Removing the financial incentive to cheat in micropayment schemes

L. Buttyán

Micropayment schemes usually do not provide fairness, which means that either the payer or the payee, or both, can cheat the other and gain a financial advantage by misbehaving in the protocols. The authors propose an extension to a family of micropayment schemes that removes the financial incentive to cheat. The proposed extension does not provide true fairness, but it renders misbehaving practically futile for both the payer and the payee. This is achieved without any substantial loss in efficiency in most practical cases.

Introduction: Micropayment schemes [1 – 5] are electronic payment schemes explicitly developed for very low value payment transactions, such as payment for information on the World Wide Web and payment for each second of a phone call. The chief design goal of micropayment schemes is efficiency. Reaching this goal requires that communication and processing costs of micropayments be kept as low as possible, otherwise these costs may exceed the value of the payment itself and, thus, applying the micropayment scheme would not be economical. Other properties, such as fairness and sometimes even security (at least to some extent) are sacrificed in favour of efficiency. Here, fairness means that either both the payer and the payee receive the expected item in the transaction (i.e. the service paid for and the payment, respectively) or neither receives anything. Providing fairness is considered to be too expensive for micropayments, because it requires either too much communication and/or computation, or the assistance of a trusted third party. Consequently, micropayment schemes are not fair; if the payer has to move first, then the payee can cheat by not providing the service after the payment has been received, otherwise the payer can cheat by not sending payment for the received service. It is argued that this potential misbehaviour of the parties is tolerable, since the potential loss is very low. While this is true considering one single transaction, it might be a problem considering the global system and longer time periods. To illustrate this, we consider a service provider, which persistently cheats by stealing one cent in each transaction. This service provider can earn more than one million dollars in a year given it has about 300000 transactions per day. As a rough guide, a rather small telecommunication network operator with 50000 subscribers processes at least 300000 phone calls per day. The question is: can micropayment schemes be improved with respect to fairness without too much loss in efficiency? In this Letter, we answer this question affirmatively by proposing an extension to a family of micropayment schemes that, although not providing true fairness, at least removes the financial incentive to cheat.

Original micropayment scheme: We only consider micropayment schemes where payment is based on the successive release of elements in a chain of cryptographic hash values (e.g. [1, 3 – 6]). In particular, we will illustrate our ideas by extending the PayWord system [5]. Other members of the same family can be extended in a similar way.

There are three roles in PayWord: the user U, the vendor V and the broker B. Each user is registered with at least one broker. This relationship is represented by a PayWord certificate signed and issued by the broker, which binds the broker's name, the user's name and the user's public key together.

When U wants to buy some services from V, they generate a fresh chain of paywords $w_1, w_2, ..., w_n$ by picking the last payword w_n at random and then computing $w_i = h(w_{i+1})$ for $i = n–1, n–z ..., 0$, where h denotes a publicly known, cryptographically strong one-way hash function and n is chosen by U. w_0 is called the root of the payword chain, and it is not a payword itself. U then signs a commitment to this payword chain, which contains the vendor's name and the root of the payword chain. This commitment is sent to V at the beginning of the service session. It authorises B to pay V for any of the paywords $w_1, w_2, ..., w_n$ that V redeems with B later.

The ith micropayment from U to V consists of the pair (w_i, i). This can be verified by V using w_{i-1} which is known from the previous micropayment or from the commitment in case of $i = 1$. A typical service session consists of a sequence of micropayments:

$U \rightarrow V$: w_1, 1
$U \leftarrow V$: first part of service
$U \rightarrow V$: w_2, 2
$U \leftarrow V$: second part of service
...
$U \rightarrow V$: w_l, l
$U \leftarrow V$: last part of service

where $A \rightarrow B$: *msg* means that A sends the message *msg* to B, and $l \leq n$.

After service provision, V contacts B and presents the commitment and the last payment (w_l, l) received. B verifies the signature on the commitment and the validity of w_l, and, if these verifications are successful, pays V the amount corresponding to l paywords and charges that amount to the billing account of U.

As discussed before, PayWord does not provide fairness. The vendor may cheat the user by sending an unexpected service or nothing at all. If such misbehaviour is detected by the user, then they can stop sending more paywords, but they still lose the last one already sent. Since a payword has a very low value, this does not cause too much damage for the user. A persistently cheating vendor, however, can earn a substantial amount of money in this way.

Modified micropayment scheme: We now present our extension to PayWord that removes the financial incentive to cheat and, thus, makes the misbehaviour described above practically futile. We modify the original scheme only slightly and show that efficiency does not decrease substantially in most practical cases. This modification was inspired by [7], which describes how electronic coins can be ripped and ripped coins can be used in payments to remove the financial incentive to cheat. Our basic idea is to double the size of the hash chain and let a payword consist of two consecutive hash values. Intuitively, these can be thought of as two half-paywords. The first half-payword is sent to the vendor before the service provision and the second half is sent after the service has been provided. Thus, the vendor can redeem the full payword only if they have provided the service. This gives an advantage to the user, who can refuse to send the second half-payword in the hope that they can escape from paying for the received service. To deter the user from doing this, we let the broker charge the full value of a payword to the user's account if the vendor presents the first half-payword (which leaves the broker with a surplus of the value of one payword). This makes cheating of no interest to the user, because they have to pay, even though the vendor cannot get this money. The surplus of the broker is handled according to some policy (e.g. it can be distributed to charity). This policy is verified and its observance controlled by independent law enforcement organisations, thus rendering collusion between the user and the broker as well as between the vendor and the broker very difficult.

We modify only the micropayment protocol and the way in which the paywords are redeemed by the vendor and the user is charged by the broker. When U wants to buy some services from V, they generate a fresh chain of hash values $w'_0, w_1, w'_1, w_2, w'_2 ..., w_n w'_n$ by picking w'_n at random and then computing $w_i = h(w'_i)$ and $w'_{i-1} = h(w_i)$ for $i = n, n–1, ..., 1$. The root of the chain is now w'_0 and U puts this value in the commitment, which they construct in the same way as in the original scheme and send to V at the beginning of the service session.

The ith micropayment has three steps. First U sends the pair $(w_i, 2i–1)$ to V (the first half-payword), then V provides the ith piece of the service to U, and finally U sends the pair $(w'_i, 2i)$ to V (the second half-payword). Each half-payword can be checked by V using the previously received half-payword. It might seem that our scheme requires twice as many messages from U to V as the original one, but fortunately this is not true in most practical cases. Typically, a service session consists of a series of consecutive micropayments, and U can send the second half-payword of the ith payment $(w'_i, 2i)$ and the first half-payword of the $(i+1)$st payment $(w_{i+1}, 2i+1)$ in one single message. Furthermore, since V can always compute w'_i from w_{i+1}, only the second pair $(w_{i+1}, 2i+1)$ has to be sent. A typical service session may thus have the following appearance:

$U \rightarrow V$: w_1, 1
$U \leftarrow V$: first part of service
$U \rightarrow V$: w_2, 3
$U \leftarrow V$: second part of service
...
$U \rightarrow V$: w_l, $2l–1$
$U \leftarrow V$: last part of service
$U \rightarrow V$: w'_l, $2l$

which involves only one additional message compared to the original scheme.

If everything goes well, then V can present the commitment and the pair $(w'_l, 2l)$ to B, which performs the same verifications (with twice as

many hash computations) as in the original scheme. If the verifications are successful, then B pays V the amount corresponding to l paywords and charges the same amount to the account of U. If something goes wrong and the last half-payword is missing, then V can only present the commitment and the pair (w_l, $2l$–1) to B. After the verifications, B pays V the amount corresponding to l–1 paywords and charges U for the amount corresponding to l paywords.

Analysis: Our scheme does not provide fairness, since theoretically it is still possible that one of the parties could cheat the other. The vendor can refuse to provide the last part of the service after receiving the first half of the last payword. In this case, the user receives services that are worth l–1 paywords, but they will be charged for l paywords. Similarly, the user can refuse to send the second half of the last payword after receiving the last part of the service. In this case, the vendor provides services that are worth l paywords but can redeem only l–1 paywords.

However, none of the parties gain any financial advantages by cheating. In the original scheme, it is possible that the vendor provides services that are worth l–1 paywords and redeems l paywords. In our scheme, the vendor can never earn more than the value of the services they provide, since the user authorises the payment after they have received the services (by sending the second half-payword). Similarly, the user can never receive services that are worth more than that they are charged for, because they are charged before receiving the service (when releasing the first half-payword).

In terms of efficiency, our scheme is not substantially less efficient than the original one, in most practical cases. First of all, the number of public key cryptographic operations (digital signature generation and verification) is the same as in the original scheme. Although each sequence of consecutive micropayments in our scheme requires one additional message from the user to the vendor, this is negligible considering that such a sequence usually consists of hundreds of messages. We require the hash chain to be twice as long as in the original scheme. This requires twice as many hash computations by the user (when the paywords are generated), the vendor (when the paywords are verified) and the broker (when the paywords are redeemed). The user and the broker cases are not real efficiency problems, because these computations are off-line. The size of the memory where the user stores the chain does not need to be doubled. It is more efficient to store only the first half-paywords (i.e. w_1, w_2, ..., w_n) and w'_n additionally, because the second half-paywords are usually not used in a sequence of consecutive micropayments. If a second half-payword is needed, then it can be easily computed by one application of the hash function on one of the stored values. The required memory sizes for the vendor and the broker are the same as in the original scheme. Thus, the only factor that makes our scheme less efficient than the original one is that verification of the micropayments requires twice as many on-line hash computations by the vendor (i.e. two hash computations per micropayment). This is the price that has to be paid for the additional guarantees that our scheme provides.

Acknowledgments: The author wishes to thank U. Wilhelm, M. Hamdi, J.-P. Hubaux, S. Staamann and B. Frajka for useful comments and helpful discussions.

20th October 1999
Electronics Letters Online No: 20000163
DOI: 10.1049/el:20000163

L. Buttyán (*Institute for Computer Communications and Applications, Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland*)

E-mail: Levente.Buttyan@epfl.ch

References

1 ANDERSON, R., MANIFAVAS, C., and SUTHERLAND, C.: 'NetCard - A practical electronic cash system'. Technical report, Cambridge University, Computer Laboratory, 1995

2 GLASSMAN, S., MANASSE, M., ABADI, M., GAUTHIER, P., and SOBALVARRO, P.: 'The Millicent protocol for inexpensive electronic commerce', *in* O'REILY and Associates, Inc. (Eds.): Proc. 4th Int. World Wide Web Conf., December 1995, pp. 603–618

3 HAUSER, R., STEINER, M., and WAIDNER, M.: 'Micro-payments based on iKP'. Technical report RZ 2791, IBM Research, February 1996

4 PEDERSEN, T.: 'Electronic payment of small amounts'. Technical report DAIMI-PB-495, Aarhus University, Computer Science Department, 1995

5 RIVEST, R., and SHAMIR, A.: 'PayWord and MicroMint: Two simple micropayment schemes'. Technical report, MIT Laboratory for Computer Science, 1996

6 MARTIN, K., PRENEEL, B., MITCHELL, C., HITZ, H., HORN, G., POLIAKOVA, A., and HOWARD, P.: 'Secure billing for mobile information services in UMTS'. Proc. IS&N'98, 1998,

7 JAKOBSSON, M.: 'Ripping coins for a fair exchange'. Proc. EUROCRYPT'95, 1995, pp. 220–230