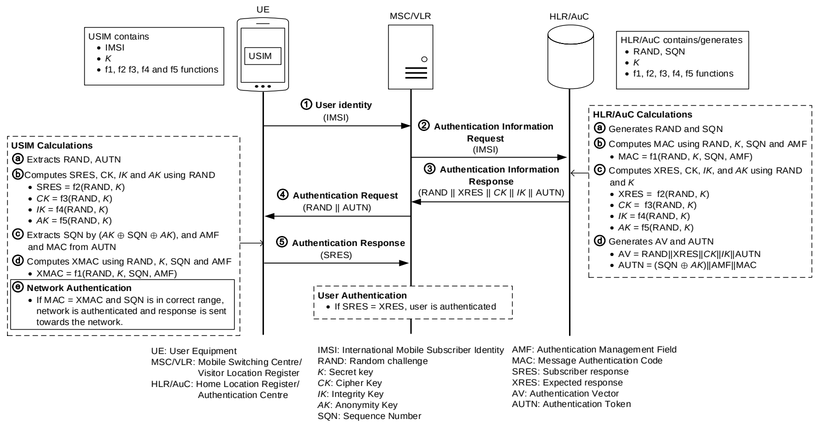
UE
MSC/VLR
HLR/AuC
USIM contains
• IMSI
• K
• f1, f2 f3, f4 and f5 functions
USIM
HLR/AuC contains/generates
• RAND, SQN
• K
• f1, f2, f3, f4, f5 functions
① User identity (IMSI)
② Authentication Information Request (IMSI)
③ Authentication Information Response (RAND || XRES || CK || IK || AUTN)
④ Authentication Request (RAND || AUTN)
⑤ Authentication Response (SRES)
USIM Calculations
ⓐ Extracts RAND, AUTN
ⓑ Computes SRES, CK, IK and AK using RAND
• SRES = f2(RAND, K)
• CK = f3(RAND, K)
• IK = f4(RAND, K)
• AK = f5(RAND, K)
ⓒ Extracts SQN by (AK ⊕ SQN ⊕ AK), and AMF and MAC from AUTN
ⓓ Computes XMAC using RAND, K, SQN and AMF
• XMAC = f1(RAND, K, SQN, AMF)
ⓔ Network Authentication
• If MAC = XMAC and SQN is in correct range, network is authenticated and response is sent towards the network.
User Authentication
• If SRES = XRES, user is authenticated
HLR/AuC Calculations
ⓐ Generates RAND and SQN
ⓑ Computes MAC using RAND, K, SQN and AMF
• MAC = f1(RAND, K, SQN, AMF)
ⓒ Computes XRES, CK, IK, and AK using RAND and K
• XRES = f2(RAND, K)
• CK = f3(RAND, K)
• IK = f4(RAND, K)
• AK = f5(RAND, K)
ⓓ Generates AV and AUTN
• AV = RAND||XRES||CK||IK||AUTN
• AUTN = (SQN ⊕ AK)||AMF||MAC
UE: User Equipment
MSC/VLR: Mobile Switching Centre/ Visitor Location Register
HLR/AuC: Home Location Register/ Authentication Centre
IMSI: International Mobile Subscriber Identity
RAND: Random challenge
K: Secret key
CK: Cipher Key
IK: Integrity Key
AK: Anonymity Key
SQN: Sequence Number
AMF: Authentication Management Field
MAC: Message Authentication Code
SRES: Subscriber response
XRES: Expected response
AV: Authentication Vector
AUTN: Authentication Token

Figure 2. User and network authentication in UMTS

To overcome GSM AKA limitations, the UMTS AKA procedure includes a means to authenticate the network. As shown in Fig. 2, the AKA procedure in UMTS starts with the user identity message from the user. In the UMTS AKA procedure, AUTN and Integrity key *IK* are the additional parameters calculated. AUTN provides the feature of authenticating the network. The user authentication request also includes the AUTN with the RAND. The network is authenticated by the user after verifying the AUTN. After the network has been authenticated by the user, the user identity is also verified by the MSC/VLR. The *IK*, one of the new keys generated in UMTS AKA procedure is used to protect the integrity of signaling messages between the user and Radio Network Controller (RNC) [7-10].

2.2 Token Authentication in Existing Smart Payment Systems

Smart payment applications, Apple Pay and Samsung Pay have brought a new momentum in the NFC enabled mobile payment system. By September 2014, Apple Inc. introduced Apple Pay in iPhone 6, iPhone 6 plus, iPad Air 2, iPad mini 3 and Apple Watch. In 2015, Samsung has introduced the mobile payment feature (Samsung Pay) in its Galaxy S6 and Galaxy S6 edge smart phones as well. Apple Pay and Samsung Pay provide a high level of security and privacy to the credit card information [11]. The consumer's credit card or debit card information is never shared with the merchant nor transmitted over the payment network. The transaction details are stored in the Secure Element (SE) embedded in the smart phone. Apple Inc. and Samsung have also introduced the Touch ID feature in iPhone 6, iPhone 6 plus and Galaxy S6, Galaxy S6 edge, respectively. Touch ID allows the consumer to enter the authentication procedure for the payment transaction using finger prints. Existing smart pay systems use EMVCo tokenization standard to secure the very sensitive consumer's PAN. Even though, both the Apple Inc. and Samsung have not published the security design details of their mobile payment system yet. Some of the overview documents reveal the relating security and privacy techniques, as in [11]. It is clear that Apple Inc. has used the EMVCo tokenization specification to ensure the security of Apple Pay [12]. Fig. 3 shows the token provisioning and token authentication procedure in existing smart payment systems.

2.2.1 Token and Shared Key Provisioning

The token is provisioned to the mobile device once at the time of registering the credit card or debit card to the smart payment system.

Step I – V: The user starts the process by transferring the credit card information into the mobile device. The credit card information entered into the mobile device is encrypted by the device and is sent to the issuer. The token service provider creates the token, encrypts it and sends it along with the shared key *Ks* to the mobile device. The token and *Ks* are stored in the Secure Element (SE) on the phone. *Ks* is used for generating the cryptogram.

2.2.2 Payment/Authorization Request

At this stage of payment transaction, the mobile device sends a token authentication request to the token service provider for authorizing the token/transaction.

Step 1 – 5: These steps are executed when a new transaction is made. When the consumer with a mobile device approaches the NFC Point-of-Sale (POS) terminal, the terminal sends nonce to the mobile device. The mobile device generates a cryptogram (E[*Ks*, Nonce || Token...]) by encrypting the data including the nonce and token with *Ks*, and sends it along with the token in an authorization request to the merchant's POS terminal. The authorization request is then forwarded to the token service provider via the acquirer bank.

2.2.3 Token Authentication

Token authenticity is verified at the following steps.

Step 6 – 9: The token service provider validates the cryptogram by decrypting it with *Ks*, that is, D[*Ks*, E[*Ks*, Nonce || Token]]. Thus the token is authenticated through the validation of cryptogram. After the token is successfully authenticated in the token service provider, it is mapped with its corresponding PAN stored in the Token Vault. Then a modified authorization request is sent to the issuer which contains token and the PAN but does not contain the cryptogram. The issuer authenticates the PAN.

2.2.4 Authorization Reply

This is the final stage of the payment authentication process. The mobile device and the merchant's POS terminal receive the transaction authorization reply and the payment process is completed.

Step 10 – 13: After the PAN has been authenticated by the issuer, it sends an authorization reply to the token service provider that the PAN is authenticated. The token service provider includes the nonce in the authorization reply and sends it to the acquirer, which then forwards the authorization reply to the merchant's POS terminal. The merchant's POS terminal compares the nonce and validates the transaction. Thus the mobile device completes the transaction successfully.

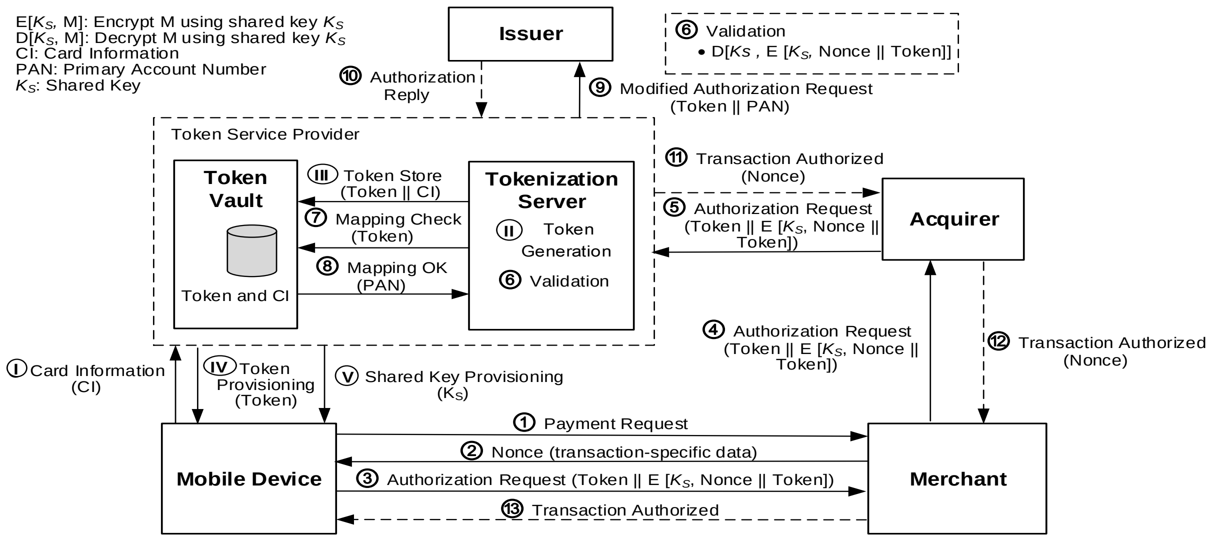
E[K_S, M]: Encrypt M using shared key K_S
D[K_S, M]: Decrypt M using shared key K_S
CI: Card Information
PAN: Primary Account Number
K_S: Shared Key
Issuer
⑥ Validation
• D[K_S , E [K_S, Nonce || Token]]
⑩ Authorization Reply
⑨ Modified Authorization Request (Token || PAN)
Token Service Provider
Token Vault
Token and CI
Ⅲ Token Store (Token || CI)
⑦ Mapping Check (Token)
⑧ Mapping OK (PAN)
Tokenization Server
Ⅱ Token Generation
⑥ Validation
⑪ Transaction Authorized (Nonce)
⑤ Authorization Request (Token || E [K_S, Nonce || Token])
Acquirer
④ Authorization Request (Token || E [K_S, Nonce || Token])
⑫ Transaction Authorized (Nonce)
Ⅰ Card Information (CI)
Ⅳ Token Provisioning (Token)
Ⅴ Shared Key Provisioning (K_S)
Mobile Device
Merchant
① Payment Request
② Nonce (transaction-specific data)
③ Authorization Request (Token || E [K_S, Nonce || Token])
⑬ Transaction Authorized

Figure 3. Token authentication mechanism of existing Smart Payment System

In existing smart payment systems the transaction authentication process primarily relies on the token authentication. In the whole process the PAN is not transmitted over the payment network, instead a token is assigned to the device and transmitted over the payment network. If the token is intercepted by the hackers, it is of no use to them because they cannot generate a valid cryptogram without the shared key *Ks*, which is a secret between the mobile device and the token service provider. Also, the token cannot be used to obtain the actual credit card number [13].

2.3 Difference of Tokens Used in UMTS and Smart Payment System

The tokens used in UMTS to authenticate the network and in smart payment system to authenticate the payment transaction are different in some aspects. As shown in Table 1 the token used in UMTS, known as authentication token, is generated cryptographically using shared secret key *K*. While the token used in smart payment system is not generated cryptographically but generated randomly in place of PAN [14].

Table 1. Comparison between tokens used in UMTS and Smart Payment System

	Token used in UMTS	Token used in Smart Payment System
Generation	Generated cryptographically using shared secret key *K*	Generated randomly in place of PAN
Token nature	The mobile device (USIM) and HSS have shared key *K* to authenticate the token.	The issuers maintain a "token vault" that maps tokens back to their respective PANs. Therefore it is impossible for a malicious agent to figure out the PAN from the token.
Token size	128 bits	13 – 19 digits
Usage	New token is created when user and network authentication is necessary	Static use once the token is provisioned

3. Proposed Per-transaction Shared Key Scheme in the Smart Payment System

3.1 Use of the Per-transaction key list

In existing smart payment systems, *Ks* is agreed between the mobile device and the token service provider once at the time of registering the credit card to the smart payment application. The long lifetime of *Ks* is needed because *Ks* is used in every transaction. The potential drawback of long lifetime of *Ks* is its vulnerability at the key storage points and also against the brute-force attack. The cryptogram generated with *Ks* is used to verify the token authenticity. Once the token is authenticated, the payment transaction is approved. If the hackers intercept the token at NFC POS terminal and they could break the shared key, they will be able to generate a hacker's side cryptogram. With the hacker's side cryptogram, the token service provider may fail to distinguish it from the original cryptogram. Hence, the vulnerable shared key can lead to forged transactions. This motivated us to propose the per-transaction shared key scheme, which enables the payment transaction to be more secure against the above attacks because the shared key changes at every payment transaction.

The key idea of the proposed per-transaction shared key scheme is related to use the per-transaction key list, which is managed by the server in the token service provider. For the new payment transaction, mobile device starts with the Diffie-Hellman key exchange procedure in order that the server can create an entry of the per-transaction key list. The new entry of the list consists of two fields: N_R and K_{DH}. N_R is used only as an index to identify the appropriate per-transaction shared key, i.e. K_{DH}. Each entry of per-transaction key list has a very short lifetime that corresponds to the duration required for token authentication process. The cryptogram, which is a transaction-unique value, is computed by encrypting nonce and token with the per-transaction shared key. When the hackers intercept the token and nonce for the payment transaction, it takes time for them to break the per-transaction shared key. However, the hacker's effort is useless because the lifetime of the per-transaction shared key already expires by the time when the hacker succeeds to break the key. Also, our proposed per-transaction shared key scheme is attractive because the server does not need to maintain the state for the mobile device that initiates to create the per-transaction shared key. Once the per-transaction shared key is created, it is identified by using the index of N_R, which has no relation with the mobile device's information.

Considering that our proposed scheme requires the per-transaction key exchange procedure, the network connection between the mobile device and the server should be available at the beginning stage of the payment transaction. So, our proposed system has additional feature to solve the scenario where the network connection is not available. As shown in Fig. 4, whenever the network is available and the key agreement procedure is successful then the per-transaction shared key could be used to generate the cryptogram. Otherwise, the pre-shared key *Ks* is used for generating the cryptogram. We assume that in our proposed scheme the per-transaction shared key is used most of the time, whereas the possibility of using *Ks* is very low. Even though the lifetime of *Ks* is relatively long, our payment transaction is secure because the pre-shared key *Ks* is seldom used.

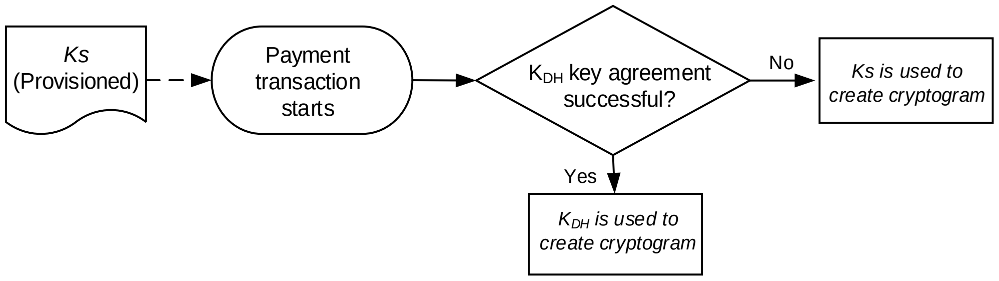
Ks (Provisioned)
Payment transaction starts
K_{DH} key agreement successful?
No
Ks is used to create cryptogram
Yes
K_{DH} is used to create cryptogram

Figure 4. Frequently used per-transaction key

3.2 Token and *Ks* Provisioning

Step I – V: Similar to existing smart payment systems, the process of token and *Ks* provisioning in our proposed system is performed only once at the time of registering the credit card to the smart payment application. Our system uses the same procedure for token and *Ks* provisioning as the existing smart payment systems.

3.3 Per-transaction Shared Key Exchange

In the proposed per-transaction shared key scheme the server and the mobile device use Diffie-Hellman key exchange protocol to agree on the per-transaction shared key. In Diffie-Hellman protocol, first of all, both the mobile device and the server publicly agree on the group parameters: prime numbers p and g, where p is the prime modulus and g is the primitive root. Both sides generate their private random numbers, x (mobile device's random number) and y (server's random number). Then the mobile device and the server compute public value e (g^x mod p) and public value f (g^y mod p), respectively, and share these values with each other publicly. Both sides use the public values e and f to compute the same Diffie-Hellman key K_{DH} as follow.

- Mobile device calculates: $K_{DH} = f^{x} \bmod p$
- Server calculates: $K_{DH} = e^{y} \bmod p$

The security strength of the per-transaction shared key K_{DH} depends on the length of the prime modulus p. The longer the prime modulus, the longer it will take to break the shared secret. Fig. 5 shows the proposed per-transaction shared key scheme.

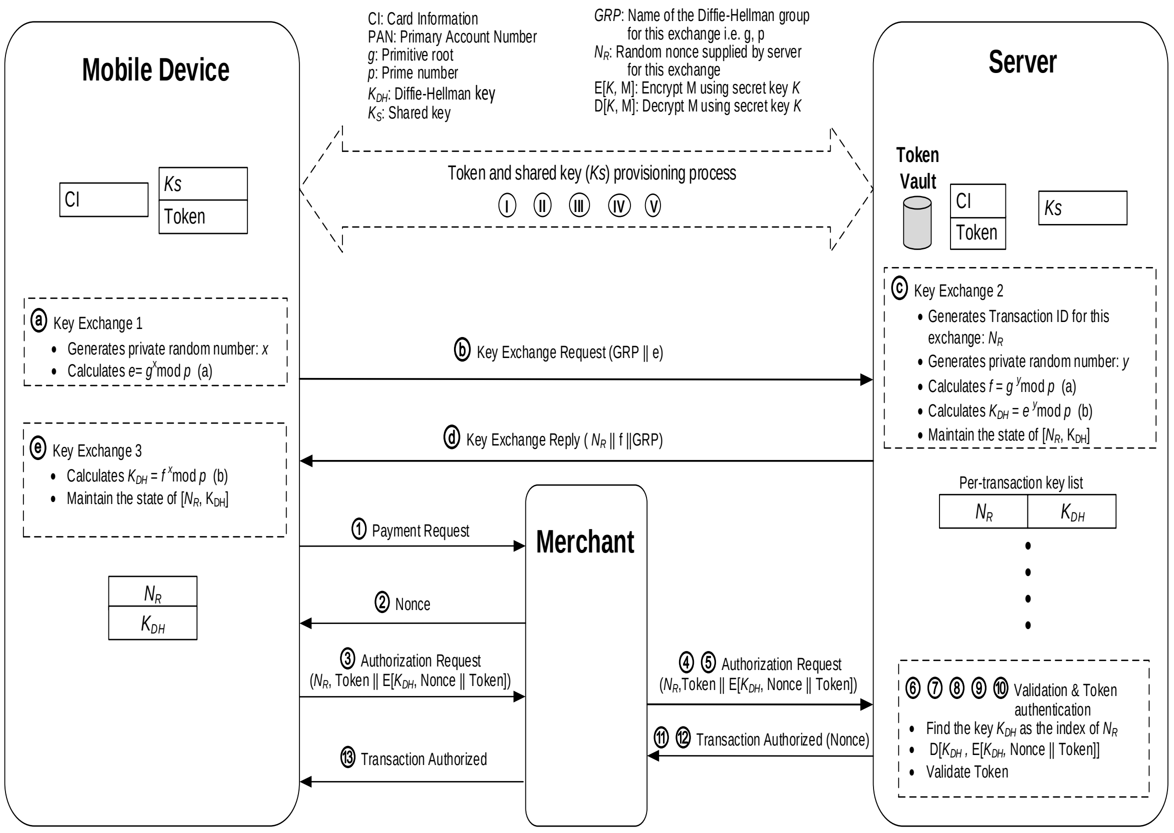
Mobile Device
Server
Merchant
CI: Card Information
PAN: Primary Account Number
g: Primitive root
p: Prime number
K_DH: Diffie-Hellman key
K_S: Shared key
GRP: Name of the Diffie-Hellman group for this exchange i.e. g, p
N_R: Random nonce supplied by server for this exchange
E[K, M]: Encrypt M using secret key K
D[K, M]: Decrypt M using secret key K
Token and shared key (Ks) provisioning process
Ⓘ Ⓘ Ⓘ Ⓘ Ⓥ
CI
Ks
Token
Token Vault
ⓐ Key Exchange 1
Generates private random number: x
Calculates e= g^x mod p (a)
ⓑ Key Exchange Request (GRP || e)
ⓒ Key Exchange 2
Generates Transaction ID for this exchange: N_R
Generates private random number: y
Calculates f = g^y mod p (a)
Calculates K_DH = e^y mod p (b)
Maintain the state of [N_R, K_DH]
ⓓ Key Exchange Reply (N_R || f ||GRP)
ⓔ Key Exchange 3
Calculates K_DH = f^x mod p (b)
Maintain the state of [N_R, K_DH]
Per-transaction key list
N_R
K_DH
① Payment Request
② Nonce
③ Authorization Request (N_R, Token || E[K_DH, Nonce || Token])
④ ⑤ Authorization Request (N_R,Token || E[K_DH, Nonce || Token])
⑥ ⑦ ⑧ ⑨ ⑩ Validation & Token authentication
Find the key K_DH as the index of N_R
D[K_DH , E[K_DH, Nonce || Token]]
Validate Token
⑪ ⑫ Transaction Authorized (Nonce)
⑬ Transaction Authorized

Figure 5. Token authentication with the per-transaction shared key

Step a – e: In our proposed system the per-transaction shared key is exchanged between the mobile device and the token service provider every time at each transaction. To begin the payment transaction, the mobile device generates its private random number x, calculates public value e (g^x mod p) and sends the key exchange request message, which contains public value along with the Diffie-Hellman group (GRP) for this exchange i.e. g and p, to the server. When the key exchange request arrives, the server begins the key exchange 2 procedure. In key exchange 2, the server generates its private random number y and N_R which also serves as a transaction ID for this particular key exchange, and calculates public value f (g^y mod p) and K_{DH} ($e^{\,y}$ mod p). The server then maintains a per-transaction key list where K_{DH} is indexed by N_R. The per-transaction key list has a short lifetime that corresponds to the duration required for token authentication process. Next, the server sends key exchange reply, which contains N_R, the public value f and GRP, to the mobile device. When the mobile device receives key exchange reply message, the key exchange 3 procedure begins. In key exchange 3, the mobile device also calculates the same K_{DH} ($f^{\,x}$ mod p) as the server and maintains the state of [N_R, K_{DH}]. The mobile device and the server agree on a per-transaction shared key and complete the key exchange process.

The rest of the process that includes transaction authorization request (**step 3 - 5**), validation and token authentication (**step 6 - 10**) and transaction authorization reply (**step 11 - 13**) is similar to the existing smart payment systems except for including an additional parameter N_R in the transaction authorization request. The payment transaction authorization in our proposed system relies on token authentication with the per-transaction shared key.

4. Performance Analysis for the Proposed System

4.1 Security Comparison

As mentioned in the preceding sections, the existing smart payment system uses the same shared key for every transaction, which is assigned to mobile device at the time of registering the credit card to the smart payment system. The long lifetime of the shared key makes its security vulnerable [15]. However, our proposed system uses per-transaction shared key, which has very short lifetime. The per-transaction shared key once used will never be reused for the other transactions, which can contribute to enhance the security level.

Another advantage of the proposed system is related to the size of the shared key. Because the shared key in the existing smart payment system possesses a long lifetime, the size of the key should be large to fulfill the security requirement of the key. On the other hand, the proposed system uses the per-transaction shared key, where each size is relatively small. Meanwhile, the existing smart payment system does not require computational and network load for the key exchange. The additional burden of our proposed system is that the computational and network load is caused because of computing a new key on every transaction over the network. Table 2 summarizes the security comparisons between the existing smart payment system and proposed system.

Table 2. Security comparison between existing smart payment system and proposed system

	Existing Smart Payment System	Proposed System with the Per-transaction Shared Key Mechanism
Shared key creation	Per registration	Per transaction
Required key size	Large	Relatively small
Security	Vulnerable due to long lifetime and repeated use of the same key	Secure because of the use of per-transaction key during a short period of time
Computational load	Not needed	Modular exponentiation computing
Network load	Not needed	Exchange of public values for each transaction

4.2 Transaction Latency and Security Strength

This subsection analyzes the required computational load for the key exchange. We neglect the network load for the key exchange because the rate of the payment transactions is very low from the mobile device viewpoint. For our proposed per-transaction shared key scheme, the mobile device additionally needs the computation of Key exchange 1 and Key exchange 3, that is, [(a) + (b)] in Fig. 5. Those computations in Key exchange 1 and Key exchange 3 are the main factors that increase the whole transaction latency. For the purpose of measuring the transaction latency for the mobile device, we used an android tablet (LG G Pad 7) with Quad-core 1.2 GHz Cortex-A7 CPU and internal memory of 1GB RAM, which runs the total latency that is composed of DH computation [(a) + (b)]. As shown in Fig. 6, the measurement results show that transaction latency remains around 264 milliseconds even though 80-bit K_{DH} is used, while the 48-bit key causes a total latency of around 144 milliseconds.

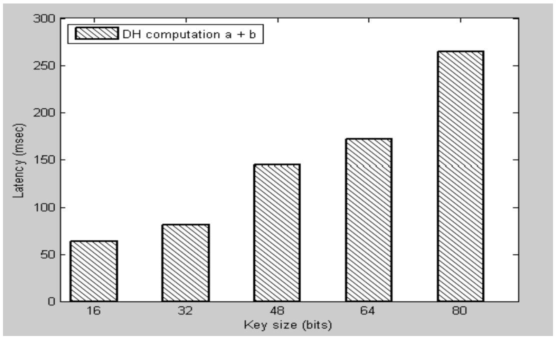

Figure 6. Average latencies for different key sizes

It is intuitively clear that the security strength is proportional to the key size. The exhaustive key search will require to try every value among all possible keys of $2^{`key\ size'}$. With the use of a massive parallel microprocessor, it may be possible to achieve processing rates in many order of magnitude. According to [16], recent attackers have capability to try up to $10^{12}$ keys per second. Then, those powerful attackers will spend $\frac{2^{`key\ size`}}{2*10^{12}}$ seconds on average to break the key successfully. However, normal attackers will use the same computing devices as that this paper uses for the purpose of analysis shown in Fig. 6. We define the computing latency to compute (a) in Key exchange 1 and (b) in Key exchange 3 in Fig. 5 as $L_{a+b}(n)$, where n is the key size in bits. Then, the latency are shown in Table 3. The normal attackers, which use normal computing devices, will spend $\frac{2^n * L_{a+b}(n)}{2}$ seconds on average to break the key successfully.

Table 3. Latency for DH key computation

L_{a+b} (16)	L_{a+b} (32)	L_{a+b} (48)	L_{a+b} (64)	L_{a+b} (80)
64 msec	81 msec	144 msec	172 msec	264 msec

Fig. 7 shows the minutes it will take for the two kinds of brute-force attackers (powerful attacker and normal attacker) to break the key. In our proposed scheme, the per-transaction shared key is valid only for a short transaction period. Considering that the lifetime of our transaction key is below 1 minute, the 48-bit key is enough to make the transactions secure against even the powerful attacker while causing the additional latency of 144 milliseconds for each payment transaction. This amount of latency burden can be neglected considering the improved security aspects in our proposed smart payment system with the per-transaction shared key scheme.

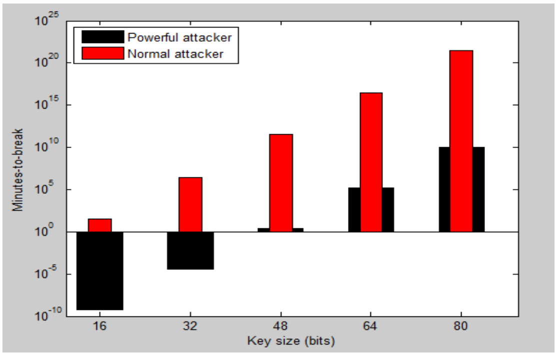

Figure 7. Minutes to break

5. Conclusion

In this paper, we discussed how mobile phone networks and smart payment systems deal with the tokens for user and network authentication. Token authentication in existing smart payment system is advantageous because token avoids credit card number to be intercepted over the network. In order to use the token for authentication, the cryptogram should be generated at each transaction in the mobile device and securely sent to the token service provider. The conventional cryptogram is computed using the shared key which was provisioned at the time of registering the credit card to the smart payment system. The long lifetime of the shared key is needed because the conventional cryptogram is repeatedly used in every payment transaction. The potential drawback of long lifetime and frequent use of the shared key is related to its vulnerability against the intrusion and brute-force attack. In this paper, we proposed a per-transaction shared key scheme, where a new per-transaction shared key is agreed between the mobile device and token service provider on every smart payment transaction basis. We introduced a per-transaction key list at the server side which is easy to handle because the per-transaction key has a short lifetime below a couple of seconds and the server does not need to maintain the state for the mobile device. We analyzed the optimum size of the per-transaction shared key which satisfy the requirements for transaction latency and security strength for secure payment transactions. Considering that the lifetime of our per-transaction key is below 1 minute, the 48-bit key was enough to make the transactions secure against even the powerful attackers while causing the additional latency of 144 milliseconds for each smart payment transaction. This amount of latency burden can be neglected considering the improved security aspects in our proposed smart payment system with the per-transaction shared key scheme. As a result, this paper proved that the proposed per-transaction shared key scheme is more effective to make the smart payment transaction secure than the existing system that employs a repeatedly used shared key with a long lifetime.

Acknowledgement

This research was supported by the Research Fund, 2015 of the Catholic University of Korea.

References

[1] Bodhani, "New ways to pay [communications near field]," Engineering Technology, vol.8, no.7, pp.32–35, August 2013.

[2] F. Corella and K. Lewison, "Interpreting the emv tokenisation specification," white paper, 2014.

[3] W. Chen, G. Hancke, K. Mayes, Y. Lien, and J.H. Chiu, "Using 3g network components to enable nfc mobile transactions and authentication," Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on, pp.441–448, Dec 2010.

[4] S. Sung, C. Youn, E. Kong, and J. Ryou, "User authentication using mobile phones for mobile payment," Information Networking (ICOIN), 2015 International Conference on, pp.51–56, Jan 2015.

[5] P. Tanvi, G. Sonal, and S. Kumar, "Token based authentication using mobile phone," Communication Systems and Network Technologies (CSNT), 2011 International Conference on, pp.85–88, June 2011.

[6] Tang, D.A. Naumann, and S. Wetzel, "Analysis of authentication and key establishment in inter-generational mobile telephony," High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC EUC), 2013 IEEE 10th International Conference on, pp.1605–1614, IEEE, 2013.

[7] Zhang, R. Zhang, X. Niu, Y. Yang, and Z. Zhang, “A new authentication and key agreement protocol of 3g based on diffie-hellman algorithm,” Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, pp.V2–110–V2–113, April 2010.

[8] K.A. Alezabi, F. Hashim, S.J. Hashim, and B.M. Ali, “An efficient authentication and key agreement protocol for 4g (lte) networks,” Region 10 Symposium, 2014 IEEE, pp.502–507, April 2014.

[9] M. Purkhiabani and A. Salahi, “Enhanced authentication and key agreement procedure of next generation evolved mobile networks,” Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, pp.557–563, May 2011.

[10] M. Purkhiabani and A. Salahi, “Enhanced authentication and key agreement procedure of next generation 3gpp mobile networks,” International Journal of Information and Electronics Engineering, vol.2, no.1, pp.69–77, 2012.

[11] “Apple pay security and privacy overview,” 2015. https://support.apple.com/en-us/HT203027.

[12] `Ortiz-Yepes, “A critical review of the emv payment tokenization specification,” Computer Fraud & Security, vol.2014, no.10, pp.5–12, 2014.

[13] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, “Potential misuse of nfc enabled mobile phones with embedded security elements as contactless attack platforms,” Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for, pp.1–8, Nov 2009.

[14] Kirk Lennon, “How Apple pay really works and why you should begin using it immediately”. http://www.kirklennon.com/a/applepay.html.

[15] Y. Jung, E. Festijo, and J.W. Atwood, “Securing rtp packets using per-packet key exchange for real-time multimedia,” ETRI Journal, vol.35, no.4, pp.726–729, 2013.

[16] W. Stallings, Network security essentials: applications and standards, Pearson Education India, 2007.