Chapter 11

μP: A MICROPAYMENT SYSTEM

Pedro A. L. Mindlin, Christiane M. Schweitzer, Tereza Cristina M. B. Carvalho, Wilson V. Ruggiero

Abstract: The development of electronic commerce has led to a new trend: the distribution of digital information. Micropayment systems come as an alternative, allowing the implementation of such transactions at low costs. This work introduces **μP**, a micropayment system based on central generation of electronic coins that are bought by and distributed among customers and easily verified – and thus accepted as payment – by electronic commerce vendors. It differentiates itself by generating a single group of tokens that can be used for shopping in all of these vendors. This process is performed concerning security and scalability requirements.

Key words: Electronic commerce, micropayments, network security.

1. INTRODUCTION

The development of electronic commerce has led to the development of a new category of payment systems. Many of them already exist and are currently being used, such as credit card and electronic money systems. However, these systems usually can't be used for small amount payments and they present high latencies for each transaction.

Micropayment systems were designed to solve both questions to small amount transactions – concerning also security and scalability requirements that are inherent to any payment system to be implemented in open networks.

The possibility of performing small amount transactions opens interesting new paths to electronic commerce: it makes selling information products – with values in the range of cents, such as images or access to information pages or individual news texts – possible.

This paper presents a micropayment scheme adequate for smaller transactions. It is organized as follows. Section 1 is this introduction. Section 2 discusses electronic commerce, including components and functionality of these systems. Section 3 specifies the scheme we developed, explaining its architecture. Section 4 presents the results obtained from implementing such scheme. Finally, section 5 presents our conclusions.

2. ELECTRONIC COMMERCE

Electronic commerce transactions consist of information exchanges such as the product information that the vendor sends to the client and the financial information that the client sends to the vendor, usually through the financial system.

The micropayments is a new category of payment systems, initially proposed circa 1992, which took advantage of new uses for hash functions like SHA [NIST] and MD5 [Rivest-92]. Among the most important proposals of micropayment schemes are the ones by [Rivest-96], [Pedersen] and [Anderson, et. al.]. Micropayment systems were designed for electronic commerce systems and are therefore electronic in their nature. As so, they require no adaptation for the electronic medium.

Micropayment systems typically consist of three entities: the user, who wants to buy something, normally a person using his or her computer connected to the Internet; the vendor, which is the company or person who is selling a product or service through the network; and the agent, which is the company – usually into the financial business – that is responsible for the system, for issuing tokens and making their payment reliable, for maintaining user and vendor accounts.

Micropayments has resurfaced over the last few years, and we believe that it could be successfully deployed, because electronic commerce is now prepared for accepting significant changes.

2.1 Related Work

One of the first important proposals of micropayment was Millicent [Manasse et. al.]. This scheme made use of a script containing the owner identification, a serial number and other information signed by a digital certificate. The user would buy one script from the agent to each vendor he or she would buy products from. At each transaction, he would send the script to the vendor, who would deduct the purchase value from it and digitally sign the new value, maintaining an account history for the user inside the script.

Millicent opened way to lots of other works, like NetCard [Anderson et al], TickPayments [Pedersen], PayWord [Rivest-96] and iKP [Hauser et. Al.], all of them making use of the same general idea of signed certificates used to authenticate digital money.

Among the most important micropayment schemes proposed at that time was PayWord [Rivest-96], which was used as the basis for the system here presented. PayWord is a credit-based system in which the user gets from the agent a digital certificate containing his name, IP address, a public key and other information. The certificate is signed by the agent. The certificate allows the user to create token chains by computing hash functions over a seed and digitally signing the last value in the chain, which is then sent to a vendor. All the subsequent values sent to the vendor during that day (the previous tokens in the hash chain) are verified against the signed token received in the beginning of the day with a simple hash function computing. As hash functions are one-way, it is impossible for the vendor to find out what is the next token to be received, but it is fairly easy to verify it.

Some other micropayment schemes have been proposed since 1997 as improvements to some of the original proposals. Among them are UpayWord [Mu et. al.], and MR1, MR2, MR3 [Micali et. al.], that are improvements over the scheme presented by [Rivest-96].

3. ARCHITECTURE

The first question to be addressed was what underlying micropayment scheme we would use. Some schemes were discarded for imposing the use of dedicated hardware, which would imply high costs. Then, it was decided to use a software-based scheme. The other central question here was the distribution requirement: the system would have to work over the Internet and multiple platforms, especially Windows, Linux and MacOS. This led to the use of Java technology.

Next came the task of creating an independent set of tokens for each vendor. The first idea was to create what we called a **specific micropayment system**: the system would work for only one vendor; the generated token set would only be used to make purchases with that vendor. This scheme would be useful for big companies selling their goods, but would not be useful for small vendors. Thus we expanded the system, developing what we called a **generic micropayment system**, which would create a unique set of tokens to be used with all vendors.

The **generic micropayment system** works as follows. A unique set of tokens is generated when the user registers within the system. When the user visits the registered vendors, he or she sends the tokens to these vendors in

order to make purchases. The tokens are then verified locally by each vendor. As the tokens are the output of hash functions, they have to be verified by applying these functions on them and comparing the results with the next value on the chain. Note that the next value on the chain has to be already stored by the vendor, so that it can be compared with the generated one. This leads to the problem of sending the root token to the vendors.

The first solution we devised for this problem was to create a vendor certificate, which is sent to each vendor when the user certificate is created. That way, if the user certificate contains a chain of n tokens, the vendor certificate contains the $n+1$ token and information about the user to whom that token belongs. When making the first purchase, the user sends to the vendor the token n ; the vendor would take this token and feed it to the hash function, obtaining token $n+1$, which it compares with token $n+1$ that is in the certificate. If they are the same, the transaction is authorized and the vendor stores token n . At the next transaction with that vendor, the user will send token $n-1$, which hashes to n , and is stored at the vendor. The vendor authenticates the purchase, and so on.

This scheme seems to function properly, but what happens if token $n-1$ is spent with some other vendor, which did not receive token n ? To solve this problem the vendor will have to hash token $n-1$ twice, obtaining token $n+1$, which is in the certificate. Thus, all tokens will have to contain an order number, which represents the number of hash function calculations to be made over that token for it to be verified. For this scheme to work, though, the root token ($n+1$ in our example) has to be self-verifiable, which can be made through the use of a digital signature by the agent when it generates the vendor certificate. We chose to embed the signed vendor certificate into the user certificate and make the user send this vendor certificate along with each token. The vendor has to verify the agent signature on the certificate and calculate the hash value of the token as many times as its order number.

3.1 Frauds

There are mainly two types of frauds: the vendor and the user frauds.

The vendor fraud happens when the vendor intentionally creates and presents to the agent a valid token, which was not spent by any real user at the vendor's site. This fraud is possible because a user can send different tokens to different vendors. Let's see an example: the user sends token n to vendor A, then sends token $n-1$ to vendor B and then sends token $n-2$ to vendor A. If vendor A wants, it can create token $n-1$ from token $n-2$ and fool the agent, because obtaining token $n-1$ from token $n-2$ is just computing the hash function once over token $n-2$. In this situation, vendors A and B present the same token to the agent, who, in lack of more information, has no choice

but paying both of them. A scheme for protection against these frauds will be presented ahead.

The user fraud happens when the user breaks into the locally stored certificate and changes values, creating false tokens. To circumvent this possibility we opted for storing the user certificate in cryptographic form. Another fraud that can occur is based on the certificate generation algorithm: the user has the right to have his certificate regenerated in the event of accidentally losing it in his computer; at the time the new certificate is generated, there can be unreported expenses from the user at some vendors, which will lead the agent to generate more tokens than the user actually has.

To prevent these frauds it is necessary for the agent to verify the tokens spent at the vendors. This can be done in four different ways: online, offline, almost online and batch mode. The chosen scheme for **μP** was almost online, though we also implemented an optional batch mode. In almost online mode, at each transaction, the vendor sends the token received to the agent, but does not wait for authorization from it. This scheme allows the agent to keep immediate track of transactions but keeps latency at a minimum. It does not prevent frauds, but makes detection fast and easy to do. The smaller the time interval, the better the fraud protection and the higher the communication and processing cost..

3.2 CRL (Certificate Revocation List)

To keep track of fraudulent users the system will maintain a CRL (Certificate Revocation List). The vendors will access the CRL at the agent's site and maintain it locally updated to use it at every transaction.

3.3 Integration

The **μP** should be integrated with other systems, such as a web application that delivers the product to the user. The simplest and easiest solution to achieve this is to use a database.

3.4 Components

μP has five elements: the token - a 20 bytes number, coded in Base64, with an associated order number; the certificate - a chain of tokens, each one being the result of the hash function computing on the previous one; the user module; the vendor module and the agent module:

Certificate: The certificate is generated by the agent at user request and is kept both in the agent's database and in the user's local disk. The user's copy is stored in cryptographic form to prevent eavesdropping and frauds. At

any given moment there will be only one valid certificate for each user in the generic micropayment system and one certificate for each user-vendor pair in the specific micropayment system.

User: The user interacts with μP when he or she is registering, making a purchase and requesting a new certificate from the agent.

Vendor: The vendor interacts with the system when it registers, sells a product and sends information to the agent. The registering of the vendor is an important moment in which it has to decide if it wants to use the specific or the generic micropayment system.

Agent: The agent is the central element in the system. It generates the tokens, collects money from the users and pays the vendors for the tokens received. It has to maintain accounts for all the users, keeping track of each token they spend, and detecting any possible frauds - in this case it revokes fraudulent users or vendors. The agent has to be always informed of spent tokens so it can be able to generate new certificates in case of loss and so it can be able to detect and prevent frauds. Certificate generation can also occur when the user runs out of unused tokens and tries to make a purchase.

3.5 Processes

μP can be described in terms of its function processes, which are: registering, token generation, purchase transaction and transaction informing and verification by the agent:

Registering: μP was not developed to provide its users with anonymity. Each prospect user must provide the system consistent identification data so he or she can be accepted as a valid user. The user has to provide a username and a password, which will be used for all further operations such as transactions or token purchases to maintain his or her privacy and to positively identify him or her to the system. As any secure registering process, μP will be done over secure SSL connections. The vendors also have to register within the system, and at this time they have to choose whether they want to work with a specific or a generic micropayment system.

Token generation: Token generation occurs when the user is a new one or when he or she does not have enough tokens to make a purchase. It can also happen if the user has lost his or her certificate. In this case the agent invalidates the former certificate and generates a new one with the same amount of unspent tokens. The token generation process can be better understood through Figure 1 and Figure 2, steps A to D:

A – Token request. The user informs the agent his or her name, password and the number of tokens he or she wants to buy.

B – Payment sequence. This phase is comprised of external protocol payments such as SET (Secure Electronic Transaction) for the user to pay for the requested tokens. It will not be part of this study.

C – Token generation. The agent generates the tokens through the following steps: it verifies the password given by the user against its database; it gets from the database the order number k of the next token to be spent by the user; it generates a random number and from this number it generates the token sequence, with order numbers starting from k ; it digitally signs the last token (root) with the user’s public key in the sequence and generates vendor certificate, and inserts this vendor certificate into the new user certificate; it revokes the user’s last valid certificate; it inserts the new certificate into the database and it makes the new certificate available for the user to download.

Figure 1. Tokens generation

D – Token delivery. The certificate is delivered through an applet that retrieves it from the agent server and stores it in the user’s local disk.

Purchase transaction: This is the most important process in μP . It can be better understood in Figure 2, steps 1 to 8.

The transaction begins when the user clicks on a link at the vendor’s page. The vendor then returns an applet to the user. This applet is the most important component of the user module. The applet searches for a user certificate on the local disk, according to the naming rules, and prompts the user for a password. From the password it obtains the cryptographic key, which is used to decrypt the file. The applet then searches for unused tokens and checks if there is the necessary number of them to make the purchase. If the number of tokens is insufficient, the applet redirects the browser to the agent’s site, going to step A.

If the certificate is valid and the number of tokens is enough, the applet sends to the vendor the next token – according to the value of the purchase – together with its order number, the signed vendor certificate which contains the root token, the serial number of the certificate, the user name, the product id and the purchase value. The vendor checks if the certificate is valid (i.e. is not in the CRL) and if the received token is authentic by computing its hash

value k times and comparing it with the signed root token, showed in the Figure 1.

If the token is authentic, the vendor generates a random number, puts this number in the database and returns the same number to the user. Right after that, the vendor sends information about the purchase to the agent (user name, the token itself, its order, the purchase value and the authorization number issued). If the token is invalid, the vendor cancels the transaction and returns a value of -1 to the user.

When it receives an authorization number different from -1, the applet redirects the browser to the product page, sending the authorization number together with the request for the page. The web application responsible for the product will then check the authorization against the database and deliver the product to the user if the authorization is valid.

Figure 2. μ P processes

Transaction informing and verification: Information about every transaction made at the vendor site can be transmitted to the agent right after the transaction or in batches at defined time intervals. There is a standalone program designed for the vendor that can sweep the database at regular or variable intervals of time and send new transactions info to the agent. On the

agent side, a servlet is responsible for parsing the data and storing it in the database, checking immediately for frauds and errors.

4. RESULTS

μP will be evaluated in accordance with the framework proposed by [Schmidt et. al.]. This framework makes qualitative evaluation of micropayment schemes based on nine parameters, clustered in three dimensions: microeconomic, technologic and social. The results of the evaluation can be seen in Table 2. To evaluate micropayment systems with these parameters, the authors attribute to each one of them a grade that ranges from “--” to “-” to “o” to “+” to “++”.In the next itens, we will examine each of these evaluations in detail:

Low transaction cost: When evaluating the transaction we can ignore the phase of sending information to the agent and concentrate on the user-vendor interaction. The transaction cost is evaluated in terms of its computational cost, which is the cost of one digital signature verification and k computing of the hash function. As k is a variable number, the cost of each transaction is non-deterministic. Moreover, k ranges can vary from one user to another, depending on the size of the user’s certificates. The hash function is fast enough and can be considered low cost, even if computed many times. The digital signature verification, though, can be considered a strong delaying factor. Thus we grade the transaction cost of our system as “+”.

Table1. μP evaluation

Dimension	Parameter	μP
Microeconomic	Low transaction cost	+
	Atomic transaction	O
	User base	O
Technologic	Security	-
	Reliability	+
	Scalability	-
	Latency	+
Social	Peer-to-peer payments	Does not apply
	Anonymity	-

Atomic Transactions: **μP** was designed and implemented so as not to consider a token as spent if the authorization number was not received. If the applet receives a -1 it will not modify the contents of the certificate, keeping the token as new. However, the atomicity of a **μP** transaction does not consider the final delivery of the product. If the connection is lost at this last stage, for example, the system cannot recover it. A **μP** transaction ends when

the authorization number is received. We consider this a flaw in the system and thus grade atomicity as “o”.

User base: The μP user base still does not exist and cannot be evaluated. The grade for this parameter is “o”.

Security: μP was implemented to handle secure connections between modules that offer authenticity, integrity and confidentiality services for messages exchanged. This was achieved deploying SSL protocol suites for communication. The fraud detection scheme is still weak and requires human intervention to work properly. One example of a fraud that is difficult to catch is the generation of false tokens by vendors. Although it was designed to be weak against frauds and gain strength from the speed and usefulness, in this particular parameter we have to be strict and grade the system as “-”.

Reliability: Due to the fact that the transaction is verified only at the vendor, the system’s reliability is distributed among all vendors’ sites. There can be some problems concerning the Java applets at the users’ computers, because they need access to local disks, but we will consider the situation in which they are authorized to do that by the user and can work normally. There is a central point of failure at the agent when it generates the tokens, but it does not comprise the actual purchase transaction. For these reasons we graded μP as “+” for this parameter.

Scalability: Up to this moment it has not been possible to perform scalability tests on the implemented system. We will consider theoretical scalability, based on the fact that the agent centralizes all operations except the transaction. Although μP is decentralized in the transaction process, it is very centralized when it concerning its, administration. The agent’s database has to contain all the tokens generated for all the certificates of all users relating to all vendors (although we must say that we believe the majority of vendors will choose the generic micropayment system), and has also to keep records of each transaction performed by users and vendors. The agent also has to check all transactions for possible frauds. This centralized dependency is a clear scalability flaw in μP and for this reason we graded it “-”.

Latency: The transaction latency is the time needed for one digital signature verification and a variable amount of hash function computations, as described in section 4.1. Although it is the most important factor for the latency, the digital signature verification can be considered as a relatively fast operation, if compared to digital signature creation. For this reason we graded μP ’s latency as “+”.

Peer-to-peer payments: μP was not conceived to make peer-to-peer payments. The grade for this parameter is “does not apply”.

Anonymity: The current implementation of μP does not allow any level of anonymity. The user has to be registered and for this he or she has to input personal data to the system. However, there is a possible

implementation of an anonymous *μP*: anonymous user-cards would be sold containing hidden passwords that would allow the card holder to access the system anonymously. This is future work, but it raises the grade of this parameter to a “-”.

5. CONCLUSIONS

Electronic commerce is steadily growing. This growth leads to new business possibilities, among them the distribution of digital information. It requires a new payment system capable of handling transactions with low cost and latency.

Micropayment systems have the necessary features for small amount transactions through simple and reasonably secure operations. The security model for such systems is a simplified one, as a result of a trade-off between the cost of implementation and the required security level for the transactions.

μP is an experimental but totally functional micropayment system, developed on the grounds of the PayWord system proposed by [Rivest-96]. The main characteristic of *μP* is the use of an electronic token generated in a central entity but verified as authentic in a distributed way by simple and fast computing operations. It differentiates from PayWord by making use of a unique set of tokens that can be used with any registered vendor. Its web interface makes it easily distributed throughout the various platforms that form the Internet. A more detailed comparison between the two systems can be seen in table 2.

Table 2. PayWord and μP comparison

PayWord	μP
One certificate per vendor per day	A single certificate for all vendors. Also accepts single-vendor certificates
User computes digital signatures	User does not compute digital signatures
Token authentication through one hash function computing	Token authentication via hash function computing and one digital signature
Tokens are generated by user	Tokens are generated by agent
Vendor sends to agent the last token received from each user each day	Vendor sends to agent all tokens received
Good for frequent relationship between user and vendor	Good for occasional relationship between user and vendor
Agent verifies tokens against user certificates	Agent checks tokens against central database
Credit-based	Debit-based (prepaid)

The micropayment system can be used to purchase papers in the Internet. Normally, in the case of a newsletter or magazine, the subscriber have access to use the full services by signature. A non-subscriber could use the micropayment system to buy only some pages of that newsletter or magazine. At this moment, the developed system is in an experimental phase at a banking site. The current purpose is to evaluate the technical and commercial feasibility of this solution to selling images and news.

REFERENCES

1. Anderson, R.; Manifavas, C; Sutherland, C. Netcard – a practical electronic cash system. <http://www.citeseer.nj.nec.com/168207.html>.
2. Hauser, R.; Steiner, M.; Waidner, M. Micro-Payments based on iKP. [S.l.], 1996. Technical report. <http://www.citeseer.nj.nec.com/hauser96micropayment.html>.
3. Manasse, M. et al. The millicent protocol for inexpensive electronic commerce. 1995. <http://www.w3.org/Conferences/WWW4/Papers/246/>.
4. Micali, S.; Rivest, R. L. Micropayments revisited. In: CT-RSA. [S.l.: s.n.], 2002. p. 149–163. <http://www.citeseer.nj.nec.com/502775.html>.
5. Mindlin, P.A.L. μP : Uma solução de micropagamentos. Master thesis-PCS/EPUSP. 2002.
6. Mu, Y.; Varadharajan, V.; Lin, Y.-X. New micropayment schemes based on paywords. In: Australasian Conference on Information Security and Privacy. [S.l.: s.n.], 1997. p. 283–293. <http://www.citeseer.nj.nec.com/mu97new.html>.
7. NIST. Fips publication 180: Secure hash standard (shs). National Institute of Standards and Technology. May 1993.
8. Pedersen, T. P. Electronic payments of small amounts. In: Security Protocols Workshop. [S.l.: s.n.], 1996. p. 59–68. <http://www.citeseer.nj.nec.com/pedersen96electronic.html>.
9. Rivest, R. The md5 message-digest algorithm. RFC 1321. April 1992.
10. Rivest, R. L. Electronic lottery tickets as micropayments. In: HIRSCHFELD, R. (Ed.). Financial Cryptography. Anguilla, British West Indies: Springer, 1997. p. 307–314. <http://www.citeseer.nj.nec.com/rivest98electronic.html>.
11. Rivest, R. L.; Shamir, A. Payword and micromint: Two simple micropayment schemes. In: Security Protocols Workshop. [S.l.: s.n.], 1996. p. 69–87. <http://www.citeseer.nj.nec.com/rivest96payword.html>.
12. Rivest, R. L.; Shamir, A.; Adelman, L. M. A Method for Obtaining Digital Signatures and Public-key Cryptosystem. [S.l.], 1977. 15 p. <http://www.citeseer.nj.nec.com/rivest78method.html>.
13. Schmidt, C.; Müller, R. A framework for micropayment evaluation. <http://citeseer.nj.nec.com/172517.html>.
14. Schweitzer, C. M. et al. Sistema para Controle de Distribuição Segura de Imagens. São Paulo, 2001. Technical Report LARC - PCS - EPUSP / Scopus.