SVP: a Flexible Micropayment Scheme

Jacques Stern, Serge Vaudenay

Ecole Normale Supérieure — CNRS
{Jacques.Stern,Serge.Vaudenay}@ens.fr

Abstract

We propose a cheap micropayment scheme based on reasonable requirements. It is flexible in the sense that many security options are possible depending on the policy of the involved participants. We avoid large data storage, heavy computations. The scheme is software based for the user and hardware based for the service provider. Possibilities of having software-based solution for both are also presented.

1 Introduction

In the forthcoming years or even months, it is anticipated that electronic payments over secure networks are going to expand rapidly. The definition of the SET protocol (see [4]) by a group of credit cards providers is a definite sign of this expected growth. Among the variety of payment schemes that have been proposed recently, several address the very specific question of micropayments (see [1, 2, 6]). Such payments arise in the context of the Internet when an individual user is browsing around and wish to access resources for which a small payment appears adequate. Of course, this can be done through some subscription scheme but it is very likely that such solutions will not meet the needs of the generic Internet user.

As was pointed out by Rivest and Shamir in [2], micropayments require exceptional efficiency in order to be economically viable. As a result, the direct use of public key cryptography appears forbidden. Even conventional cryptosystems such as DES are questionable. The best choice seems to point to hash functions, possibly keyed so as to be used as Message Authentication Codes. We assume that the reader is familiar with these cryptographic tools and refer to Schneier's book [7] for more information.

At this point, it is in order to describe the three parties involved in a micropayment scheme. They are:

1. the user who would like to access the resource against a micropayment.
2. the service provider or vendor who offers service and needs to be paid for.
3. the broker who offers support for the transaction.

In the micropayment context, we also assume that the communications with the broker are rare (say daily or weekly-based for the providers and monthly-based for the users). Thus, "expensive" cryptography can be allowed at this level. On the other hand transactions between users and providers are frequent and, for these communications, we will only use a message authentication code Mac which is supposed to be "cheap"; actually, we will also use an unkeyed one-way hash function which we denote by Hash.

When adopting a cryptographic setting where only "cheap" algorithmic tools are allowed (in terms of computing power, communication load and ... licenses), it is necessary to closely examine the resulting level of security. There are two major concerns

1. On the service provider's side, the risk of overspending: a customer might use the rights granted by the broker in order to buy more than what was originally agreed.

2. On the customer's side, the risk of having his rights stolen by a "sniffing" attacker and/or of being improperly billed by the broker.

Otherproposals also address these problems: in Millicent (see [1]), they are solved by a systematic use of secret key cryptography, both on the user's side and on the provider's side. Thus, a key-management rather heavily enters the picture. In Payword([2]), Rivest and Shamir, following a design independently imagined by many researchers, use public key in order to sign a chain of iterated hash. Besides using the apparatus of public key, this has the drawback that the customer has to use a different chain for each provider. A similar design with trees in place of chains has been considered by Jutla and Yung ([6]), again using algorithmic tools similar to those studied by other researchers (see e.g. Vaudenay [5]).

Our setting is quite similar to the one considered by Rivest and Shamir in Micromint in that overspending is basically "punished" by a form of blacklisting, which is acceptable for micropayments. However, we try to offer a stronger guarantee on the user's side by introducing a challenge-based payment protocol, which, albeit very simple, protects from fraudulent copies of the user's tokens. We do not make use of the clever idea of multiple collisions but we only need MACs and hash functions, thus introducing a very "cheap" system which we call SVP for Small Value Payments.

2 Description of SVP

In the setting of SVP, we assume that the providers have been given a tamper proof device for validating micropayments. Each broker may decide to distribute his specific device or to share it with others (e.g. use a device provided by a bank

$$\begin{array}{ccc} \text{Provider } P & & \text{User } U \\ & \xleftarrow{\qquad U,t,r \qquad} & r \text{ random} \\ q \text{ random} & \xrightarrow{\qquad q \qquad} & \\ \text{check} & \xleftarrow{\qquad v \qquad} & v = \mathrm{Hash}(U, P, c, q, r, s) \end{array}$$

Figure 1: Payment protocol

consortium). This turns out to be a very reasonable assumption since most of the stores in the real life already have such a device for bank card payment. We assume that the device is tamper proof and trusted by the broker.

In the protocol below, we assume that communications both between the broker and the provider's device and between the broker and the user are confidential. This can be achieved with strong cryptographic schemes such as digital signatures and encryption algorithms. We postpone the discussion on this issue to a later section.

We first describe a simple protocol with a memoryless tamper resistant device. The protocol is flexible in the sense that several options are possible depending on the chosen policy. It is illustrated on Figure 1.

Provider initialization The broker B fixes its own secret key k_B and communicates it in a secure way to the device of each provider.

Withdrawal protocol In order to allow a user U to pay, B generates a random token t together with a spending key $s = \mathrm{Mac}_{k_B}(U, t)$. By doing this, the broker authorizes the user to spend a given amount of money with the key s. As was already mentioned, the relationship between the broker and the user is trust-based, so the control of the amount spent is left to the user. Furthermore, must be such that systematic fraud can be detected and the dishonest user black-listed.

Payment protocol When willing to spend a microamount c to a provider P, the user introduces himself as being U and having a token t to the provider. The user also chooses a random number r and communicates it to the provider. Then, the provider sends a random challenge q. The user then reveals $v = \mathrm{Hash}(U, P, c, q, r, s)$ and the provider queries his device with (U, c, t, q, r, v) which checks whether $v = \mathrm{Hash}(U, P, c, q, r, \mathrm{Mac}_{k_B}(U, t))$ or not. At this point two options are possible for the provider (see Figure 2). The provider can just keep an account balance for each user and just increase U's account by c (Option 1). He can store (t, q, r, v) as well if he is suspicious about this payment, which shall depend on his policy (Option 2).

Option 1
Payment: P increases U's account A_U by c.
Clearing: P sends U's account A_U to B. B records P in the list of U's providers. B pays if U's balance is ok. Otherwise, B registers a problem between U and one of its providers.

Option 2
Payment: P adds (U, c, t, q, r, v) in U's records.
Clearing: P sends U's records to B. B adds all (P, q, r) in U's records. B checks and pays for all payments unless (P, q, r) has already been used.

Figure 2: Two options for the provider

Payment clearing The provider regularly sends the broker the amount of money spent by user U and he controls if the accounts are consistent. If not, the broker requests for a valid proof (U, c, t, q, r, v) of payment (Option 2). If it cannot be provided, the broker just refuses the payment and records that there must be a problem with U or its providers. If such a proof is released, the broker pays and checks if (q, r) has already been spent to P. If it has, the broker suspects the provider to be dishonest. If not, the broker stores (P, q, r) in the (U, t)-records.

3 Tamper resistant device with memory

For more security, it is preferable to use tamper-resistant devices with permanent memory. Even if the capacity is low, we can improve the simple protocol significantly.

Small capacity device The device can detect multiple successive failures as an attempt to use it as an oracle. Another feature is that the device can store the total amount to be paid to the provider like a tamper resistant cashier machine. This avoids that dishonest providers request over-payment from the broker before the users' accounts are spent. This shall provide a notable level of trust in the clearing process: errors would be caused either by unpurposed mistakes or by dishonest users. Options 1 and 2 are thus replaced by Options 3 and 4 this way (see Figure 3).

High capacity device Here we discuss the extra level of security that a more expensive tamper-resistant device provide, if the policy of the broker or the provider allows it. First, we modify the protocol in a way that the q value is challenged by the device. When high storage capacity devices are affordable, we can assume the device can store (U, c, t) (or increase a (U, t)-indexed register by c). (Since the device is trusted by the broker, we don't need to keep q and r since

Option 3
Payment: P increases U's account A_U by c. P's device increase a global balance by c.
Clearing: P's device sends the global balance to B. P sends U's account A_c to B. B checks if this is consistent with the global balance. B records P in the list of U's providers. B pays if U's balance is ok. Otherwise, B register a problem between U and one of its providers.

Option 4
Payment: P adds (U, c, t, q, r, v) in U's records. P's device increases a global balance by c.
Clearing: P's device sends the global balance to B. P sends U's records to B. B checks if this is consistent with the global balance. B adds all (P, q, r) in U's records. B checks and pays for all payments unless (P, q, r) has already been used.

Figure 3: Small capacity device options

v has been checked by the device. We don't need to keep v either since the device generated q, so v is fresh for sure). Then, during the clearing process, the broker only checks that user U did not overspend his token t. This Option is illustrated on Figure 4.

4 Possible attacks

Assuming the k_B transmission between the broker and the providers is secure, the only way to recover k_B (which enables to create money) is to deduce it from the other informations. For instance, (dishonest) users may try to get k_B from the equation $s = \mathrm{Mac}_{k_B}(U, t)$. The Mac function must therefore resist to this kind of attack.

A cheater may try to spend money on another user's account (U, t). If U kept his key s secret, the cheater only can tap the payment communication. Then, either he/she cracks s from the equation $v = \mathrm{Hash}(U, P, c, q, r, s)$ (which must therefore be protected by the Hash function) or he/she tries to answer to spend without the knowledge of s. But then, he/she has to answer to a fresh challenge q. Since the real user picked a random number r before getting q, the cheater must commit himself to an r number, but his/her knowledge may be limited to a very few (q, v) pairs. The probability the challenge is one of these is therefore small, and payment disruption will alert the provider as a tentative attack.

A dishonest user may try to overspend the key, but this will be detected by the clearing protocol.

A dishonest provider may try to be paid for fake transactions. Using the tamper resistant device as an oracle to forge valid payment proof will thus be

Option 5
Payment: P's device generates q. P's device increases U's account A_U by c.
Clearing: P's device sends U's balance to B. B pays.

Figure 4: High capacity device option

detected by the device. Other forgeries are either improbable, or protected by the strength of the Mac and Hash functions.

A cheater may also try to actively attack the protocol by rerouting the communication between the user U and the provider P towards another provider P' so that he can benefit of the services from P' by making U pay for it. This is avoided by the use of the identifier P of the real provider in the v answer. Similarly, the cheater can still benefit of P's services with the payment of U, but this is solved if the provider check the identity of U.

5 Secret key strengthening

Tamper resistant devices are never totally tamper proof, as publicly discussed in relation with the recent work on differential fault analysis. Whatever the real threat is, it might be better not to let a single key k_B in mass manufactured devices. We rather propose to use several keys $k_B^1, \ldots, k_B^n$. Each provider has an identifier encoded as a set of $n/2$ indices I, which means that a provider identified by I knows the keys k_B^i for $i \in I$.

In the payment protocol, the provider sends I together with q. Then the user pays with

$$v = \mathrm{Hash}(U, P, c, q, r, \mathrm{Mac}_{k_B^i}(U, t) : i \in I)$$

The use of sets with size $n/2$ comes from Sperner's Theorem [3] which basically says this is an optimal choice (see [5]). We can propose other combinatorial improvements, but this downgrades the simplicity of the protocol.

6 On the Mac and Hash choice

The Mac and Hash primitives shall be chosen so that they are fast and cheap to evaluate. The aim of Hash is to ensure that s is not disclosed by $\mathrm{Hash}(U, P, c, q, r, s)$ and that it is infeasible to predict its value within an acceptable probability. Therefore it can eventually hash onto a very small piece of information, say 16 bits or 32 bits.

The Mac primitive shall hash onto a larger length since it must resist to exhaustive search attacks. For instance, it can hash onto 64 bits. Similarly, the key k_B must be large enough (say 64 to 128 bits, but with a smaller validity period) to thwart brute force attack.

7 Secure communications

Communications with the broker must be secure in two ways. Firstly, the secrecy of the keys (k_B for the provider and s for the user) must be enforced. For this, we need either encryption, or physical off-line delivery. For instance, the key k_B can be implemented in hardware in the devices (eventually all the future ones with it), and the key s can be given once for all when the user registers with the broker.

Secondly, the authenticity of the clearing process must be achieved from the provider to the broker. This can be done with the Mac function by usual protocols.

Secrecy may be required in the clearing process too, in order to protect the users' privacy. This can be achieved by encryption, but legal issues may only allow this together with the 5th option (since the encryption capacity is only given to a trusted sealed device).

8 Privacy issue

Our protocol however suffers from the lack of anonymity: the provider records U's identity and communicates it to the broker. This would harm the user's privacy. But U can use a "privacy provider" who shall request and pay for P's services and provide the service to U and get paid for.

9 On the use of the tamper resistant device

We can propose guidelines to avoid the use of a tamper resistant device. The main problem is that dishonest providers can use their keys k_B to pay for any user. We can solve the problem by adding a proof of payment $x = \mathrm{Mac}_{k_U}(P, c, q, r)$ to be verified only by the broker who shares a secret key k_U with the user. This proof x shall be hashed in the v value to so that no-one can separate them. The broker also needs to check if x has not been used several times. But then, if this proof is not valid, we cannot say whether the provider or the user is dishonest. We can decide that the broker will not pay under such circumstances, but the (honest) provider will be stopping providing services to the (dishonest) user. The problem is solved if we assume that the provider agrees to lose the corresponding amount of money.

10 Conclusion

We have defined a flexible payment system that can be implemented through very basic cryptographic primitives in a setting where the service providers use a tamper-proof device trusted by the broker. SVP thus offers a very simple solution to the problem of micropayment, while achieving a significant level of security to the individual customer.

References

[1] S. Glassman, M. Manasse, M. Abadi, P. Gauthier, P. Sobalvarro. The Millicent protocol for inexpensive electronic commerce. In *World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, pp. 603–618, O'Reilly, 1995.

[2] R. L. Rivest, A. Shamir. PayWord and MicroMint: two simple micropayments schemes. *CryptoBytes*, vol. 2, num. 1, pp. 7–11, 1996.
`http://theory.lcs.mit.edu/ rivest`

[3] E. Sperner. Ein Satz über Utermengen einer endlichen Menge. *Mathematische Zeitschrift*, vol. 27, pp. 544–548, 1928.

[4] Secure Elecronic Transactions (SET) specifications. Draft. 17 july 1996
`http://www.mastercard.com/set`

[5] S. Vaudenay. One-time identification with low memory. In *Proc. EUROCODE'92*, Udine, Italy, CISM Courses and Lectures 339, pp. 217–228, Springer-Verlag, 1993.
`http://www.dmi.ens.fr/ vaudenay`

[6] C. S. Jutla, M. Yung. Paytree. "Amortized-signature" for flexible micropayments. Presented at the Rump Session of ASIACRYPT'96.
`http://www.kreonet.re.kr/AC96/AC96.html`

[7] B. Schneier. *Applied Cryptography*, 2nd Edition, John Wiley and sons, 1996.