Course: Cryptographic systems

The μiKP protocol: a micro description

22 June 2005

Nikos Mavrogiannopoulos 0581476

Contents

1 Introduction

The iKP protocols are protocols intended to provide security to on-line transactions, and most specifically credit card payments. They were proposed in 1995 by a group of IBM researchers in [1]. However these protocols were considered to be computationally expensive in some contexts, such as environments that are memory constrained, or with limited CPU capabilities. For this reason and for the context of payments of small amounts of money[1], the μiKP protocol has been proposed by some of the original authors of iKP in [2]. This proposal is marked as a research report, mainly intended for internal use and gathering of comments, thus several parts of the protocol are undefined, reasoning behind some design decisions is not given, and the security requirements are not formally listed anywhere[2].

As in the iKP protocols there are three roles in the protocol. The customer, the merchant and the acquirer. In some cases a broker role is introduced. In this memo we will not follow the conventions found in the original document for the parties, but rather follow the conventions used in iKP paper, in order to be consistent in our descriptions and comparisons. That is Acquirer is the party responsible for issuing and checking the account numbers (IE. credit card numbers), and Merchant and Customer have their natural meaning, while the Broker acts as a mediator between the Merchant and the Customer. Since all of the roles have direct equivalent in the real world of money transactions, a potential user of this protocol will be acquainted with the roles and his responsibilities.

2 Security requirements

The security requirements and goals set for on-line transactions are not explicitly defined in the original paper. Here for our analysis we will compare this protocol to the original iKP protocol. The reasoning behind this is to show the benefits and the losses by using a thin protocol intended for micro payments, to the use of a secure payment protocol. So we will be using the requirements[3] set for the iKP family of protocols[4] as shown in Figure 1. The definitions of these requirements can be found in [1] with the addition of the following ones:

Acquirer requirements.

A3- *Proof of transaction authorization by Broker.* This is the same as the "proof of transaction authorization by customer" since when the acquirer debits the account of the broker for a certain amount, he must be in possession of an unforgeable proof that the owner of the account has authorized this request. We will distinguish between weak proof and undeniable proof, which provides non-repudiation.

Merchant requirements.

M3- *Proof of transaction authorization by Broker.* The merchant might need an unforgeable proof that the broker has authenticated the transaction in progress. Again we distinguish between weak proof and undeniable proof.

Customer requirements.

C6- *Receipt from Broker.* As with case where the customer talks directly to the merchant, when the broker is involved the customer wants a proof that the broker has received payment and promised to forward the amount to the merchant.

[1] called micro-payments.

[2] Although not explicitly written, the authors imply that protocol to be comparable to other existing micro-payment protocols.

[3] We will not discuss the customer privacy requirement –$C5$– although it seems that the same levels of customer privacy are achieved between the two protocols.

[4] with the addition of the role of Broker.

C7- *Certification and authentication of the Broker.* The customer may need a proof that the broker is accredited at an acquirer.

Broker requirements.

B1- *Proof of transaction authorization by Acquirer.* The broker needs an unforgeable proof that the acquirer has authorized the transaction in progress. That is assured that the payment can take place. Again we distinguish between weak proof and undeniable proof.

B2- *Proof of transaction authorization by Customer.* The merchant might need an unforgeable proof that the customer has authorized the transaction in progress. We distinguish between weak proof and undeniable proof.

B3- *Receive a receipt from Merchant.* The broker may need a proof that the merchant has received the payment. Again we distinguish between weak proof and undeniable proof.

Acquirer
A1. Proof of transaction authorization by Customer
A2. Proof of transaction authorization by Merchant
A3. Proof of transaction authorization by Broker
Merchant
M1. Proof of transaction authorization by Acquirer
M2. Proof of transaction authorization by Customer
M3. Proof of transaction authorization by Broker
Broker
B1. Proof of transaction authorization by Acquirer
B2. Proof of transaction authorization by Customer
B3. Receipt from Merchant
Customer
C1. Unauthorized payment is not possible
C2. Proof of Transaction Authorization by Acquirer
C3. Certification and Authentication of Merchant
C4. Receipt from Merchant
C6. Receipt from Broker
C7. Certification and Authentication of Broker

Figure 1: Requirements for the iKP protocols

3 The μiKP protocol

In the iKP description two protocols are proposed that are used in two different scenarios. These are

- Repeated micro-payments to a single Merchant
- Non-Repeated micro-payments to many Merchants

They are on-line protocols, and they require all the involved parties[5] to be active during a transaction. The message flow for the protocol changes for each scenario, but the second protocol is based on the first one, and in both cases they rely on the iKP family of protocols. Although not specified in the protocol, it is assumed that the iKP protocol that will be used[6] depends on

[5] Customer, Merchant, Broker and Acquirer.
[6] that is 1KP, 2KP or 3KP.

current iKP protocol deployment and the security requirements acceptable by all parties. For this reason we will make our description a bit abstract and will not consider the details that a single iKP protocol may imply. For example we will not study only particular combinations of protocols such as μiKP with 3KP, but we'll assume a general iKP framework protocol.

The security of the μiKP protocols relies on hash chains. A brief description of this concept can be found in section A. The usage of hash chains in the protocol is done as follows. We assign the name coupon to each hash in the chain and a fixed value that applies to all of them. Each time somebody wants to pay a multiple, say k, of the value of the coupons, he reveals the next last k hashes to the other party. The synchronization and verification of the coupon will be defined in the individual protocols.

Here we list all the cryptographic primitives and the quantities that are common to these protocols. This is based on [1] so one familiar with the document may skip parts this section. These are given in order to understand the message flow in the protocol descriptions that follow.

- Keys

PK_X, SK_X	These stand for the public and secret keys of party X. X in this document can take the values C for customer, M for merchant, A for acquirer and CA for the certificate authority.
$CERT_X(\cdot)$	The public key certificate of party X. This has to be issued by the CA.

- Cryptographic Primitives

$H(\cdot)$	A strong collision and preimage resistant hash function.
$\mathcal{E}_X(\cdot)$	Public key encryption using PK_X, done in a way to provide confidentiality and message integrity.
$MAC_X(\cdot)$	Message authentication code X as key. Provides message integrity and authentication.
$A^i(\cdot)$	The $n-i$ element of a hash chain with n elements.
$S_X(\cdot)$	Signature with respect to SK_X. It is assumed that the signature function hashes the message before signing.

- Quantities occurring in the iKP protocols

$SALT_C$	A random value generated by the customer.
$DATE$	This is a merchant's time stamp.
$NONCE_M$	A random value (nonce) generated by merchant.
ID_M	A merchant's identifier. Usefull to acquirer.
TID_M	A transaction identifier chosen by the merchant. Must uniquely identify a transaction.
$DESC$	The description of purchase and delivery address. It also includes payment information such as credit card name, bank identification number and currency. For μiKP the description must be broad since the contents of the transaction are not known.
CAN	The customer's account number (credit card number).
R_C	A random value chosen by customer.
CID	This is a customer ID for a transaction. This is formed as $H(R_C, CAN)$.
Y/N	Responce from the acquirer. A YES/NO responce.
V	A random number generated by merchant.
PIN	The customer's PIN associated with his account number.

- New quantities occurring in the μiKP protocols

P_C	A random number generated by customer.
P_B	A random number generated by broker.
V'	A random number generated by merchant.
$VALUE/COUPON$	The value assigned to each coupon.
n	The number of coupons.
$PFLAGS$	An extra field used to distinguish μiKP from iKP .

The constructed fields in μiKP are not identical to the corresponding iKP fields. In the $COMMON$ structure the $PRICE$ field has been removed and the fields $A^n(P_C)$, n, $VALUE/COUPON$, $H(V')$ have been added. All the constructed fields used in the protocol are listed in Figure 2. The fields available in 2KP and 3KP are marked with a single asterisk (*), while the fields of 3KP only are marked with a double (**).

Structure	Contents
COMMON	$A^n(P_C)$, n, $VALUE/COUPON$, $H(V')$, $PFLAGS$, ID_M, TID_M, $DATE$, $NONCE_M$, CID, $H(DESC, SALT_C)$, $H(V)^*, H(V')$.
CLEAR	$A^n(P_C)$, ID_M, TID_M, $DATE$, $NONCE_M$, $H(COMMON)$, $H(V)^*$.
SLIP	n, $VALUE/COUPON$, $H(COMMON)$, CAN, R_C and optionally the PIN of the CAN holder.
ENCSLIP	$\mathcal{E}_A(SLIP)$.
SIG^*_M	$S_M(H(COMMON), H(V), H(V'))$.
SIG_A	$Y/N, S_A(Y/N, H(COMMON))$.
SIG^{**}_C	$S_C(ENCSLIP, H(COMMON))$.

Figure 2: The constructed fields in the μiKP protocol

4 Scenario 1: Repeated micro-payments

4.1 The protocol

In this scenario there are only three roles involved. These are the Customer, the Merchant and the Acquirer. The protocol consists of two phases. The first phase is the iKP phase, where all parties are involved. In this phase instead of an iKP payment we have the customer generate some random coupons with a fixed value for each and perform the iKP protocol. That way he gets a clearance from the acquirer to use his coupons with this merchant. Since the merchant is also involved, he becomes aware of the value of coupons. Also a description of goods or services that the customer may access is included in the protocol (in the iKP 's $DESC$ field).

The second phase is the μiKP phase where the customer spends his coupons to this merchant. During the spending time the acquirer does not need to be involved. After customer has finished spending his coupons, the merchant sends the coupons he got to the acquirer[7]. The μiKP protocol message sequence for this scenario is shown in Figure 3.

4.2 iKP message sequence

Let's now examine the message sequence in detail. We will examine the messages as shown in Figure 3 in their generated sequence. Some familiarity with the iKP protocols is required here. Again we follow the convention that quantities marked with the single asterisk are used in 2KP and 3KP while with a double in 3KP only.

[7] Actually he only needs to send the last one he received.

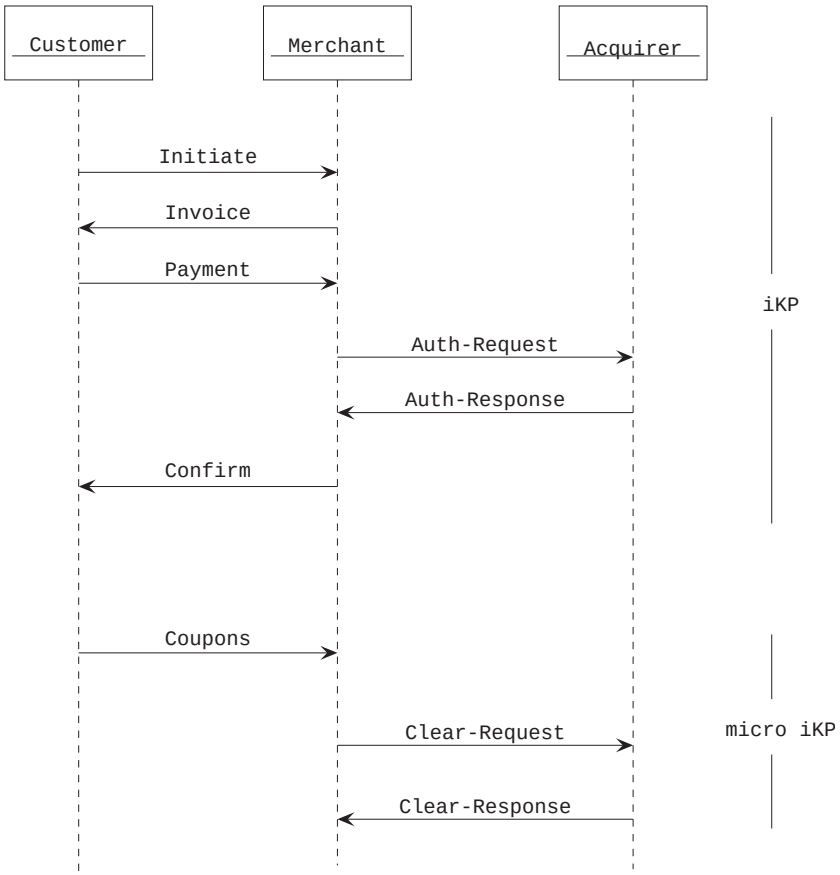

Figure 3: The μiKP message flow

Initiate: Customer $\xrightarrow{A^n(P_C),CID,SALT_C,CERT_C^{**}}$ Merchant
The customer sends a initiate message to the merchant that includes the last hash $A^n(P_C)$ in his hash chain. Everything else is the same as in the iKP protocol in use, that is the identification of the customer CID, a salt $SALT_C$ and, if using 3KP, a certificate of customer $CERT_C$.

Invoice: Merchant $\xrightarrow{CLEAR,SIG_M^*}$ Customer
After the receipt of *Initiate* message, the merchant responds with this message. This consists of $CLEAR$, that is his identity, a transaction id, a date and a nonce that are supposed to make the transaction unambiguously referenced by the acquirer. The hash of $COMMON$ is included also in this message and is used through this session by all parties to unambiguously reference the session. In 2KP or 3KP this message will be signed using the merchant's secret key.

Payment: Customer $\xrightarrow{ENCSLIP,SIG_C^{**}}$ Merchant
At that point the customer replies by sending this message. The contents are the same as in the iKP protocol in use, and that is the contents of $SLIP$ encrypted with acquirer's public key and in 3KP the signature of the customer over the encrypted $SLIP$ and the hash of $COMMON$.

With this message the customer, identifies himself to the acquirer and associates his account number with the transaction. In the case of 1KP and 2KP no digital signatures are used, thus non-repudiation is impossible. The $ENCSLIP$'s contents cannot be read by the merchant.

Auth-Request: `Merchant` $\xrightarrow{CLEAR,H(DESC,SALT_C),ENCSLIP,SIG_M^*,SIG_C^{**}}$ `Acquirer`
This message is sent by the merchant to the acquirer. It contains the $CLEAR$ structure, a hash of the services/goods description together with a salt selected by the customer, the encrypted $SLIP$ and, if available, the signatures of the customer and the merchant. That is the transaction details plus the client's approval of it, either by a digital signature or by the encryption function[8].

Auth-Response: `Acquirer` $\xrightarrow{Y/N,SIG_A}$ `Merchant`
On receipt of the *Auth-Request* message the Acquirer checks the given transaction details for replays, decrypts the $ENCSLIP$ and checks the client's CAN for validity. If everything was right he responds by sending this message to the merchant. It contains the clearance of the acquirer over the requested action, and this approval is undeniable since it is signed with his key, thus everyone can verify it.

Confirm: `Merchant` $\xrightarrow{Y/N,V^*,SIG_A}$ `Customer`
Then the merchant after verifying the signature of the acquirer, forwards the relevant parts of acquirer's responce to the customer, and this is the final message of this phase. In 2KP and 3KP the value V is revealed at the point to the customer. This V is used as an assurance that the merchant has accepted the authorization request, and is used instead of a signature.

4.3 μiKP message sequence

Let's now examine the message sequence in detail. We will examine the messages as shown in Figure 3 in their generated sequence.

Coupons: `Customer` $\xrightarrow{i,A^i(P_C)}$ `Merchant`
This is not a single message. It can be send several times (maximum is n), from the Customer to the Merchant. It contains the values $i, A^i(P_C)$, which corresponds to a coupon together with it's numerical value in the chain. Each time i is decreased by one.

Clear-Request: `Merchant` $\xrightarrow{SIG_A,V'^*,A^{n-j}(P_C)}$ `Acquirer`
At the moment the Merchant decides that the customer is no longer going to use any more coupons, he sends this message to the Acquirer. This message consists of $SIG_A, V'A^{n-j}(P_C)$. That is the authorization of the acquirer for the Customer's coupons as sent in the iKP phase and the last coupon received by the Merchant, and j is the number of coupons received. In 2KP or 3KP V' is revealed to acquirer, and, like the V introduced in 2KP, is used as a signature[9] from the merchant.

Clear-Reply: `Acquirer` $\xrightarrow{Y/N,SIG'_A}$ `Merchant`
The acquirer then, replies to the merchant with this message. It contains
$Y/N, S_A(Y/N, SIG_A, V', A^{n-j}(P_C))$. And this is his approval for the transaction. This approval is undeniable since it is signed with his key, thus everyone can verify it.

4.4 Security properties

Let's now examine the security properties that this protocol has. We will now discuss and try to explain in details each one separately.

[8] Which is supposed to provide not only confidentiality but also some integrity. See [1].
[9] The usage of V' is not defined in the [2] paper. This is our interpretation of it based on facts from [1].

A1. Acquirer has a proof of transaction authorization by Customer: The acquirer only receives the *Auth-Request* and the *Clear-Request* messages from the merchant. Thus these messages should provide the proof of authorization by the customer. The first message contains a proof that the customer committed to a hash chain, and thus requested some coupons to spend later. In the case of 3KP usage this proof is undeniable since it signed by his private key. This of course is in no way a transaction authorization, because the customer has not really initiated the transaction yet (he may not use the coupons at all). The *Clear-Request* message contains the last coupon that the customer gave to the merchant. Since the used hash function is considered preimage resistant this can be seen as a proof that the customer actually spent this coupon. This value can be associated with the given customer, and defines a precise amount of money, thus can be seen as a proof of transaction authorization by the customer. Since it does not serve as proof for third parties it cannot be seen as an undeniable proof.

A2. Acquirer has a proof of transaction authorization by Merchant: Since the only messages that acquirer gets from the merchant are the *Auth-Request* and the *Clear-Request* messages, the proof of transaction authorization, if any must be there. The first message *Auth-Request*, in the case of 2KP or 3KP contains a signature of the merchant, authenticating $H(V')$. This can be seen as a commitment scheme where the merchant commits to a random value V' and authenticates his commitment with a signature. Then at the *Clear-Request* message he reveals his commited value to the acquirer. This value can be used by the acquirer to prove any third party that the merchant has accepted the transaction. However this proof cannot be associated with any amount of money since this is only known when the *Clear-Request* has been sent, far later than the merchant's commitment.

M1. Merchant has a proof of transaction authorization by Acquirer: Here the merchant has to be ensured that the customer has indeed the required amount of money for the coupons he will spend. This part of the protocol is negotiated in the iKP part, in the *Auth-Response* message, thus this property depends on it. In the *Auth-Response* message the merchant and the customer get a clearance to proceed with the protocol given the number of coupons and the amount of money per coupon. This message is signed by the acquirer and contains a hash of all the values in use (such as the $A^n(P_C), n, VALUE/COUPON$), thus the merchant has an undeniable proof of transaction authorization by the acquirer for all flavors of iKP . Note that although the acquirer did not authorize a specific amount of money to be used, he authorized an upper limit of money the Customer can spend in this Merchant. This is sufficient as a proof of transaction authorization.

M2. Merchant has a proof of transaction authorization by Customer: Here the merchant has to be ensured that the customer has indeed requested the coupons and the services. About the coupons, he can only be assured with an undeniable signature when using the 3KP protocol. In that case he has a proof, although non undeniable, for the μiKP part of the protocol too. The release by the Customer of the values $A_i(P_C)$, is the assurance for the merchant, although it is not undeniable.

C1. Unauthorized payment is not possible: As discussed in [1] the iKP protocols, in brief, ensure that unauthorized payment is impossible with non repudiation features only in 3KP. Thus one should trust the iKP part of the protocol. The second (μiKP) part bases its security on the hash chains. It assumes the preimage resistance of the hash function –that no one given the hash value $\mathcal{H}(X)$ can generate a value Y such that $\mathcal{H}(Y) = \mathcal{H}(X)$. This is requirement for a cryptographic hash function, so one should expect this part of the protocol to be secure too. So neither the merchant nor any eavesdropper that see that value $A_i(P_C)$ cannot generate $A_{i-1}(P_C)$. Thus unauthorized payment is impossible.

C2. Customer has a proof of transaction authorization by Acquirer: In the iKP *Confirm* message the Customer receives a signed by the acquirer's authorization to use the coupons

to the given merchant. This is an undeniable proof of transaction authorization.

C3. Certification and authentication of Merchant by the Customer: This depends only in the iKP part of the protocol, and is achieved when using the 2KP or 3KP versions.

C4. Customer receives a receipt from merchant: In μiKP the customer does not receive any receipt from merchant for the coupons he spent.

4.5 Summary

As we saw the properties of the protocol depend on the iKP protocol in use. There is no 1-1 mapping though between say 1KP and $\mu 1KP$, because in the plain iKP protocols these proofs apply to the price and a description of the goods, but in the μiKP case these proofs apply to a number of coupons the final value in the hash chain and the value per coupon. To sum up we list the properties of this protocol in Figure 4 and for a comparison in Figure 7 we list the properties of the iKP protocols.

Properties	$\mu 1KP$	$\mu 2KP$	$\mu 3KP$
Acquirer			
A1. Proof of Transaction Authorization by Customer	○	○	○
A2. Proof of Transaction Authorization by Merchant[10]		⋆	⋆
Merchant			
M1. Proof of Transaction Authorization by Acquirer	⋆	⋆	⋆
M2. Proof of Transaction Authorization by Customer	○	○	○
Customer			
C1. Unauthorized payment is not possible	○	○	○
C2. Proof of Transaction Authorization by Acquirer	⋆	⋆	⋆
C3. Certification and Authentication of Merchant		⋆	⋆
C4. Receipt from Merchant			

Index: ○ means limited coverage. Non-repudiation is not possible.
⋆ means full coverage. That would be by an undeniable proof.

Figure 4: A comparison of the μiKP protocols

[10] In this proof of transaction authorization the amount of money spent isn't authorized.

5 Scenario 2: Non-repeated micro-payments

5.1 The protocol

In this scenario there are four roles involved. These are the Customer, the Merchant, the Acquirer, and an additional party, the Broker. Here the same idea of repeated payments is applied but this time the Broker establishes an μiKP session with the Merchant, and the Customer with the Broker. Additionally the Customer shares a session key with the Broker[11] as well as the Broker with the Merchant. The quantities used here are listed in the table that follows.

K(CB)	The shared key between customer and broker.
K(BM)	The shared key between broker and merchant.

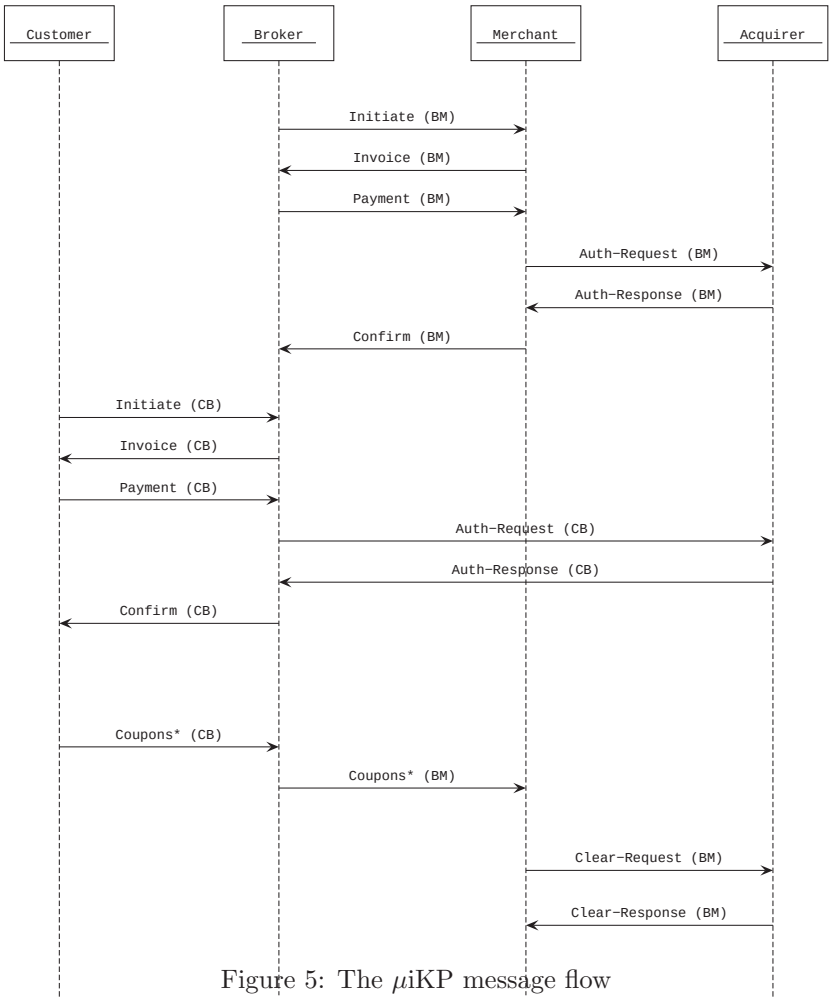

Figure 5: The μiKP message flow

5.2 μiKP message sequence

The μiKP protocol message sequence for this scenario is shown in Figure 5. We will now examine the message sequence in detail. We will discuss the new messages and not the messages that constitute a repeated μiKP payment relation, since the description is the same as in the previous discussion.

[11] This session key is not exchanged within the μiKP protocol. The authors suggest the use of SSL/TLS for that purpose.

Coupons* (CB): $\texttt{Customer} \xrightarrow{MAC_{K(CB)}(A_i(P_C),M,DESC)} \texttt{Broker}$
As in the repeated payment relation μiKP the customer sends the coupons but this time to the broker. The message contains $MAC_{K(CB)}(A_i(P_C), M, DESC)$. The $DESC$ field is the description of goods that the customer would like to get from the Merchant, M is the name of the Merchant, $A_i(P_C)$ is the next coupon for his session with the broker, and the whole message is protected by a message authentication code using the shared key between them. The message is protected to prevent modifications to the description of goods sent by the customer, and the merchant's name.

Coupons* (BM): $\texttt{Broker} \xrightarrow{MAC_{K(BM)}(A_j(P_B),C,DESC)} \texttt{Merchant}$
After having verified the coupon sent by the customer, the Broker sends a coupon of his shared session with the Merchant to the Merchant[12]. The message contains $MAC_{K(BM)}(A_j(P_B), C, DESC)$. The $DESC$ field is the same as above, C is the name of the customer, $A_j(P_B)$ is the next coupon in the session of the Broker and Merchant. The message is protected by a message authentication code using the shared key between them, for the same reasons as above.

5.3 Security properties

Let's now examine the security properties that this protocol has. In this case we will use a strict definition of transaction to avoid misunderstandings. Here as in iKP we define the transaction to be the money-goods exchange between the customer and the merchant. This is a deliberate decision since we believe that customer must authorize a particular exchange and not a blanket statement to authorize the broker to do anything with his money. We will now discuss and try to explain in details each one separately.

A1. Acquirer has a proof of transaction authorization by Customer: In this case the Acquirer only knows from the *Auth-Request (CB)* message sent by the Broker, that Customer requested some coupons to spend later using that Broker. This can be an undeniable proof In the case of 3KP since it signed by Customer's private key. But this does not constitute a transaction authorization by the Customer. Unlike the repeated payment scenario, in that case the customer hasn't even agreed on a description of the goods/services to buy, not even the merchant. So according to our definition of transaction we get no proof of transaction authorization by the client to the Acquirer.

A2. Acquirer has a proof of transaction authorization by Merchant: As in the repeated payments scenario, the acquirer can get an undeniable proof of transaction by the Merchant in the case of 2KP or 3KP.

A3. Acquirer has a proof of transaction authorization by Broker: In this case the Acquirer only knows from the *Auth-Request (BM)* message sent by the Merchant, that Broker requested coupons to spend later in this Merchant. Since there is a limit in the amount of coupons and thus their value, this can be seen as a transaction authorization.

M1. Merchant has a proof of transaction authorization by Acquirer: In general the merchant needs a proof that the customer has actually the money to spend. But in this case since he is only interacting with the Broker, so he has to be assured that the acquirer authorized the Broker. As in the repeated payments scenario, he has this guarantee with an undeniable signature of the acquirer.

[12] This can be done directly, or he can just send it to the Customer.

M2. Merchant has a proof of transaction authorization by Customer: The merchant does not receive anything from the customer that may be used as a transaction authorization. The only thing he has is the name[13] of the customer as sent by the Broker in the *Coupons* (BM)* message. Although this message is authenticated by a key shared between the merchant and the broker, it doesn't offer a proof of authorization by the customer.

M3. Merchant has a proof of transaction authorization by Broker: This is the same as in M2 of the repeated payments scenario, with the broker role exchanged by customer. So no undeniable proof is possible.

B1. Broker has a proof of transaction authorization by Acquirer: This is exactly the same as M1 of the repeated payments scenario, with the broker role exchanged by merchant. So the broker has an undeniable proof from the acquirer.

B2. Broker has a proof of transaction authorization by Customer: As in M2 of the repeated payments scenario, with the broker role exchanged by merchant, the broker has an proof from the acquirer, but not undeniable.

B3. Broker receives a receipt from Merchant: This is the same as in C4 of the repeated payments scenario, with the broker role exchanged by customer. This is not possible in μiKP protocol.

C1. Unauthorized payment is not possible: Since the customer actually runs an instance of the μiKP protocol for repeated payments, with the broker, the properties here are the same as in repeated payments scenario.

C2. Customer has a proof of transaction authorization by Acquirer: In the iKP *Confirm* message the Customer receives a signed by the acquirer's authorization to use the coupons to the given broker. This is an undeniable proof of transaction authorization.

C3. Certification and authentication of Merchant by the Customer: The customer has no communication with the merchant any more –at least as far as the μiKP protocol is concerned– thus he cannot authenticate the merchant.

C4. Customer receives a receipt from merchant: In μiKP the customer does not receive any receipt from merchant for the coupons he spent.

C6. Customer receives a receipt from broker: In μiKP the customer does not receive any receipt from broker for the coupons he spent.

C7. Certification and authentication of Broker by the Customer: This depends on the iKP protocol in use. In the case of 2KP or 3KP he gets an undeniable proof.

5.4 Summary

To sum up we list the properties of this protocol in Figure 6.

[13] and probably some other details.

Properties	$\mu 1KP$	$\mu 2KP$	$\mu 3KP$
Acquirer			
A1. Proof of Transaction Authorization by Customer			
A2. Proof of Transaction Authorization by Merchant		$\star$	$\star$
A3. Proof of Transaction Authorization by Broker		$\star$	$\star$
Merchant			
M1. Proof of Transaction Authorization by Acquirer	$\star$	$\star$	$\star$
M2. Proof of Transaction Authorization by Customer			
M3. Proof of Transaction Authorization by Broker	$\circ$	$\circ$	$\circ$
Broker			
B1. Proof of transaction authorization by Acquirer	$\star$	$\star$	$\star$
B2. Proof of transaction authorization by Customer	$\circ$	$\circ$	$\circ$
B3. Receipt from Merchant			
Customer			
C1. Unauthorized payment is not possible	$\circ$	$\circ$	$\circ$
C2. Proof of Transaction Authorization by Acquirer	$\star$	$\star$	$\star$
C3. Certification and Authentication of Merchant			
C4. Receipt from Merchant			
C6. Receipt from Broker			
C7. Certification and Authentication of Broker		$\star$	$\star$

Index: $\circ$ means limited coverage. Non-repudiation is not possible.
$\star$ means full coverage. That would be by an undeniable proof.

Figure 6: A comparison of the μiKP protocols

6 Overview

By comparing the security properties of μiKP for repeated transactions to the same merchant, as shown in Figure 4 and the properties of iKP , we can conclude that some requirements are now more relaxed. These are:

- Acquirer no longer has an undeniable proof of transaction authorization by customer in 3KP.
- Merchant no longer has an undeniable proof of transaction authorization by customer in 3KP.
- Unauthorized payment is not undeniable any more in 3KP.
- No possibility for receipt in all iKP protocols.

However we can also see that for the cases of 1KP and 2KP, the merchant can get a proof of transaction authorization by the customer, although he hadn't this possibility in plain iKP . On the other hand the μiKP protocol for non-repeated transactions using a broker, has more weakend properties. We can see that:

- The acquirer no longer gets a proof of transaction authorization by the customer.
- The merchant no longer gets a proof of transaction authorization by the customer in 3KP.
- Unauthorized payment is not undeniable any more in 3KP.
- The customer can no longer authenticate the merchant.
- The customer can no longer receive receipt from the merchant.

Especially the first one in the list is important for the security of such a protocol. Given the fact that in practice we expect the latter μiKP protocol to be deployed and not the repeated payments scenario, we believe that this is a serious drawback.

Requirements	1KP	2KP	3KP
Acquirer			
A1. Proof of Transaction Authorization by Customer	∘	∘	⋆
A2. Proof of Transaction Authorization by Merchant		⋆	⋆
Merchant			
M1. Proof of Transaction Authorization by Acquirer	⋆	⋆	⋆
M2. Proof of Transaction Authorization by Customer			⋆
Customer			
C1. Unauthorized payment is not possible	∘	∘	⋆
C2. Proof of Transaction Authorization by Acquirer	⋆	⋆	⋆
C3. Certification and Authentication of Merchant		⋆	⋆
C4. Receipt from Merchant		⋆	⋆

Index: { ∘ means limited coverage. Non-repudiation is not possible.
⋆ means full coverage. That would be by an undeniable proof.

Figure 7: A comparison of the iKP protocols

A Hash chains

The basis of the μiKP protocols is a constructions of cryptographic hash functions called hash chains. This construction exploits the preimage resistance (onewayness) of cryptographic hash functions. A brief description of this algorithm is given below.

Say we want to authenticate two parties Alice and Bob. Given a hash function $\mathcal{H}$, and a random value X Alice generates a large sequence of hashes, say n. Thus if we name A_i the i-th hash we have

$$\begin{aligned} A_0(X) &= X \\ A_1(X) &= \mathcal{H}(A_0(X)) \\ &\vdots \\ A_{n-1}(X) &= \mathcal{H}(A_{n-2}(X)) \\ A_n(X) &= \mathcal{H}(A_{n-1}(X)) \end{aligned}$$

This sequence of hashes can be used for authentication in the following way. Firstly there must be a secure exchange of the last hash ($A_n(X)$) between the two parties. This exchange doesn't have to be secret. Then each time Alice wants to prove her identity to Bob she sends a previous hash in the chain. For example, the first time she sends A_{n-1}, the second A_{n-2} and so on. Bob can easily verify Alice's proof of identity by checking $\mathcal{H}(A_{n-1}(X)) = A_n(X)$, $\mathcal{H}(\mathcal{H}(A_{n-2}(X))) = A_n(X)$ etc. In order for Bob to prevent replay attacks, he has to keep somewhere a counter, or the last received hash and check if the new one is the previous to the one in his database.

References

[1] M. Bellare et al. "iKP - A family of secure electronic payment systems", 1995

[2] R. Hauser et al. "Micro-Payments based on iKP", 1996