PayWord Micro-Payment Scheme. Strengths, Weaknesses and Proposed Improvements.

By Ilan Azbel

Department of Computer Science, University of Cape Town, South Africa
Telephone : +27-21-6503799
Email : iazbel@cs.uct.ac.za

Contents

Introduction

Many promising applications on the Internet depend on the ability to pay small amounts of money for information and/or services. Currently, such services and applications are funded by advertising or subscriptions. The prevailing wisdom in the Internet community is that net surfers will not be willing to pay for content of web pages. The concept behind micropayments is that if the charge was low enough people wouldn't mind paying for content.

There are a number of proposed protocols for electronic commerce such as DigiCash [6], Open Market [7], Cyber Cash [8], First Virtual [9], and NetBill [10]. All these protocols are directed at medium to large purchases, from $10 and up. The costs for such transactions are normally a few cents and some percentage. So if a small transaction, say 10 cents or less was to be processed, the costs of the transactions would become a large part of the purchase price.

As a rough guide, hash functions are 100 times faster than RSA signature verification and about 10,000 times faster than RSA signature generation. Typical workstations are capable of 2 public key signature creations, 200 public key signature verifications and 20,000 hash functions per second [3]. So to support micropayments we can see that minimizing the amount of public-key verifications and creations should be a priority.

A few proposals have been put forward for these type of 'small' transactions, called micro-payment schemes. Each of these schemes has limitations, and until these limitations are ironed out there will not be sufficient confidence to support micro-transactions.

In this report I will be discussing one micro-payment scheme, namely PayWord [3] and then move onto a discussion about a scheme that I have based on PayWord, called iPay. In iPay, I have tried to improve on some of the weaknesses of the PayWord scheme by adding more verification stages at the beginning of any set of transactions.

As in any security protocols, both iPay and PayWord schemes have their own strengths and weaknesses. The PayWord scheme is heavily dependent on its level of trust. That is, the Broker and the vendor are both trusted parties while the user is the untrusted individual. This is the weakness that I have concentrated on fixing in the iPay scheme. Due to the added verification steps at the beginning of any transactions, the computational load on both parties is increased from one public key signature, to two. This added computational cost is not a very large downfall on the user's side, but could be computationally costly on the vendor side, especially during peek trading hours. For this reason, hardware based encryption might be preferred over software implementation, although hardware encryption is beyond the scope of this paper.

For each of the schemes I will give a general description, then give a more detailed explanation of the protocol and finally run down on some of the strengths and weaknesses of the protocol that I was able to pick up on. I also include a short concluding paragraph summary of the computational and storage burdens on each of the parties involved in the transactions and discuss a risk model involved in the micro-payment schemes.

Parties involved

PayWord and iPay involve three parties; a user (U) who makes the purchase and makes the payment, a vendor (V) who sells the goods/information and collects the payment and a broker (B) who keeps the accounts for U and V. For both iPay and PayWord, only a single broker model is considered, this is that both user and vendor must share a common broker for settlement of accounts.

Notations

Public keys of B, U and V are denoted PK_B, PK_U and PK_V respectively. Their secret keys are denoted SK_B, SK_U and SK_V. A message M with digital signature produced by SK is denoted $\{M\}SK$. This signature can be verified using the corresponding PK.

Let h be a cryptographically strong hash function such as MD5 [5] . The important property of h is the fact that it is a one-way function. That is, a very large search needs to be performed to find the input producing a given output.

PayWord

Introduction

The PayWord micro-payment scheme has been used in the Micro-payments Transfer Protocol (MPTP) [4]. In MPTP it is extended and defined into greater detail than in the original proposed paper [3]. I will be giving a brief description of the protocol without delving into too much detail. For a more in depth discussion, please refer to [4] in the references.

General Description

PayWord is a credit based MicroPayment scheme. This means that the buyer of the goods pays a fee at the end of each month for that months spending.

First U establishes an account with a B who issues the user with a PayWord certificate. This certificate gets renewed after a certain period of time (maybe monthly). This certificate authorizes the user to make PayWord chains which are redeemable by B to V.

For a first request, U computes and signs a commitment to a specific chain of paywords $w_1 \ldots w_n$. The user creates the PayWord chain in reverse order by picking a PayWord w_n at random and then computes

$$w_i = h(w_{i+1}) \text{ for } i = n-1, n-2, \ldots 0.$$

Here w_0 is the root and not a PayWord itself. U sends this commitment to V who verifies their signatures. The $i-th$ payment from U to V consists of the pair (w_i, i), which V can verify from the commitment and w_0.

At the end of each day (or any other period), V reports the last payment (w_l, l) and commitment to B for each user. B then charges U's account l cents and pays the vendor l cents.

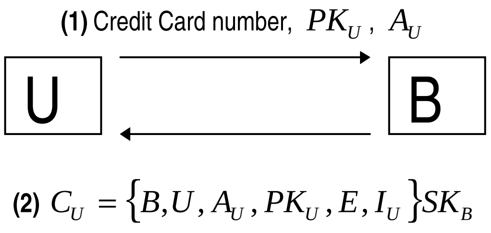
Protocol Details

User-Broker certificates

The User (U) must first start a relationship with the Broker (B) by requesting an account and a PayWord certificate. Through a secure authenticated channel, U gives B his/her details, including a credit card number, Public Key PK_U and delivery address A_U. B then sends U a certificate C_U of the form

$$C_U = \{B, U, A_U, PK_U, E, I_U\}SK_B$$

where E is the expiry date of this certificate and I_U is any additional broker information.


(1) Credit Card number, PK_U, A_U
U
B
(2) $C_U = \{B, U, A_U, PK_U, E, I_U\}SK_B$

This PayWord certificate is a statement by B, to any vendor, that B will redeem authentic paywords produced by U before the expiry date of this certificate.

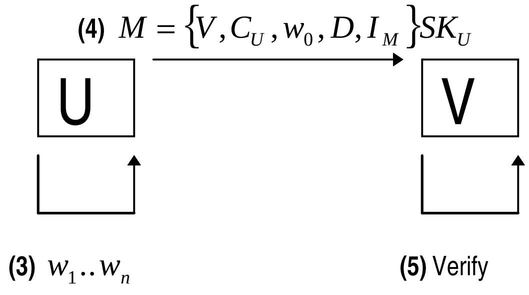
User-Vendor Relationships

User-Vendor relationships are transient, this means that a user can visit a web site purchase a few things and move on elsewhere. When U is about to contact V, U creates a PayWord chain $w_1 \ldots w_n$ with root w_0. This PayWord chain may be of any length. U then computes his/her commitment for that chain which comes in the form of

$$M = \{V, C_U, w_0, D, I_M\}SK_U$$

where V identifies the vendor, C_U is the PayWord certificate, w_0 is the root of the PayWord chain, D is the date and I_M is any additional information which may be required. M is signed by U and sent to V. This commitment authorizes B to pay V for any of the paywords $w_1 \ldots w_n$ that V redeems with B before date D.

V then verifies U's signature on M and the B's signature on C_U.



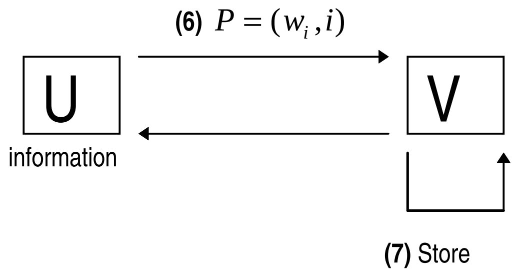
V should cache M to prevent a replay attack. U could try send earlier commitments and/or payments if this is not done. After this commitment has been verified and a price of a page has been agreed to, a payment needs to be made.

A payment P from U to V consists of

$$P = (w_i, i).$$

This payment is not signed by U because V can easily authenticate it through the commitment by applying i hash functions on w_0. The user spends the paywords in order, from 1 to n. So for each payment the user want to make he/she discloses the appropriate amount of paywords according to the price.

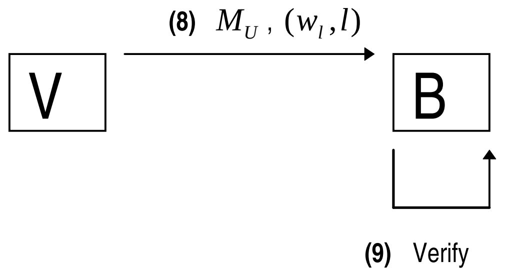
So for each commitment a vendor is paid l cents, where l corresponds to the highest index received in (w_l, l). With this methodology, V only need to store one payment (the one with the highest index) and commitment per user.



Vendor-Broker Relationship

V does not need to have a prior relationship with B, but does need to obtain PK_B in a secure way. Also, a means with which B pays V needs to be established. This is done outside of the PayWord scheme.

At the end a period of time, V sends B messages containing, for each of B's users the commitment M_U and the last payment (w_l, l). The Broker verifies each commitment (B can recognize his own signature) and each payment (by performing l hash functions on w_0).



Protocol Comments

Protocol Strengths

- No records are kept as to which documents/information has been purchased. [*Step **(6)***]
- PayWord can be used for larger MacroPayments because the only computational burden on the user is producing PayWord chains, but there is no limitation on the amount of chains one can produce. [*Step **(6)***]
- PayWord supports variable sized payments. This means that by sending the vendor $(w_1, 1)$ followed by $(w_5, 5)$ just means that U can pay the vendor 5 units in just one transmission. [*Step **(6)***]
- When B verifies the payments that V claims from him, B only needs to compute some hash functions which are cheap and only one signature verification (which are also relatively inexpensive). B can also charge for his computational time to cover his expenses. [*Step **(9)***]
- B never needs to respond to V in real-time. This means that B can batch his processing off-line. [*Step **(9)***]

Protocol Weaknesses

- Despite the fact that no names are sent over the network, the transmission of A_U destroys U's anonymity. [*Step **(1)***]
- If the public key of B is known, all of U's details can be accessed. [*Step **(2)***]
- U must sign a commitment for each vendor he/she pays. If the user switches between vendors, this can become computationally costly for the user. [*Step **(4)***]
- If U's public key is known then details of the commitment can be seen, including the root of the pay-word chain. This can lead to counterfeiting of pay-words. [*Step **(4)***]
- The transmission of payment and information do not seem to be coupled closely enough. The vendor may cheat the user by sending the wrong, or no information. If this happens often, the vendor may get a bad reputation and will lose business, so it is not in the vendor's interest to do such things. [*Step **(6)***]

Efficiency

- **Broker** signs each certificate, verifies each user's commitment (signature) and payment (one hash per payment). Broker stores each certificate and keeps accounts for users and vendors.
- **User's** verify his/her certificate, signs each commitment and performs one hash function per payment. Users stores his commitments and PayWord chains.
- **Vendor** verifies all certificates and commitments and performs one hash function per payment. Vendor stores commitments and last payment per user.

iPay

Introduction

The PayWord scheme is heavily dependent on its trust hierarchy. That is, we trust the broker the most, then the vendor and the most untrusted party is the user or customer. I disagree with this assumption. I feel that it is far more financially beneficial for an external party to masquerade as the vendor than it would be to masquerade as the user. I take this view because if an external party masquerades as the user, he/she can get access to information which would cost him only few cents. This could be done on a larger scale but does not come close to he amount of financial gain that could be realized by masquerading as the vendor and receiving thousands of payments from legitimate users.

I would like to give an analogy. Think of a chewing gum vending machine, where children put in 10 cents and receive a chewing gum. If a child puts in a fake coin, he would get the chewing gum he wants and the owner of the machine would lose out. It can be assumed that not a large percentage of children would do such a thing. Now think of a fake chewing gum vending machine. Lots of children would put in money hoping to get a chewing gum and would be disappointed because the machine does not really have any gum. The child would most probably think the machine is broken and move on, but if many children do this, the financial gain to the fake vendor is immense. Chances are that if a child saw another gum machine with the same brand name as the one that stole his money, he would not approach it; even if this was a real chewing gum vending machine.

With this idea in mind we can easily see that not only do we need to keep the vendors sure that the users are honest, we need to keep the users sure that the vendors are honest. If the users are unsure of vendors on a large scale then this could be far more detrimental to the future of micro-payments than if a few dishonest users make a few purchases. Therefore in iPay, we do not mind if a few illegal transactions take place (transactions are cheap) as long as a high percentage is legitimate and gets paid for by the user.

iPay is a scheme that uses the same ideas as PayWord for the payment for goods, but involves more authentication before any transactions takes place.

General Description

iPay is a micropayment scheme that also involves three parties, namely a user (U) that purchases information, a vendor (V) that sells the information and a broker (B) that guarantees the vendor that he will pay for the users' purchases. These scheme is directed more at a longer term relationship between the user and the vendor and tends to get computationally costly if the user changes between vendors often.

With this model the vendor does not process each transaction that the user makes with him, but rather accumulates all the transactions and processes them in a group. In this way transaction costs are reduced for the vendor and the user. This type of scheme conforms to the idea of reducing computational costs for micro-payments. The main difference in the payment scheme is that the user does not make a commitment to a set of paywords, but rather has a card (analogous to a bank card) that verifies that a broker has committed to pay for all of the user's transactions.

This tighter security comes in the form of some extra computation before any payment takes place. This extra computation involves two public key signatures and two signature verifications.
After the protocol has verified that the user and the vendor are both valid and are really who they say they are, we continue in a similar fashion to the PayWord scheme. That is, the user creates tokens (or paywords, as in the PayWord scheme) and uses them to pay for goods.

The fact that the user and vendor need to make two public key computations each may seem much but the PayWord scheme involves one public key computation each, so the computation is not increased too much as a whole if one looks at the total computation involved, including all the hash functions that need to be processed to make the actual transactions.

For this protocol to work it is assumed that the public key signatures of U and V are known to both parties. This could lead to a problem but can be solved by the implementation of a public-key directory that was maintained by the brokers and updated as soon as a new user opens an account.

Protocol Details.

User-Broker Cards

The protocol is initiated by U requesting an account with B. The user sends his/her details to B via a secure channel. (This process is outside of this protocol but a simple public/private key transaction would suffice). In return, B supplies U with a 'account card'. This card simply contains any details about U that B would like to keep track of. B signs this card with his public key. Therefore we note that only B can decrypt this message with his private key. Let us call this encrypted card C.

$$C = \left\{E, I_U\right\} PK_B$$

E would be the expiry date of the card and I_U would be any information that B needs to keep about U. The fact that this card is signed with B's public key offers a problem; namely the fact that the vendor is not able to check the validity of the card once transactions take place. This is a problem that is overcome by making sure that the user is really who he says he is and therefore has a valid credit reference. So if there are any problems with the validity of card, the user can be traced.

$$\begin{tikzcd}
\boxed{U} \arrow[r, shift left, "\textbf{(1)}\text{ Details via a secure channel}"] & \boxed{B} \arrow[l, shift left, "\textbf{(2)}\ C=\left\{E,I_U\right\}PK_B"]
\end{tikzcd}$$

After U has set up an account with B, he/she can purchase information at any vendor by using this card and a confirmation procedure.

User-Vendor Relationship

The user proceeds by requesting information from V. He first sends C to V. V responds by sending back a NONCE (N_1). U then signs C, N_1 and l (a limit on the total number of transactions he is willing to make in this communication) with his secret key and sends this to V. V decrypts this message by applying U's public key to the message and compares C to the C received earlier and also the NONCE from this message to N_1 .

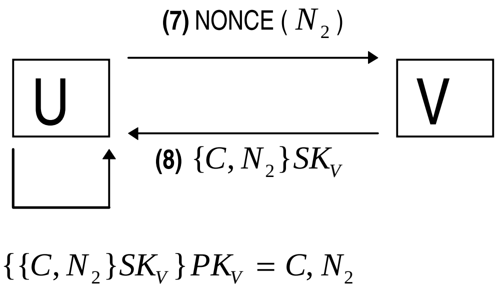
The reason why U includes an l in this communication is to prevent a party listening to the line producing a large amount of tokens and continuously purchasing goods with these tokens. It is included to limit the financial damage to either U or V (depending on the agreed contract with B).

$$\begin{tikzcd}
\boxed{U} \arrow[r, shift left, "\textbf{(3)}\ C=\left\{E,I_U\right\}PK_B"] & \boxed{V} \arrow[l, shift left, "\textbf{(4)}\text{ NONCE (}N_1\text{)}"]
\end{tikzcd}$$

$$\begin{tikzcd}
\boxed{U} \arrow[r, "\textbf{(5)}\ \{C,N_1,l\}SK_U"] & \boxed{V} \arrow[loop below, "\textbf{(6)}\ \{\{C,N_1,l\}SK_U\}\,PK_U = C,N_1,l"]
\end{tikzcd}$$

If all these match, then we know that U is who he says he is. This is true because if U wasn't who he says he is, he would not have been able to encrypt C and the NONCE with the correct secret key; therefore it would have decrypted incorrectly and V would not get correct results when comparing all the information. Also note that l is part of the message encrypted with U's secret key, this means that a party listening in on the conversation would be able to *read* l , but not change it because they would not be able to encrypt it again with U's secret key. This means that any unauthorized party would not be able to increase that amount of money that one could spend on this set of transactions.

Now we know that U is who he says he is, but we need to follow the same process to confirm that V is a valid vendor too. We begin this by U sending V a NONCE (N_2). V then sends back C and a NONCE signed by V's secret key. U then goes through the same process as V, namely, applying V's public key to this message and comparing C to the original C, and the nonce to N_2 . If all these comparisons match then V is really V.



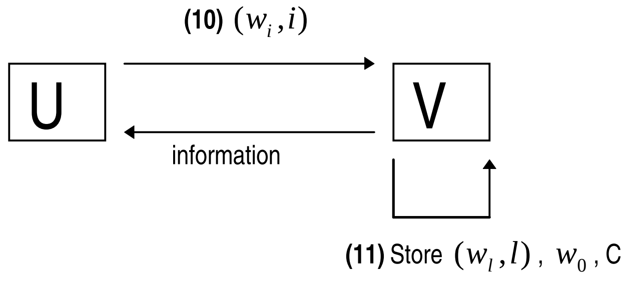
We can see immediately that there is a computational overhead cost involved here, but note that we have confirmed both parties and both parties have remained anonymous.

When all these messages have been completed we may start with the transactions. The user calculates tokens, which are equivalent to the payword chains described in the PayWord scheme. These tokens are created with a cryptographically secure algorithm such as MD5 or SHA.

The tokens are referred to as w_i, they are produced by $w_{i+1} = h(w_i, i)$ with $w_0 = N_2$ (N_2 is the NONCE that the user produced)

Because the user uses his nonce to create the tokens, he may do this off-line before even connecting with V. This is done by choosing a NONCE before any transactions take place. This type of off-line processing can reduce the amount of computational strain on U's computer. Furthermore, U may create only l (chosen earlier) tokens. This is to limit the number of valid transactions that can be made from w_0 .

The user then sends (w_i, i) to the vendor for each purchase that he makes. U does not need to encrypt this message because it is not feasible to try get w_{i-1} from w_i because MD5 and SHA are secure one way hash functions.

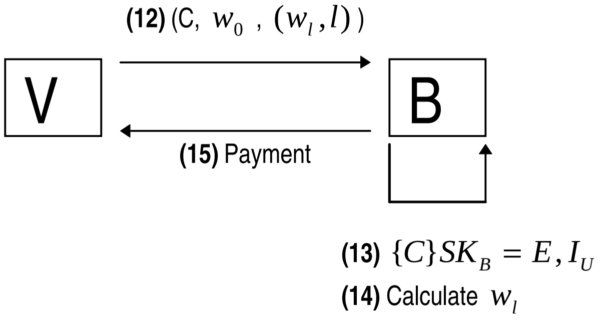


V only needs to apply one hash function for every token received, to verify the validity of that token. At the end of all the transactions, V needs to store the last token received (w_l, l) , w_0 and C (card). This is enough information to check how much money U has spent.

A question that arises here is how the vendor knows that the user is not overspending. He does not. The broker would need to pay the vendor all the money that the user has spent, even if the user is not going to pay the broker at a later stage. This is a problem in the protocol but can happen only on a small scale before the user gets 'black-listed' from this broker. This type of 'black listing' would call for an inter-broker listing of dishonest users that would not get valid cards again. Therefore it is not in the user's interest not to pay his/her account at the end of the agreed period.

Vendor-Broker Relationship

After a certain period of time, say one month, the vendor would like to settle with the broker and receive payment for all of the broker's users. To do this, the vendor sends (C, w_0 , (w_l, l)) to the broker. The broker then decrypts C to see if it is a valid card, then performs l hashes on w_0 to get w_l and thereby verifying that all these payments are valid. B would then pay via a financial
institution, and B would bill U with that months spending.



(12) (C, w_0 , (w_l, l))
V
B
(15) Payment
(13) $\{C\}SK_B = E, I_U$
(14) Calculate w_l

Protocol Strengths

iPay maintains all of the advantages of the PayWord scheme and also has additional advantages

- No records are kept as to which documents/information has been purchased. [*Step **(10)***]
- iPay can be used for larger macro-payments because the only computational burden on the user is producing tokens, but there is no limitation on the amount of chains one can produce. [*Step **(10)***]
- iPay supports variable sized payments. This means that by sending the vendor $(w_1, 1)$ followed by $(w_5, 5)$ just means that U can pay the vendor 5 units in just one transmission. [*Step **(10)***]
- When B verifies the payments that V claims from him, B only needs to compute some hash functions which are cheap and only one signature verification (which are also inexpensive). B can also charge for his computational time to cover his expenses. [*Step **(14)***]
- B never needs to respond to V in real-time. This means that B can batch his processing off-line. [*Step **(14)***]

The above were the same advantages that PayWord offered, but in addition to these, iPay offers more advantages such as :

- The User and the Vendor are both treated as untrusted parties. This lessons the ability of a third party to masquerade as either the user or the vendor. [*Step **(3 -> 9)***]
- User anonymity is maintained. The user details are only contained in C, which can only be decrypted by B.
- By choosing an l, the user limits himself to losses because even if somebody reproduces some tokens from w_0, only l of these tokens may be used.

Protocol Weaknesses

- Both U and V need to verify 2 public key signatures each. This is a computationally costly procedure. Therefore iPay is aimed more towards a large number of small purchases.
  [*Step **(3 -> 9)***]
- The transmission of payment and information do not seem to be coupled closely enough. The vendor may cheat the user by sending the wrong, or no information. If this happens often, the vendor may get a bad reputation and will lose business, so it is not in the vendor's interest to do such things. [*Step **(10)***]

Efficiency

- **Broker** signs each card, verifies each user's card (signature) and payment (one hash per payment). Broker keeps accounts for users and vendors.
- **User's** applies two signatures during the verification stages and performs one hash function per payment.
- **Vendor** applies two signatures during verification stages and performs one hash function per payment received. Vendor stores the card and last payment per user.

Risk Model

I will be discussing a risk model presented in the MicroPayment Transfer Protocol (MPTP) [4].

The main risk faced by the users of micro-payment schemes is the liability for unauthorized payments and either not receiving goods or receiving the wrong goods. The vendor risks not being paid although both iPay and PayWord can be changed to a debit style protocol where a user prepays the broker when opening an account. The broker risks the cost of user services due to malfunction and liability for payments in cases where the user and vendor are in dispute.

Credit Liability

A credit liability may be incurred by the broker if he chooses to guarantee payment to the vendor for all his users. This type of liability can be avoided by changing the micro-payment scheme from a credit based system to a debit based system. In a debit based system the user prepays a certain amount of money before being given the facilities (in terms of keys, cards or certificates) to make any purchases.

Credit abuse

An account is used to make payments with no intention to pay. Credit abuse may be discovered through a broker tracing payment patterns to detect sudden increases in purchase activity. Once again a debit system (as described in

Credit Liability above) can be deployed. Also a list of bad users can be maintained to prevent a user with bad credit from opening another account with a broker.

Counterfeiting

In both the iPay and PayWord scheme, an unauthorized party may produce counterfeit payment tokens. In iPay this is not of much use because the number of payments that will be accepted by the vendor is described by an upper limit which the user chooses. So any tokens made above this chosen limit cannot be used.

Unauthorized Withdrawal

This is not possible without the detection of the user because the user may request an audit of all transactions within a period of time. This does require a large amount of transaction logging on the broker's part.

Purchase Order Modification

In a purchase order modification attack, an external party may change an order to cause different goods to be delivered. This issue is not addressed in iPay or PayWord. Without authentication of the purchase order there is no way of avoiding this type of attack. If authentication does become a necessity, then a session key could be shared during the establishment of the protocol. Such session keys could have a lifetime of a few transactions.

Double Spending

This is not possible with iPay or PayWord. Each set of payment tokens is uniquely identified by a unique root. In iPay if the same NONCE in used twice, the vendor should refuse to sell the goods to the user.

Failure to Credit Payments

If a broker deliberately deducts a certain amount of money from a user's account, and does not make a corresponding entry to a vendor, the vendor may choose not to do business with that broker again. In this way users will not open accounts with this broker because not all vendors would like to deal with him; therefore limiting the amount of information available to the user.

Denial of Service

The Internet does not have the capabilities of protecting against such an attack. In my opinion, it is not a significant attack in terms of purchase of goods paradigm (unlike for example, military information exchange) because the denial of information does not produce any financial gain to the party denying the service. It can be done a few times for 'fun' but does not hold any long term financial benefits for any party involved.

Repudiation

Once a user sends his/her card and the authentication process has been completed, the user cannot deny having made purchases.

Failure to deliver

Failure to deliver may occur for many reasons including vendor fraud. The Internet is not a reliable transport medium, both the user and the vendor may be in good faith to pay and deliver the goods respectively, but the delivery fails nevertheless. A solution to this type of problem would be to enable some satisfaction guarantee policy in which the user could refuse payment for goods.

Conclusion

I have discussed an existing micro-payment scheme, PayWord and have shown its strengths and weaknesses. The main weakness (in my opinion) of PayWord is the trust model (as discussed in the introduction to iPay). I have tried to adjust the initial verification steps of this protocol to produce iPay.

The additional verification steps have successfully fixed the trust model but have increased the computational costs on both the vendor and user sides. I feel that this added verification is essential for micro-payments to work on a large scale. Trust within a transaction environment needs to work both ways; user trusts vendor and vendor trusts user. I believe that only a protocol that implements this two way trust will be the one that finally gets implemented and widely accepted.

References

[1] *Security, Payment, and Privacy for Network Commerce*. B. Clifford Neuman. IEEE Journal, Vol. 13 No. 8 October 1995.

[2] *iKP - A Family of Secure Electronic Payment Protocols.* Mihir Mellare, Juan A. Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, Michael Waider. July 12, 1995

[3] *PayWord and MicroMint - Two Simple MicroPayment Schemes*. R.L.Rivest, A.Shamir

[4] *Micro Payment Transfer Protocol (MPTP) Version 0.1. Working Draft 22-Nov-95*. Phillip M. Hallam-Baker.

[5] *RFC 1321, The {MD5} Message-Digest Algorithm*. R.L. Rivest, Internet Request for Comments, April 1992

[6] DigiCash Inc., http://www.digicash.com/

[7] Open Market Inc., http://www.openmarket.com/

[8] CyberCash Inc., http://www.cybercsh.com/

[9] First Virtual Holdings Inc., http://www.fv.com/

[10] NetBill, Carnegie Mellon University, http://www.ini.cmu.edu/netbill/

[11] Millicent. Mark S. Manasse, 1995, http://www.research.digital.com/SRC/personal/Mark_Manasse/uncommon/ucom.html