Multi-Vendor PayWord with Payment Approval

Andrea Huszti

Faculty of Informatics
University of Debrecen
Debrecen, Hungary
huszti.andrea@inf.unideb.hu

Abstract. We extended the PayWord scheme, it supports shopping at multiple vendors without an on-line broker or an on-line secure database. The credit-based system uses one hash chain, hence besides the secret key only the seed and a random value should be securely stored. The proposed scheme is analyzed in applied pi calculus, we prove that it fulfills payment approval, secure payment authorization, secrecy of payment information and unreusability.

1 Introduction

In recent years electronic commerce has grown rapidly as Internet and web technologies have progressed. Usually content and service providers charge very small amount (e.g. less than a dollar), hence a special payment solution is required, called micropayment. Compared to macropayment schemes, the computational time of micropayment schemes are less because it uses the one-way, collision-resistant hashing extensively but not public key cryptosystems. One can find more information about micropayments in [11].

One of the most well known micropayment scheme is the PayWord scheme [7]. PayWord is designed to be one-vendor, thus if we apply it for multiple vendors, it does not protect against double spending. There are three main categories of multi-vendor solutions with double-spending prevention. Some schemes require an on-line broker for changing vendors [6], these are called semi-online systems. Some solutions expect an on-line database [5] that is available for all vendors. Schemes from the third category use black lists [9], [10], meaning brokers maintain a list of users who did not pay for the products or services.

We have introduced definition of payment approval in [3], where we extended Payword that it avoids misunderstandings between the consumer and the vendor that would cause penalty for the vendor or ruin its reputation. In case of micropayment schemes costs should be minimized, involving an on-line broker or maintaining an on-line database increase expenses and black lists do not prevent the first violations. We modified PayWord in a way that it is applicable for multiple vendors and also provides payment approval. We did not increase the number of asymmetric key operations or the number of asymmetric key pairs, we use only one hash chain that requires two random values stored securely.

Electronic payment systems have critical security requirements, hence the design of its cryptographic protocol requires special care. Substantial evaluation should be provided in order to prevent flaws. Although there are several PayWord based scheme ([5],[6],[9],[10]), they do not give formal security evaluation. PayWord is formalized in spi calculus in [2] and we gave proof for payment approval, payment authorization in applied pi calculus with the help of ProVerif [4] in [3]. In case of the proposed scheme besides payment approval and payment authorization, we also provide proof for double spending detection with labeled semantics.

2 Payment approval for PayWord

Let us review the extended PayWord micropayment scheme according to [3]. We have three participants: a user U, a vendor V and a broker B. The protocol consists of three phases: user-broker, user-vendor and broker-vendor relationship. We should emphasize that only the user-vendor relationship is short-term the others are assumed to be long-term. According to basic PayWord [7] user-broker and broker-vendor relationship could happen off-line. We consider the case when U registers on-line.

Let us describe the extended PayWord micropayment scheme in details. We employ digital signatures, where user's and broker's public and secret keys are denoted by K_U^+, K_B^+ and K_U^-, K_B^-, respectively. $\{M\}_{K_i^-}$ denotes a message signed by K_i^-, where $i \in \{U, B\}$.

User-Broker relationship

User U initiates a relationship with broker B by requesting an authorized PayWord Certificate. U transports his credit-card number and his public key K_U^+ on an authenticated encrypted channel.

$$1.\ U \rightarrow B : U, K_U^+$$

B digitally signs B, U, K_U^+ and E with key K_B^-, where E denotes the expiration date of the certificate. This certificate ensures any vendor that the amount will be paid-off before date E, since U is able to cover it.

$$2.\ B \rightarrow U : \{B, U, K_U^+, E\}_{K_B^-}$$

User-Vendor relationship

For achieving payment approval we have extended the basic PayWord scheme with vendor authentication. Secret MAC key is sent to the vendor encrypted. This solution does not increase the complexity significantly, since public key encryption should be run only once, at the beginning of the relationship.

Before shopping U generates a *payword chain*: $w_0, w_1, \ldots w_n$. First U generates a random number w_n, then calculates

$$w_i = H(w_{i+1}),$$

where $i \in \{n-1, n-2, \ldots, 0\}$. We call $w_1, \ldots w_n$ paywords, w_0 is the root of the chain. U chooses the number n arbitrarily beyond the requested amount and generates a certificate containing the vendor's identification information V, his PayWord Certificate, the commitment w_0, the actual date D and the MAC key encrypted. This certificate is signed by the user's secret key K_U^-.

$$3\ U \to V : \{V, \{B, U, K_U^+, E\}_{K_B^-}, w_0, D, \{K\}_{K_V^+}\}_{K_U^-}$$

The vendor verifies U's signature by public key K_U^+, and B's signature by K_B^+, checks whether D is before E and stores w_0 with the user information.

Following U sends order information and a MAC proof for the vendor containing order information and a payword. Then the vendor sends the requested product on an encrypted channel without verifying the payword, we should mention that MAC verification happens after the user receives his product and sends the payment. In case of micropayments loosing the amount of one payment is not a large loss, if either the hash or the MAC verification is unsuccessful, then the vendor might decide to refuse other purchases. A payment is a pair of a payword and the corresponding index (w_i, i), where $i \in \{1, 2, \ldots, n\}$. It is important that the user sends his paywords starting from w_1, the w_2 and so on.

$$\begin{gathered}
4.1\ U \to V : OrderInf_1, Mac((OrderInf_1, w_1), K) \\
V \to U : Product_1 \\
U \to V : (w_1, 1) \\
\vdots \\
4.n\ U \to V : OrderInf_l, Mac((OrderInf_l, w_l), K) \\
V \to U : Product_l \\
U \to V : (w_l, l)
\end{gathered}$$

Vendor V verifies the received payword w_i by applying i times the hash function on it, *i.e.* checks $H^i(w_i) = w_0$, in case of the first shopping, otherwise verifies w_i with the stored payword w_j, where $j < i$, *i.e.* checks $H^{i-j}(w_i) = w_j$. V also verifies correctness of the MAC value, with help of order information and the payword.

Vendor-Broker relationship

V sends all necessary information to B for the pay-off. V transmits the certificate generated by U, the last payword received from U and the corresponding index.

$$5.\ V \to B : \{V, \{B, U, K_U^+, E\}_{K_B^-}, w_0, D\}_{K_U^-}, w_l, l$$

B verifies the signature of user's certificate with K_U^+, checks whether identity information of the vendor received matches with V, the expiring date and validity of the payword, *i. e.* $H^l(w_l) = w_0$. If all verifications hold, then B pays the proper amount to V.

3 Shopping at Multiple Vendors

Users in advance should decide about the vendors they would like to connect with and generate their hash chain. The hash chain is generated in a way that even if a vendor receives a hash value, he is not able to calculate other vendors' paywords. The user asks the broker to authorize his hash chain commitments to prove that he can cover his purchases. The system is credit-based, hence users are allowed to pay for as many as paywords posteriorly as he used up. We do not need the broker to be on-line, since his authorization is necessary only during preparation and only once. Users are allowed to spend each payword only at the vendor that the corresponding commitment assigned to, the vendor can verify whether a payword is spent before or not, hence we do not need an on-line database.

We describe the modified scheme in details. Let us denote the vendors by $V_1, V_2, \ldots, V_j$, the user and the broker by U and B, respectively. The user possesses a (K_U^-, K_U^+) signature key pair for authentication, the broker also has a (K_B^-, K_B^+) signature key pair for generating a certificate verifying that U is able to afford his shopping and each vendor has a $(K_{V_i}^-, K_{V_i}^+)$ public key encryption key pair for key exchange. We would like to mention, that we did not increase the number of keys necessary for multiple vendor shopping comparing to the one vendor one.

We apply one payword hash chain per user, hence each user generates a random seed w_n and calculates hash values. The same hash chain is applied for all vendors, hence users need to generate another random value denoted by v. Both random values are kept secret and stored securely. The hash chain is generated as follows.

Each user should decide about the number of paywords he would like to spend at each vendor. Let us assume that U would like to spend k_i paywords at V_i, where $i = 1, \ldots, j$, hence $w_n = w_{j,k_j}$. In general

$$w_{i,l} = \begin{cases} H(Enc_v(w_{i+1,0})), \text{ if } l = k_i, \\ H(w_{i,l+1}), \qquad\quad \text{if } 0 \leq l < k_i. \end{cases}$$

Paywords $w_{i,1}, w_{i,2}, \ldots, w_{i,k_i}$ can be spent at vendor V_i, the corresponding commitment is $w_{i,0}$. Since the last payword U is able to spend depends on v, hence V_{i+1} cannot calculate $w_{i,1}, w_{i,2}, \ldots, w_{i,k_i}$.

User-Broker relationship

After generating the payword chain, U transports his credit-card number, his public key K_U^+, the list of vendors $V_1, V_2, \ldots, V_j$ and the list of commitments $w_{1,0}, w_{2,0}, \ldots, w_{j,0}$ on an authenticated encrypted channel to B.

$$1.\ U \rightarrow B : U, K_U^+, (V_j, w_{j,0}), \ldots, (V_2, w_{2,0}), (V_1, w_{1,0})$$

B digitally signs all data received, B and E with key K_B^-, where E denotes the expiration date of the certificate.

$$2.\ B \rightarrow U : \{B, U, K_U^+, E, (V_1, w_{1,0}), \ldots, (V_j, w_{j,0})\}_{K_B^-}$$

The certificate received from B is called PayWord certificate, and we denote it by C_U. The main difference between the one-vendor and multi-vendor solutions is that commitments are signed by the broker. Each vendor is able to verify all the paywords sent by the user with the help of these commitments. The system is credit-based, hence the user is able to spend a certain amount of money in advance and later, e.g. once a month, refunds it to the broker.

User-Vendor relationship

Users can spend his paywords at different vendors. U chooses a vendor V_i and generates a fresh symmetric encryption key KV_i. The user sends his PayWord certificate, the actual date and the encrypted symmetric key digitally signed.

$$3.\ U \to V_i : \{C_U, D, \{KV_i\}_{K_{V_i}^+}\}_{K_{U^-}}$$

The vendor verifies U's signature by public key K_U^+, the PayWord certificate by K_B^+. V decrypts the symmetric key KV_i and if D is before E, then stores $(w_{i,0}, 0)$, U and KV_i. User sends the corresponding order information and the proof of what he would like to buy and for what price. The proof is a MAC value generated with the hash of KV_i. Vendor sends the encrypted product, we might think of a song or an article. After receiving the product the user sends the next payword.

$$4.1\ U \to V_i : OrderInf_1, Mac((OrderInf_1, w_1), H(KV_i))$$
$$4.2\ V_i \to U : \{Product_1\}_{KV_i}$$
$$4.3\ U \to V_i : (w_1, 1)$$

V verifies whether the MAC value is valid, *i.e.* the proof received before is the MAC value of the proper order information and the payword. Paywords as hash values prove that they are fresh and originate from the user. V also verifies whether the hash value of the payword received equals to the one stored in his database. If all verifications are correct, then modifies the payword stored to the new one and updates the index.

$$\vdots$$

$$4.k_i\ U \to V_i : OrderInf_{k_i}, Mac((OrderInf_{k_i}, w_{k_i}), H(KV_i))$$
$$V_i \to U : \{Product_{k_i}\}_{KV_i}$$
$$U \to V_i : (w_{k_i}, k_i)$$

Vendor-Broker relationship

Vendor V_i sends the PayWord certificate, the payword stored with the corresponding index and the actual date to B.

$$5.\ V_i \to B : \{C_U, D, \{KV_i\}_{K_{V_i}^+}\}_{K_{U^-}}, (w_{k_i}, k_i), V_i$$

B verifies the user's PayWord certificate, checks its validity according to the date D and calculates $H^{k_i}(w_{k_i})$, whether it is $w_{i,0}$. If all values are valid, then B pays the proper amount to V_i.

4 Applied pi calculus

The applied pi calculus is based on the pi calculus. Detailed description of this topic can be found in [1]. The calculus assumes an infinite set of names, an infinite set of variables and a signature. A signature $\sum$ is a set of function symbols, each with an arity. A function symbol f with arity 0 is a constant symbol. Let us denote channel names by a, b, c, and names of any sort by m, n. Also, let x, y, z range over variables.

$L, M, N, T, U, V ::=$	**terms**
$a, b, c, \ldots, k, \ldots, m, n, \ldots, s$	name
x, y, z	variable
$f(M_1, ..., M_l)$	function application

The grammar for processes is the following:

$P, Q, R ::=$	**processes**
$\mathbf{0}$	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction (new)
$if\ M = N\ then\ P\ else\ Q$	conditional
$u(x).P$	message input
$\overline{u}\langle N\rangle.P$	message output

Replication of process P means infinite number of copies of P running in parallel. Name restriction process $\nu n.P$ creates a new, private name and behaves as P. Finally, process $u(x).P$ is ready to input from channel u, then to run P with the message replaced for the formal parameter x, and process $\overline{u}\langle N\rangle.P$ is ready to output N on channel u, then to run P. We extend processes with active substitutions. We denote the substitution that replaces the variable x with the term M by $\{M/x\}$. With active substitutions we capture the knowledge exposed to the adversarial environment. For modeling multi-vendor PayWord scheme, we add a restriction to active substitution as follows: $\nu x.(\{M/x\}|P)$ that corresponds to $let\ x = M\ in\ P$.

A frame is an extended process built from 0 and active substitutions, which are composed by parallel composition and restriction. Every extended process A can be mapped to its frame $\varphi(A)$ by replacing every plain process with 0. The frame represents the static knowledge output to the environment.

4.1 Operational semantics

The signature is equipped with an equational theory E, that is a set of equations of the form $M = N$, where terms M, N are defined over the signature. Equality is the smallest equivalence relation on terms, that contains E and is closed under application of function symbols, substitutions of terms for variables and bijective

renaming of names. We give formalization for cryptographic primitives with the help of equational theory as follows:

Hash function, message authentication code. We represent a oneway hash function as a unary function symbol H with no equations. The absence of an inverse for H models the onewayness of H. Similarly we denote a oneway MAC function as a binary function symbol Mac, where the second argument corresponds to the secret key of MAC.

$fun\ H/1.$
$fun\ Mac/2.$

Symmetric encryption. We take binary function symbols *senc* and *sdec* for encryption and decryption, respectively, with the equation: $sdec(senc(x, y), y) = x$. Here x represents the plaintext and y the secret key.

$fun\ senc/2.$
$reduc\ sdec(senc(x, y), y) = x.$

Asymmetric encryption. In case of asymmetric encryption we have to generate a keypair, a public and a secret key. We have an unary function symbol pk for generating the public key, where the secret key is the argument. Similarly to symmetric encryption we represent asymmetric encryption and decryption with binary function symbols *aenc* and *adec* with the equation of $adec(aenc(x, pk(y)), y) = x$, where x denotes the plaintext and y is the secret key.

$fun\ pk/1.$
$fun\ aenc/2.$
$reduc\ adec(aenc(x, pk(y)), y) = x.$

Digital signatures. In order to formalize digital signatures that also employ secret and public keys we use function symbol pk for generating public keys, and binary function symbols *sign*, *checksign*. We interpret digital signatures with message recovery, meaning we have equation $checksign(sign(m, k), pk(k)) = m$, where m is the message and k is the secret key.

$fun\ pk/1.$
$fun\ sign/2.$
$reduc\ checksign(sign(m, k), pk(k)) = m.$

We define context $C[\_]$ to be an extended process with a hole. An evaluation context is a context whose hole is not in the scope of a replication, a conditional, an input, or an output.

Furthermore operational semantics is defined in terms of structural equivalence ($\equiv$), internal reduction ($\rightarrow$) and labeled reductions. Structural equivalence captures rearrangements of parallel compositions and restrictions and the equational rewriting of the terms in a process. Internal reduction defines the semantics of process synchronization and conditionals. For detailed description we refer to [7].

While internal reduction rules are applied for executing a process without contact with its environment, labeled semantics enables us to reason about the

interaction between processes and the environment. We have a ternary relation $A \xrightarrow{\alpha} B$, where α is a label of the form $a(M)$, $\overline{a}\langle u\rangle$, $\nu u.\overline{a}\langle u\rangle$ such that u is either a channel name or a variable. The following rules are given in the labeled operational semantics:

$$\textsc{In} \qquad a(x).P \xrightarrow{a(M)} P\{M/x\}$$

$$\textsc{Out-Atom} \qquad \overline{a}(u).P \xrightarrow{\overline{a}\langle u\rangle} P$$

$$\textsc{Open-Atom} \qquad \frac{A \xrightarrow{\overline{a}\langle u\rangle} A' \;\; u \neq a}{\nu u.A \xrightarrow{\nu u \overline{a}\langle u\rangle} A'}$$

$$\textsc{Scope} \qquad \frac{A \xrightarrow{\alpha} A' \;\; u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

$$\textsc{Par} \qquad \frac{A \xrightarrow{\alpha} A' \;\; bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = 0}{A|B \xrightarrow{\alpha} A'|B}$$

$$\textsc{Struct} \qquad \frac{A \equiv B \;\; B \xrightarrow{\alpha} B' \;\; B' \equiv A'}{A \xrightarrow{\alpha} A'}$$

We should mention that $\overline{c}\langle M\rangle.P \xrightarrow{\nu x.\overline{c}\langle x\rangle} P \mid \{M/x\}$. For the full derivation see [8].

5 Formal security evaluation

5.1 Modeling Multi-Vendor PayWord

The main process.

As a first step secret and public keys are generated and we issue identity numbers by applying function host for all participants. Identity information and public keys are made public.

$$\begin{aligned}
\text{process} \triangleq\ & \nu ssk_U.\nu ssk_B.\nu esk_B.\nu esk_{V_1}.\ \ldots\ .\nu esk_{V_j}.\\
& \text{let } spk_U = pk(ssk_U) \text{ in let } spk_B = pk(ssk_B) \text{ in let } epk_B = pk(esk_B) \text{ in}\\
& \text{let } epk_{V_1} = pk(esk_{V_1}) \text{ in } \ldots \text{ let } epk_{V_j} = pk(esk_{V_j}) \text{ in}\\
& \text{let } hU = host(spk_U) \text{ in let } hB = host(spk_B) \text{ in}\\
& \text{let } hV_1 = host(epk_{V_1}) \text{ in } \ldots \text{ let } hV_j = host(epk_{V_j}) \text{ in}\\
& \overline{a}\langle hB, epk_B, spk_B\rangle.\overline{a}\langle hU, spk_U\rangle.\overline{a}\langle hV_1, epk_{V_1}, \ldots, hV_j, epk_{V_j}\rangle.\\
& ((!processU) \mid (!processB) \mid (!processV_1) \mid \ldots \mid (!processV_j))
\end{aligned}$$

The user process.

Our user-broker relationship is managed on-line. We define several events. $UacceptsB$ event happens after authenticating the broker via asymmetric key decryption, $ValidCertificate$ proceeds if the certificate received from the broker is valid. $V_iReadytoPurchase$ runs at the beginning of the shopping process with V_i and $UV_iEndsPayment_{k_i}$ happens after transferring the k_ith payword.

$$
\begin{aligned}
\text{processU} \triangleq\ & \nu w_{j,k_j}.\nu v.\ \text{let } w_{j,k_{j-1}} = H(w_{j,k_j}) \text{ in let } w_{j,0} = H(w_{j,1}) \text{ in} \\
& \text{let } w_{j-1,k_{j-1}} = H(senc(w_{j,0}, v)) \text{ in } \ldots \text{ let } w_{1,0} = H(w_{1,1}) \text{ in} \\
& \nu K.\overline{a}\langle aenc(sign((K, hU, hB), ssk_U), epk_B), hU\rangle.a(mes). \\
& \text{let } (n, = hU) = sdec(mes, K) \text{ in } \overline{UacceptsB}\langle hU, hB, K, n\rangle. \\
& \overline{a}\langle senc((CardInf, hU, n, w_{j,0}, hV_j, \ldots, w_{2,0}, hV_2, w_{1,0}, hV_1), K)\rangle. \\
& a(CU).\ \text{let } (= hB, = hU, E, = w_{j,0}, = hV_j, \ldots, = w_{1,0}, = hV_1) = \\
& = checksign(CU, spk_B) \text{ in } \overline{ValidCertificate}\langle hU, hB, K, n\rangle. \\
& (V_j shopping | V_{j-1} shopping | \ldots | V_2 shopping | V_1 shopping)
\end{aligned}
$$

$$
\begin{aligned}
V_i shopping \triangleq\ & a(s, = hV_i).\nu KV_i.\overline{V_i ReadytoPurchase}\langle hU, hV_i\rangle. \\
& \overline{a}\langle sign((CU, D, s, aenc(KV_i, epk_{V_i})), ssk_U)\rangle. \\
& \overline{a}\langle OrderInf_1, MAC((OrderInf_1, w_{i,1}), H(KV_i))\rangle. \\
& a(msg_1).\text{let } page_1 = sdec(msg_1, KV_i) \text{ in} \\
& \overline{a}\langle w_{i,1}\rangle. \\
& \ldots \\
& \overline{a}\langle OrderInf_{k_i}, MAC((OrderInf_{k_i}, w_{i,k_i}), H(KV_i))\rangle. \\
& a(msg_{k_i}).\text{let } page_{k_i} = sdec(msg_{k_i}, KV_i) \text{ in} \\
& \overline{a}\langle w_{i,k_i}\rangle.\overline{UV_i EndsPayment_{k_i}}\langle hU, hV_i\rangle
\end{aligned}
$$

The broker process. Event $BconnectstoU$ happens right before the broker establish communication with the user. $BacceptsU$ comes after authenticating the user via symmetric key decryption. Finally $BpaystoV_i$ proceeds if all data sent by vendor V_i is valid.

$$
\begin{aligned}
\text{processB} \triangleq\ & a(s, = hU).\text{let } sig = adec(s, esk_B) \text{ in} \\
& \text{let } (K, = hU, = hB) = checksign(sig, spk_U) \text{ in} \\
& \nu n.\overline{BconnectstoU}\langle hU, hB, K, n\rangle. \\
& \overline{a}\langle senc((n, hU), K))\rangle.a(mes).\text{let } inf = sdec(mes, K) \text{ in} \\
& \text{let } (CardInf, = hU, = n, w_{j,0}, hV_j, \ldots, w_{1,0}, hV_1) = sdec(mes, K) \text{ in} \\
& \overline{BacceptsU}\langle hU, hB, K, n\rangle. \\
& \overline{a}\langle sign((hB, hU, E, w_{j,0}, hV_j, \ldots, w_{1,0}, hV_1), ssk_B))\rangle. \\
& (V_j chargeoff | V_{j-1} chargeoff | \ldots | V_2 chargeoff | V_1 chargeoff)
\end{aligned}
$$

$$
\begin{aligned}
V_i chargeoff \triangleq\ & a(signMU, w_{k_i}, k_i, = hV_i).let(CU, D, x, y) = checksign(signMU, spk_U) \text{ in} \\
& \text{let } (= hB, = hU, = E, = w_{j,0}, = hV_j, \ldots, = w_{1,0}, = hV_1)) = \\
& = checksign(CU, spk_B) \text{ in} \\
& \text{if } H^{k_i}(w_{k_i}) = w_{i,0} \text{ then } \overline{BpaystoV_i}\langle hV_i, hB\rangle
\end{aligned}
$$

The vendor process.

$V_i AcceptsMU$ denotes the event that vendor V_i successfully authenticated the user via digital signature and the payword certificate is valid.

$V_i AcceptsPayment_l$ happens if the proof of payment approval and the lth payword are valid. $V_i RejectsPayment_l$ runs if the payword received is not valid.

$processV_i \triangleq \nu s.\overline{a}\langle s, hV_i\rangle.a(signMU).$
let $(CU, D, = s, enckey) = checksign(signMU, spk_U)$ in
let $(= hB, = hU, E, w_{j,0}, hV_j, \ldots, w_{i,0}, = hV_i, \ldots, w_{1,0}, hV_1) =$
$= checksign(CU, spk_B)$ in let $KV_i = adec(enckey, esk_{V_i})$ in
$\overline{V_i AcceptsMU}\langle hU, hV_i\rangle.a(OrderInf_1, proof_1).$
$\overline{a}\langle senc(page_1, KV_i)\rangle.a(pw_1).$ if $H(pw_1) = w_{i,0}$ then
if $proof_1 = MAC((OrderInf_1, pw_1), H(KV_i))$ then
$\overline{V_i AcceptsPayment_1}\langle hU, hV_i\rangle.$
$\ldots$
$a(OrderInf_k, proof_k).\overline{a}\langle senc(page_k, KV_i)\rangle.a(pw_k).$
if $H(pw_k) = pw_{k-1}$ then
if $proof_k = MAC((OrderInf_k, pw_k), H(KV_i))$ then
$\overline{V_i AcceptsPayment_k}\langle hU, hV_k\rangle.$
$\ldots$
$a(OrderInf_{k_i}, proof_{k_i}).\overline{a}\langle senc(page_{k_i}, KV_i)\rangle.a(pw_{k_i}).$
if $H(pw_{k_i}) = pw_{k_i-1}$ then
if $proof_{k_i} = MAC((OrderInf_{k_i}, pw_{k_i}), H(KV_i))$ then
$\overline{V_i AcceptsPayment_{k_i}}\langle hU, hV_i\rangle.\overline{a}\langle signMU, pw_{k_i}, k_i, hV_i\rangle$
else 0 else $\overline{V_i RejectsPayment_{k_i}}\langle hU, KV_i, pw_{k_i}\rangle$
$\ldots$
else 0 else $\overline{V_i RejectsPayment_1}\langle hU, KV_i, pw_1\rangle$

5.2 Security analysis

In this section we prove that our proposed scheme accomplish secure payment authorization, payment approval, secrecy of payment information and unreusability. First of all we recall security definitions given in [3], then we give a formal proof employing applied pi calculus. We also prove unreusability, *i.e.* double-spender detection employing labeled semantics.

Payment authorization. Payment authorization guarantees a proof for the vendor, that there is sufficient fund on the user's account. This proof or certificate is created by the broker.

Definition 1. *We state that a payment scheme fulfills payment authorization if the following conditions hold:*

1. *Broker authentication: The consumer successfully authenticates the broker, it is indisputable that the certificate is authorized by the broker.*
2. *Consumer authentication: The broker successfully authenticates the consumer, it is indisputable that the consumer's account is questioned.*
3. *Certificate: There is a proof for the vendor, that the sufficient fund is available.*

Secure *payment authorization is achieved, if the certificate is undeniable.*

Payment approval. Payment approval process generates a proof for the vendor that a consumer agrees to pay a certain amount of money for a particular product. We would like to emphasize requisiteness of vendor authentication, since if the consumer sends the proof to an adversary, then the adversary might be able to masquerade the consumer.

Definition 2. *We state that a payment scheme fulfills payment approval if the following conditions hold:*

1. *Consumer authentication: The vendor successfully authenticates the consumer, it is indisputable that the proof originates from the consumer.*
2. *Vendor authentication: The consumer successfully authenticates the vendor, it is indisputable that the certificate is sent to the vendor.*
3. *Order information: The proof contains precise description of the product.*
4. *Price information: The proof contains the amount of money the consumer tends to pay.*

Secure *payment approval is achieved, if the proof is undeniable.*

Secrecy of payment information. In case of payment schemes it is crucial that payment information such as credit card information should be kept secret.

Definition 3. *We state that a payment scheme possesses secrecy of payment information, if confidential payment information is not revealed for adversaries.*

Unreusability. Multi-vendor schemes should be protected against double spenders.

Definition 4. *If the same coin is spent more than once, then the identity of the second spender should be detected.*

Theorem 1. *Our proposed multi-vendor PayWord scheme accomplish secure payment authorization, payment approval, secrecy of payment information and unreusability.*

Proof. We give a proof for secure payment authorization and secrecy of payment information in the general case with the help of ProVerif. In order to prove payment approval ProVerif queries for a concrete case, when there are two vendors with $k_1 = 4$, $k_2 = 3$ are considered. Since the payment phase is the same for each purchase considering these queries is sufficient for us.

Our scheme fulfills *secure payment authorization*, since it provides an undeniable certificate: $sign((hB, hU, E, w_{j,0}, hV_j, \ldots, w_{1,0}, hV_1), ssk_B))$. Broker and consumer authentication are achieved, since the following queries return logical value true:
$query\ evinj : ValidCertificate(x, y, z, t) \Longrightarrow evinj : BconnectstoU(x, y, z, t)$.
$query\ evinj : BacceptsU(x, y, z, t) \Longrightarrow (evinj : UacceptsB(x, y, z, t) \Longrightarrow evinj : BconnectstoU(x, y, z, t))$.

Our scheme possesses *secrecy of payment information*, since the following query returns true:

$query\ attacker : CardInf$.

Our scheme fulfills *payment approval*, since it provides a proof of order and price information for each purchase: $MAC((OrderInf_l, w_{i,l}), H(KV_i))$ and fulfills consumer and vendor authentication, since the following queries return true.
$query\ evinj : V1AcceptsPayment2(x, y) \Longrightarrow evinj : V1ReadytoPurchase(x, y)$.
$query\ evinj : V2AcceptsPayment2(x, y) \Longrightarrow evinj : V2ReadytoPurchase(x, y)$.

$query\ evinj : UV1EndsPayment2(x, y) \Longrightarrow evinj : V1AcceptsMU(x, y).$
$query\ evinj : UV2EndsPayment2(x, y) \Longrightarrow evinj : V2AcceptsMU(x, y).$

Unreusability. We consider a dishonest user, an attacker with the largest power, who intends to deposit the same coin twice either at the same, or at different vendors. We prove that in this case the identity of the double-spender is revealed. Active attackers with less power are handled similarly. Users' shopping processes:

$$
\begin{aligned}
V_i shopping\_ds \triangleq\ & a(s, = hV_i).\nu KV_i.\\
& \overline{a}\langle sign((sign((hB, hU, E, w_{j,0}, hV_j, \ldots, H^{k+l}(w_{i,k+l}),\\
& hV_i, \ldots, w_{1,0}, hV_1), ssk_B)), D, s, aenc(KV_i, epk_{V_i})), ssk_U)\rangle.\\
& \overline{a}\langle OrderInf_1, MAC((OrderInf_1, H^{k+l-1}(w_{i,k+l})), H(KV_i))\rangle.\\
& a(msg_1).\text{let } page_1 = sdec(msg_1, KV_i) \text{ in } \overline{a}\langle H^{k+l-1}(w_{i,k+l})\rangle.\\
& \ldots\\
& \overline{a}\langle OrderInf_k, MAC((OrderInf_k, H^{l}(w_{i,k+l})), H(KV_i))\rangle.\\
& a(msg_k).\text{let } page_k = sdec(msg_k, KV_i) \text{ in } \overline{a}\langle H^{l}(w_{i,k+l})\rangle.\\
& \ldots\\
& \overline{a}\langle OrderInf_{k+l}, MAC((OrderInf_{k+l}, H^{l}(w_{i,k+l})), H(KV_i))\rangle.\\
& a(msg_{k+l}).\text{let } page_{k+l} = sdec(msg_{k+l}, KV_i) \text{ in } \overline{a}\langle H^{l}(w_{i,k+l})\rangle\\
V_{i,i-1} shopping\_ds \triangleq\ & a(s, = hV_i).\nu KV_i.\\
& \overline{a}\langle sign((sign((hB, hU, E, w_{j,0}, hV_j, \ldots, H^{k+l}(w_{i,k+l}),\\
& hV_i, H^{k_{i-1}}(Enc(H^{k+l}(w_{i,k+l}), v)), hV_{i-1}, \ldots, w_{1,0}, hV_1), ssk_B)),\\
& D, s, aenc(KV_i, epk_{V_i})), ssk_U)\rangle.\\
& \ldots\\
& \overline{a}\langle OrderInf_k, MAC((OrderInf_k, H^{l}(w_{i,k+l})), H(KV_i))\rangle.\\
& a(msg_k).\text{let } page_k = sdec(msg_k, KV_i) \text{ in } \overline{a}\langle H^{l}(w_{i,k+l})\rangle.\\
& a(r, = hV_{i-1}).\nu KV_{i-1}.\\
& \overline{a}\langle sign((sign((hB, hU, E, w_{j,0}, hV_j, \ldots, H^{k+l}(w_{i,k+l}),\\
& hV_i, H^{k_{i-1}}(Enc(H^{k+l}(w_{i,k+l}), v)), hV_{i-1}, \ldots, w_{1,0}, hV_1), ssk_B)),\\
& D, r, aenc(KV_{i-1}, epk_{V_{i-1}})), ssk_U)\rangle.\\
& \overline{a}\langle OrderInf_1, MAC((OrderInf_1, H^{l}(w_{i,k+l})), H(KV_{i-1}))\rangle.\\
& a(msg_1).\text{let } page_1 = sdec(msg_1, KV_{i-1}) \text{ in } \overline{a}\langle H^{l}(w_{i,k+l})\rangle
\end{aligned}
$$

First we study the one vendor - double spender case. Let our evaluation context be: $C[\_] = \nu ssk_U.\nu ssk_B.\nu esk_{V_i}(\_|(!V_i shopping\_ds)|(!processV_i))$ and our process is: $P \equiv C[\overline{a}\langle hB, pk(ssk_B)\rangle.\overline{a}\langle hU, pk(ssk_U)\rangle.\overline{a}\langle hV_i, pk(esk_{V_i})\rangle]$. We show that $\varphi(P') \vdash (hU, KV_i, H^l(w_{i,k+l}))$, that gives the identity of the double-spender, its secret MAC key and the invalid payword. Let us consider the vendor's process only with event $V_i RejectsPayment_k(hU, KV_i, H^l(w_{i,k+l}))$ and deal with the following execution path:

$$
\begin{aligned}
& P \xrightarrow{\nu b\_pk.\overline{a}\langle b\_pk\rangle} \xrightarrow{\nu u\_pk.\overline{a}\langle u\_pk\rangle} \xrightarrow{\nu v\_pk.\overline{a}\langle v\_pk\rangle} \xrightarrow{\nu x.\overline{a}\langle x\rangle} \xrightarrow{a(s, hV_i)} \xrightarrow{\nu y.\overline{a}\langle y\rangle} \xrightarrow{\nu z.\overline{a}\langle z\rangle}\\
& \xrightarrow{a(sign((sign((hB, hU, E, w_{j,0}, hV_j, \ldots, H^{k+l}(w_{i,k+l}), hV_i, \ldots, w_{1,0}, hV_1), ssk_B)),}\\
& \xrightarrow{D, s, aenc(KV_i, pk(esk_{V_i}))), ssk_U))} \xrightarrow{a(OrderInf_1, MAC((OrderInf_1, H^{k+l-1}(w_{i,k+l})), H(KV_i)))}\\
& \xrightarrow{\nu v.\overline{a}\langle v\rangle} \xrightarrow{a(senc(page_1, KV_i))} \xrightarrow{\nu w.\overline{a}\langle w\rangle} \xrightarrow{a(H^{k+l-1}(w_{i,k+l}))} \ldots\\
& \xrightarrow{\nu f.\overline{a}\langle f\rangle} \xrightarrow{a(OrderInf_k, MAC((OrderInf_k, H^l(w_{i,k+l})), H(KV_i)))} \xrightarrow{\nu g.\overline{a}\langle g\rangle} \xrightarrow{a(senc(page_k, KV_i))}\\
& \xrightarrow{\nu h.\overline{a}\langle h\rangle} \xrightarrow{a(H^l(w_{i,k+l}))} \ldots \xrightarrow{\nu o.\overline{a}\langle o\rangle} \xrightarrow{a(OrderInf_{k+l}, MAC((OrderInf_{k+l}, H^l(w_{i,k+l})), H(KV_i)))}
\end{aligned}
$$

$\xrightarrow{\nu p.\overline{a}\langle p\rangle} \xrightarrow{a(senc(page_{k+l},KV_i))} \xrightarrow{\nu q.\overline{a}\langle q\rangle} \xrightarrow{a(H^l(w_{i,k+l}))} \xrightarrow{\nu r.\overline{a}\langle r\rangle} P'$

$P' \equiv \nu ssk_U.\nu ssk_B.\nu esk_{V_i}.\nu KV_i(\{(hB, pk(ssk_B))/b\_pk\} \mid$
$\{(hU, pk(ssk_U))/u\_pk\} \mid \{(hV_i, pk(esk_{V_i}))/v\_pk\} \mid \{(s, hV_i)/x\} \mid$
$\{sign((sign((hB, hU, E, w_{j,0}, hV_j, \ldots, H^{k+l}(w_{i,k+l}), hV_i, \ldots, w_{1,0}, hV_1), ssk_B)),$
$D, s, aenc(KV_i, pk(esk_{V_i}))), ssk_U)/y\} \mid$
$\{(OrderInf_1, MAC((OrderInf_1, H^{k+l-1}(w_{i,k+l})), H(KV_i)))/z\} \mid \{senc(page_1, KV_i)/v\}$
$\mid \{H^{k+l-1}(w_{i,k+l})/w\} \mid \ldots \mid \{(OrderInf_k, MAC((OrderInf_k, H^l(w_{i,k+l})), H(KV_i)))/f\} \mid$
$\{senc(page_k, KV_i)/g\} \mid \{H^l(w_{i,k+l})/h\} \mid \ldots \mid$
$\{(OrderInf_{k+l}, MAC((OrderInf_{k+l}, H^l(w_{i,k+l})), H(KV_i)))/o\} \mid \{senc(page_{k+l}, KV_i)/p\} \mid$
$\{H^l(w_{i,k+l})/q\} \mid \{(hU, KV_i, H^l(w_{i,k+l}))/r\})$

If there are two vendors, let our evaluation context be:
$C[\_] = \nu ssk_U.\nu ssk_B.\nu esk_{V_i}.\nu esk_{V_{i-1}}(\_|(!V_{i,i-1}shopping\_ds)|(!processV_{i-1})|(!processV_i))$
and our process is:
$P \equiv C[\overline{a}\langle hB, pk(ssk_B)\rangle.\overline{a}\langle hU, pk(ssk_U)\rangle.\overline{a}\langle hV_i, pk(esk_{V_i})\rangle.\overline{a}\langle hV_{i-1}, pk(esk_{V_{i-1}})\rangle]$.
We show that $\varphi(P') \vdash (hU, KV_{i-1}, H^l(w_{i,k+l}))$.

$P \xrightarrow{\nu b\_pk.\overline{a}\langle b\_pk\rangle} \xrightarrow{\nu u\_pk.\overline{a}\langle u\_pk\rangle} \xrightarrow{\nu vi\_pk.\overline{a}\langle vi\_pk\rangle} \xrightarrow{\nu vi{-}1\_pk.\overline{a}\langle vi-1\_pk\rangle} \xrightarrow{\nu x.\overline{a}\langle x\rangle} \xrightarrow{a(s,hV_i)} \xrightarrow{\nu y.\overline{a}\langle y\rangle}$

$\xrightarrow{a(sign((sign((hB,hU,E,w_{j,0},hV_j,\ldots,H^{k+l}(w_{i,k+l}),hV_i,H^{k_{i-1}}(Enc(H^{k+l}(w_{i,k+l}),v)),}$
$\xrightarrow{hV_{i-1},\ldots,w_{1,0},hV_1),ssk_B)),D,s,aenc(KV_i,pk(esk_{V_i}))),ssk_U)} \cdots \xrightarrow{\nu z.\overline{a}\langle z\rangle}$
$\xrightarrow{a(OrderInf_k,MAC((OrderInf_k,H^l(w_{i,k+l})),H(KV_i)))} \xrightarrow{\nu v.\overline{a}\langle v\rangle} \xrightarrow{a(senc(page_k,KV_i))} \xrightarrow{\nu w.\overline{a}\langle w\rangle}$
$\xrightarrow{a(H^l(w_{i,k+l}))} \xrightarrow{\nu p.\overline{a}\langle p\rangle} \xrightarrow{a(r,hV_{i-1})} \xrightarrow{\nu q.\overline{a}\langle q\rangle}$
$\xrightarrow{a(sign((sign((hB,hU,E,w_{j,0},hV_j,\ldots,H^{k+l}(w_{i,k+l}),hV_i,H^{k_{i-1}}(Enc(H^{k+l}(w_{i,k+l}),v)),hV_{i-1},\ldots,w_{1,0},hV_1),ssk_B)),}$

$\xrightarrow{D,r,aenc(KV_{i-1},pk(esk_{V_{i-1}}))),ssk_U))} \xrightarrow{\nu t.\overline{a}\langle t\rangle} \xrightarrow{a(OrderInf_1,MAC((OrderInf_1,H^l(w_{i,k+l})),H(KV_{i-1})))}$

$\xrightarrow{\nu u.\overline{a}\langle u\rangle} \xrightarrow{a(senc(page_1,KV_{i-1}))} \xrightarrow{\nu g.\overline{a}\langle g\rangle} \xrightarrow{a(H^l(w_{i,k+l}))} \xrightarrow{\nu h.\overline{a}\langle h\rangle} P'$

$P' \equiv \nu ssk_U.\nu ssk_B.\nu esk_{V_i}.\nu esk_{V_{i-1}}.\nu KV_i.\nu KV_j(\{(hB, pk(ssk_B))/b\_pk\} \mid$
$\{(hU, pk(ssk_U))/u\_pk\} \mid \{(hV_i, pk(esk_{V_i}))/vi\_pk\} \mid \{(hV_{i-1}, pk(esk_{V_{i-1}}))/vi-1\_pk\} \mid$
$\{(s, hV_i)/x\} \mid \{sign((sign((hB, hU, E, w_{j,0}, hV_j, \ldots, H^{k+l}(w_{i,k+l}), hV_i,$
$H^{k_{i-1}}(Enc(H^{k+l}(w_{i,k+l}), v)), hV_{i-1}, \ldots, w_{1,0}, hV_1), ssk_B)), D, s, aenc(KV_i, pk(esk_{V_i}))), ssk_U)/y\}$
$\mid \ldots \mid \{(OrderInf_k, MAC((OrderInf_k, H^l(w_{i,k+l})), H(KV_i)))/z\} \mid \{(senc(page_k, KV_i))/v\}$
$\mid \{H^l(w_{i,k+l})/w\} \mid \{(r, hV_{i-1})/p\} \mid \{(sign((sign((hB, hU, E, w_{j,0}, hV_j, \ldots,$
$H^{k+l}(w_{i,k+l}), hV_i, H^{k_{i-1}}(Enc(H^{k+l}(w_{i,k+l}), v)), hV_{i-1}, \ldots, w_{1,0}, hV_1), ssk_B)),$
$D, r, aenc(KV_{i-1}, pk(esk_{V_{i-1}}))), ssk_U)/q\} \mid \{(OrderInf_1, MAC((OrderInf_1, H^l(w_{i,k+l})),$
$H(KV_{i-1})))/t\} \mid \{senc(page_1, KV_{i-1})/u\} \mid \{H^l(w_{i,k+l})/g\} \mid \{(hU, KV_{i-1}, H^l(w_{i,k+l}))/h\})$

6 Conclusion

We extended PayWord to support payments at multiple vendors. The proposed scheme is a credit-based, off-line solution, with minimal computational and

storage costs. We also provided a detailed formal proof for payment approval, payment authorization, secrecy of payment information and unreusability in the applied pi calculus.

Acknowledgment

The publication was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund. The author is also supported by the Hungarian National Foundation for Scientific Research Grant No. K75566 and NK 104208.

References

1. M. Abadi, C. Fournet, *Mobile Values, New Names, and Secure Communication*, 28th ACM Symposium on Principles of Programming Languages, (2001), 104–115.
2. L. Aszalós, A. Huszti, *Applying Spi-calculus for Payword*, Proceedings of ICAI'10 8th International Conference on Applied Informatics, (2010), 295–302.
3. L. Aszalós, A. Huszti, *Payment Approval for PayWord*, D. H. Lee, M. Yung (Eds.): Information Security Applications - 13th International Workshop (WISA) 2012, Lecture Notes in Computer Science 7690, (2012), 161–176, Springer-Verlag.
4. B. Blanchet, B. Smyth, *ProVerif 1.85:Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, http://www.proverif.ens.fr/manual.pdf, (2011).
5. M. Hosseinkhani, E. Tarameshloo, M. Shajari, *AMVPayword: Secure and Efficient Anonymous Payword-Based Micropayment Scheme* International Conference on Computational Intelligence and Security (CIS), (2010), 551–555.
6. M. Payeras-Capella, J. L. Ferrer-Gomila, L. Huguet-Rotger, *An efficient anonymous scheme for secure micropayments*, Web Engineering, Lecture Notes in Computer Science 2722, (2003), 80–83, Springer-Verlag.
7. R. Rivest, A. Shamir, *PayWord and MicroMint: Two simple micropayment schemes*, Security Protocols, (1997), 69–87.
8. M. D. Ryan, B. Smyth, *Applied pi calculus*, Formal Models and Techniques for Analyzing Security Protocols, (2011), chapter 6.
9. C.T. Wang, C.C. Chang, C.H. Lin, *A new micro-payment system using general payword chain*, Electronic Commerce Research, 2, (2002), 159–168.
10. H. Wang, J. Ma, J. Sun, *Micro-payment protocol based on multiple hash chains* Second International Symposium on Electronic Commerce and Security, 1, (2009), 71–74.
11. Weidong Kou, *Payment Technologies for E-Commerce*, Springer, (1998).