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 can be used for any payment which is online between the customer and the vendor and offline with the broker. 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 customer and hardware based for the vendor. 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 [1]) 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 [6, 13, 8]). Such payments arise in the context of the Internet when an individual user is browsing around and wishes 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 [13], 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 [14] for more information.

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

1. the customer 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 vendors and monthly-based for the customers). Thus, "expensive" cryptography can be allowed at this level. On the other hand transactions between customers and vendors are frequent and, for these communications, we will only use message authentication codes Mac (and Mac') which are 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 vendor'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.

Additionally, robustness criteria shall be considered in the case that a customer looses his secret key.

Other proposals also address these problems: in Millicent (see [6]), they are solved by a systematic use of secret key cryptography, both on the customer's side and on the vendor's side. Thus, a key-management rather heavily enters the picture. In Payword([13]), 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 vendor. A similar design with trees in place of chains has been considered by Jutla and Yung ([8]), again using algorithmic tools similar to those studied by other researchers (see e.g. Vaudenay [16]).

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 black-listing, which is acceptable for micropayments. However, we try to offer a stronger guarantee on the customer's side by introducing a challenge-based payment protocol, which, albeit very simple, protects from fraudulent copies of the customer'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 as for *Small Value Payments*.

2 Description of SVP

In the setting of SVP, we assume that the vendors have been given a tamper proof device for validating micropayments. For instance, this device can be a smart card or a PCMCIA card. Each broker may decide to distribute his specific device or to share it with others (*e.g.* use a device provided by a bank consortium). This turns out to be a very reasonable assumption since most stores in the real world already have such a device for bank card payment or cashier machines. 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 vendor's device and between the broker and the customer are secure (from a confidentiality and integrity viewpoint). 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 the most secure version of our protocol. We assume that the device has an internal (small) permanent memory and an external (larger) memory (which does not need to be physically secured). The internal memory has two registers Σ_B^c and Σ_B^d which are the global credit and debit sums of all transactions to be cleared by the broker B. In addition it contains a few information about previously aborted or failed transaction (to alert for fraudulent access attempts). This mechanism leads to usual security cares and will not be detailed here. The external memory has many m_T^c and m_T^d registers for all token T which have been used to pay to V (tokens will be defined below). Each register has the form

$$m_T^i = (\text{date}, Id_T, \Sigma_T^i, Mac_{k_B}(\text{date}, Id_T, \Sigma_T^i))$$

where "date" is the date when the register has last been updated and k_B is the key of the broker B for the token T. The payment protocol itself is illustrated on Figure 1.

Vendor initialization The broker B fixes its own secret key k_B and communicates it in a secure way to the device of each vendor. The device initializes its registers to zero and clears the external memory.

Withdrawal protocol In order to allow a customer C to pay, B generates a "token" T which is a bitstring with the form

$$Id_T = [\text{token number}||\text{expiration date}||Id_B]$$

together with a spending key $k_T = Mac_{k_B}(\texttt{token}, Id_T)$. Here "`token`" is a fixed bitstring which is used to avoid bad interactions between several Mac computations with different types. The string Id_B is an identifier of the broker. By doing this, the broker authorizes the customer to spend a given amount of money with the key k_T. As was already mentioned, the relationship between the broker and the customer is trust-based, so the control of the amount spent is left to the customer. Furthermore, it must be such that systematic fraud can be detected and the dishonest customer black-listed. So, the token T shall be (privately) associated to the customer C in the Broker's database.

Payment protocol When willing to spend an amount a to a vendor V, the customer introduces himself as having a token T to the vendor. The customer also chooses a random number r_C and communicates it to the vendor. Then, the vendor gives (Id_T, a, r_C) to his device which outputs a random challenge r_V to be sent to C. (The device shall records that a transaction happened to start with (Id_T, a, r_C, r_V) in order to later on detect if it aborted or not.) The vendor also send the identification string Id_D of his device. The customer then reveals a "purchase"

$$p = Mac'_{k_T}(\texttt{purchase}, Id_T, Id_D, a, r_C, r_V)$$

$$\begin{tikzcd}
\text{Vendor } V & \text{Customer } C \\
{} & r_C \text{ random} \arrow[l, "Id_T{,}r_C"'] \\
r_V \text{ random} \arrow[r, "Id_D{,}r_V"] & {} \\
\text{Device } D \text{ checks} & \text{compute } p \arrow[l, "p"']
\end{tikzcd}$$

$$p = Mac'_{k_T}(\texttt{purchase}, Id_T, Id_D, a, r_C, r_V)$$

Figure 1: Payment protocol for amount a

where "`purchase`" is a fixed pattern string and the vendor queries his device with (Id_T, p) which checks whether

$$p = Mac'_{Mac_{k_B}(\texttt{token}, Id_T)}(\texttt{purchase}, Id_T, Id_D, a, r_C, r_V)$$

or not (the values a, r_C and r_V are read from memory). If the payment is successful, the register Σ_B^c is increased by a and the record m_T^c is requested to the memory. If it does not exist, the device shall create a new one initialized to zero. The register Σ_T^c is then increased by a by the device and saved in the external memory.

Cancellation Similarly, a purchase from token T can be cancelled by the vendor by using a cancellation key. In this case the device increase Σ_B^d and Σ_T^d by a in a similar way.

Payment clearing The vendor regularly sends the broker the amount spent by his customers. He just ask the device to send Σ_B^c and Σ_B^d, and the vendor sends all registers m_T^c and m_T^d. The broker checks the consistency of the counters (*i.e.* that the Mac values are correct, that the register dates are between the last clearing date and the current date, and that the sum of all Σ_T^c is Σ_B^c and the sum of all Σ_T^d is Σ_B^d). Then the broker pays for $\Sigma_B^c - \Sigma_B^d$ and increase each of his counter Σ_T by $\Sigma_T^c - \Sigma_T^d$. The counter Σ_T is the money spent by the token T. If the token T has overspent, the corresponding customer shall be contacted for explanations.

3 Additional security protocol

A problem with the above payment scheme is that a customer can overspend his credit, that his secret key k_T can be lost (purposedly or not) or stolen. Revocation of tokens shall thus be performed by an additional security protocol based on black-lists. The verification that a token Id is not in the last updated black-list shall be checked upon the responsibility of the vendor (*i.e.* clearing for revoked token will be refused by the broker). On the other hand, payment performed by a token before its revocation date shall be cleared (with the timestamp of the device) as soon as the vendor realizes the token has been revoked.

Frauds are still possible during the latency period of the revocation process. Another approach (which can be added to reduce the fraud rate) was suggested by Jarecki and Odlyzko [7] and Yacobi [17]. The main idea is that the vendor shall probabilistically send a message to the broker (which thus needs to be online) to check for the credit. The broker may use the received message to estimate the real credit spent by the customer and send an alert before discovering effective overspending.

We also recall that the relationship between the vendor and the broker, and the customer and the broker is contract-based. So we don't need a huge cryptographic system to make frauds impossible. We can only try to make it hard and detectable.

4 Tamper resistant device without memory

The payment scheme is flexible depending on the vendor/broker policy and how much they want to pay (in particular for the tamper resistant module). Much simpler tamper resistant device without permanent memory are possible.

In this case we propose two options 1 and 2 (see Figure 2). In the first option (which is less expensive for the vendor), the risk is taken by the vendor: if a customer is cheating, the vendor will not get paid by the broker. In the second option, the risk is taken by the broker.

In both options, the random challenge r_V is generated by the vendor (and not the device as in the previous protocol).

The first option consists in keeping an account balance Σ_T for each token T (as in the previous protocol, but without the secure control of the device).

Option 1
Payment: V increases T's account Σ_T by a.
Clearing: V sends all T's accounts Σ_T to B. B records V in the list of C's vendors. B pays if balance T is ok. Otherwise, B registers a problem between C and one of its vendors.

Option 2
Payment: V adds (a, r_C, r_V, p) in T's records.
Clearing: V sends all T's records to B. B adds all (Id_D, r_C, r_V) in T's records. B checks and pays for all payments unless (Id_D, r_C, r_V) has already been used.

Figure 2: Two options for the vendor

On payment with amount a, the vendor just increase Σ_T by a. In the clearing process, the vendor communicates all Σ_T to B. The broker then pays only if the customer did not overspent with the token. If he did, then either he cheated or one of his previous vendors cheated. Such a problem shall be recorded.

In the second option, the vendor keeps (Id_T, a, r_C, r_V, p) in memory. In the clearing, the broker keeps all (Id_T, Id_D, r_C, r_V) in memory. He pays for any valid $(Id_T, Id_D, a, r_C, r_V, p)$ unless (Id_T, Id_D, r_C, r_V) is already in memory.

5 Tamper resistant device with memory

For more security, it is preferable to use tamper-resistant devices with permanent memory. For instance, the protocol presented in Section 2 is much more secure than protocols presented in Figure 2. Additionaly, tamper resistant device shall use there memory in order to detect multiple successive failures as attempts to use it as an oracle.

If we can afford devices with huge memory, the proposed protocol can be substantially simplified: records which was originally stored in the external memory can rather be stored in the physically secured internal memory. Then the Mac in record m_T^i is not useful any more. We can also use a global balance register instead of a credit register and a debit register.

6 Possible attacks

Assuming the k_B transmission between the broker and the vendors 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) customers may try to get k_B from the equation $k_T = Mac_{k_B}(\texttt{token}, T)$. The Mac function must therefore resist to this kind of attack. We recommand a key-length in the range 80–128 bits for k_B.

A cheater may try to spend money on another customer's account T. If C kept his key k_T secret, the cheater only can tap the payment communication. Then, either he/she cracks k_T from the equation

$$p = Mac'_{k_T}(\texttt{purchase}, Id_T, Id_D, a, r_C, r_V)$$

or he/she tries to answer to spend without the knowledge of k_T. But then, he/she has to answer to a fresh challenge r_V. Since the real customer picked a random number r_C before getting r_V, the cheater must commit himself to an r'_C number, but his/her knowledge may be limited to a very few (r_V, p) pairs. The probability the challenge is one of these is therefore small, and payment disruption will alert the vendor as a tentative attack. We believe that a 16-bit output is enough for Mac'. Similarly, we recommand a keylength of 48–64 bits for k_T.

A dishonest customer may try to overspend the key, but this will be detected by the clearing protocol. Additionally, probabilistic polling may detect it earlier in order to reduce the fraud rate.

A dishonest vendor may try to be paid for fake transactions. Using the tamper resistant device as an oracle to forge valid payment proof will thus be detected by the device if it has permanent memory. If not, the device shall be sufficiently slowed down for such an attack to be infeasible. Other forgeries are either improbable, or protected by the strength of the Mac function.

A cheater may also try to actively attack the protocol by rerouting the communication between the customer C and the vendor V towards another vendor V' so that he can benefit of the services from V' by making C pay for it. This is avoided by the use of the identifier Id_D of the real vendor's device in the p answer. Similarly, the cheater can still benefit of V's services with the payment of C, but then the real customer can complain to V.

7 Secret key strengthening

Tamper resistant devices are never totally tamper proof, as publicly discussed in relation with the recent work on transient fault analysis[2, 4, 5, 10]. Whatever the real threat is, it might be better not to have a single key k_B in mass manufactured devices. Rather we propose to use several keys $k_B^1, \ldots, k_B^n$. Each vendor has an identifier which is secretly hashed onto a set of $n/2$ indices I_D, which means that a vendor identified by I_D knows the keys k_B^i for $i \in I_D$.

In the payment protocol, the customer pays with

$$p_i = Mac'_{Mac_{k_B^i}(Id_T)}(\texttt{purchase}, Id_T, Id_D, a, r_C, r_V) : i = 1, \ldots, n$$

The use of sets with size $n/2$ comes from Sperner's Theorem [15]: the number of subsets such that any two subsets are not comparable for the inclusion relation is maximal when we take all the subsets with size $n/2$. This has been used for optimizing signature schemes based on hash functions. Those algorithms have been studied and discovered by several authors including by one author of the present paper [9, 11, 12, 16, 3]. We can adapt the last updated combinatorial technics from this area, but this downgrades the simplicity of the protocol.

8 Secure communications

Communications with the broker must be secure in two ways. Firstly, the secrecy of the keys (k_B for the vendor and s for the customer) 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 customer registers with the broker.

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

Secrecy may be required in the clearing process too, in order to protect the customers' privacy. This can be achieved by encryption.

9 Privacy issue

Our protocol however suffers from the lack of anonymity: the vendor may records C's identity and communicates its purchases to the broker. This would harm the customer's privacy. But C can use a "privacy provider" who shall request and pay for V's services and provide the service to C and get paid for.

10 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 vendors can use their keys k_B to pay for any customer. We can solve the problem by adding a proof of payment

$$t = Mac'_{k_C}(\mathtt{tag}, Id_t, Id_D, a, r_C, r_V)$$

to be verified only by the broker who shares a secret key k_C with the customer. (This key may be obtained with a relation like $k_C = Mac_k(Id_C)$ where k is a master key used by the broker.) This proof t shall be hashed in the v value to so that no-one can separate them. The broker also needs to check if t has not been used several times. But then, if this proof is not valid, we cannot say whether the vendor or the customer is dishonest. We can decide that the broker will not pay under such circumstances, but the (honest) vendor will be stopping providing services to the (dishonest) customer. The problem is solved if we assume that the vendor agrees to lose the corresponding amount of money.

In this protocol the key k_B must however be strengthened as in Section 7. There is still a problem with colluding dishonest vendors which shall forge fake purchases for another vendor. This is solved by the fact that the set of k_B^i used by the device is secret: unless the set of colluding vendors recover all the keys, the probability that a honest vendor accepts a forged payment is lower than 1/2.

11 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 for the individual customer.

Acknowledgments

We wish to thank the European "Human Capital and Mobility" programme for funding attendance to the Financial Cryptography Workshop for one of the authors.

References

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

[2] R. J. Anderson, M. G. Kuhn. Tamper resistance — a cautionary note. In *Proceedings of the 2nd Usenix Workshop on Electronic Commerce*, Oakland, California, U.S.A., pp. 1–21, nov. 1996.

[3] D. Bleichenbacher, U. M. Maurer. Directed acyclic graphs, one-way functions and digital signatures. In *Advances in Cryptology CRYPTO'94*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 839, pp. 75–82, Springer-Verlag, 1994.

[4] D. Boneh, R. A. deMillo, R. J. Lipton. On the importance of checking computations. Preprint, nov. 1996. To appear in the proceedings of Eurocrypt'97.

[5] E. Biham, A. Shamir. Research anouncement: a new cryptanalytic attack on DES. Oct. 1996. Presented at the Rump Session of *Fast Sofwtare Encryption'97*.
`http://jya.com/dfa.htm`

[6] 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.

[7] S. Jarecki, A. Odlyzko. An efficient micropayment system based on probabilistic polling. In these proceedings.

[8] 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`

[9] L. Lamport. Constructing digital signatures from a one way function, Technical report CSL-98, SRI Intl., 1979.

[10] A. K. Lenstra. Memo on RSA signature generation in the presence of faults. Sep. 1996. Unpublished.

[11] R. Merkle. A Certified Digital Signature, In *Advances in Cryptology CRYPTO'89*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 435, pp. 218–238, Springer-Verlag, 1990.

[12] M. Naor, M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, Seattle, Washington, U.S.A., pp. 33–43, ACM Press, 1989.

[13] 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`

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

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

[16] 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`

[17] Y. Yacobi. On the continuum between on-line and off-line e-cash systems. In these proceedings.