Micropayments Revisited

Silvio Micali and Ronald L. Rivest

Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, rivest@mit.edu, silvio@tiac.net

Abstract. We present new micropayment schemes that are more efficient and user friendly than previous ones.
These schemes reduce bank processing costs by several orders of magnitude, while preserving a simple interface for both users and merchants. The schemes utilize a probabilistic deposit protocol that, in some of the schemes, may be entirely hidden from the users.

1 Introduction

A payment scheme consists of a set of protocols involving (at least) three basic parties: the buyer or user, the merchant, and the bank. These could be individual entities —such people, devices, computer programs— or collections of entities.[1]

ELECTRONIC CHECKS. The simplest form of payment scheme is an electronic check. Very informally, this consists of a check that is digitally signed rather hand-signed. In essence, a user pays a merchant for a transaction by digitally signing a piece of data that identifies the transaction (the user, the merchant, the "merchandise," the amount to be paid, the time, etc.) and possibly other data (such as account and credit information). The merchant deposits an electronic check by sending it to the bank. Having verified that the check is genuine and that has been deposited for the first time, the bank credits the merchant with the proper amount and charges the buyer with the same amount.

IT GOES WITHOUT SAYING. A few side details are supposed to be included in all payment schemes discussed herein. For instance, the parties' digital signature capabilities may supported by digital certificates (proving that they own their public keys, or that they are authorized to conduct electronic payments). These certificates can be sent alongside other messages, and their current validity may be verified before accepting an electronic payment. (See [13] for very efficient ways to verify certificates' current validity.)

When we say that the bank credits the merchant with $\$x$ and charges the user with the same amount, we do not mean that these amounts are exactly equal: the bank may deduct fees from the merchant's credit or add fees to the

[1] Of course, in a payment system there may be a plurality of users and merchants, and there may be too a plurality of banks. Indeed, in each transaction the user may have his own bank and the merchant his own, separate bank.

user's charge. The bank may also impose entry or subscription fees to users and merchants for their participation in the payment system.

A payment scheme per se does not provide the assurance that the merchant will deliver the relevant "merchandise" (goods, services, information, etc.). Other schemes such as that of [12] may guarantee that a "fair exchange" takes place. Also, a payment scheme per se does not guarantee that the user has enough money to pay. A separate mechanism may insure the merchant against this risk (e.g., the bank may guarantee payment if the user had a valid certificate).

Anonymity is not explicitly added to our payment schemes, though it can be added to them. [2]

MICROPAYMENTS. Payment schemes that emphasize the ability to make payments of small amounts are called *micropayment* schemes. Several micropayment schemes have been suggested, including *Millicent*[10] by Manasse et al., the *Payword* and *MicroMint*[16] schemes of Rivest and Shamir, Anderson's *NetCard*[2] scheme, Jutla and Yung's *PayTree*[7], Hauser et al.'s *Micro iKP*[5], the "Micropayment Transfer Protocol (MPTP)" of the W3C [4,18], the probabilistic polling scheme[6] of Jarecki and Odlyzko, the "Electronic Lottery Ticket" proposal[15] of Rivest, Wheeler's similar "Transactions Usings Bets"[19], Pedersen's similar scheme[14], and the related proposal for micropayments by efficient coin-flipping by Lipton and Ostrovsky[9] (this last paper includes an excellent survey of the field).

Applications of micropayments include paying for each web page visited, and for each minute of music or video as it is streamed to the user.

In principle, micropayments could be implemented by electronic checks. Merchant and user are otherwise engaged in the transaction, and the computation time they may devote to digital signing is not a real problem. The real problem lies with the *bank's processing cost* of any form of payment. Indeed, it seems likely that the cost of doing any transaction with the bank will be many times larger than the value of the micropayment itself. For example, processing a credit card transaction costs about 25 cents today, while a typical micropayment may be worth 1 cent. This processing cost seems unlikely to diminish dramatically in the near future.

Fortunately, while users and merchants are expected to be involved in every transaction, there is no essential reason why the bank must do work proportional to the number of transactions made. Micropayment schemes thus try to aggregate many small payments into fewer, larger payments, whose processing costs (by the bank) are relatively small.

1.1 Two Old Micropayment Schemes

Let us quickly recall two well-known micropayment schemes: (1) Rivest and Shamir's "PayWord" [16], and (2) Rivest's "electronic lottery tickets as micro-

[2] Payment schemes that emphasize user anonymity are often called *electronic coin* schemes. See Law et al. [8] for an excellent overview of electronic coin schemes.

payments" [15]. Indeed, our proposed micropayment schemes retain some of their ideas, and fix some of their drawbacks.

PAYWORD. Let H be a one-way function; that is, a function easy to evaluate but hard to invert. A user computes a "H-chain" consisting of values

$$x_0, x_1, x_2, \ldots, x_n$$

where

$$x_i = H(x_{i+1}) \quad \text{for } i = 0, 1, \ldots, n-1,$$

and commits to the entire chain by sending his signature of the root x_0 to the merchant. After that, each successive payment of the user is made by releasing the next consecutive value in the chain, which can be verified by checking that it hashes to the previous element. This allows the merchant to conveniently aggregate the buyer's payments as follows. Assume that the user has made i micropayments, and the merchants feels that, taken together, they constitute a sizable enough macropayment. Then, the merchant can make a single deposit for i cents by giving the bank only two values: x_i and the user's signature of x_0. The bank can verifies the user's signature of x_0 and iterates H on x_i i times, to verify that this operation yields x_0.

PAYWORD'S PROBLEM. PayWord suffers from a main problem:

> *A merchant cannot aggregate micropayments of different users.* Each user has established his own H-chain with the merchant, and there is no way of "merging" different chains. Thus, if a user spends only 1 cent with a given merchant, to deposit that cent the merchant or the bank would lose money, because the deposit's processing cost will exceed the payment value.

RIVEST'S LOTTERY SCHEME. In this scheme, there is a known *selection rate*, s, between 0 and 1. For each micropayment user and merchant interact according to a pre-determined protocol so as to *select* it with probability s: a non-selected micropayment is worthless and can be discarded, while a selected one can be deposited for an amount $1/s$ times bigger than the originally specified amount. For instance, if $s = 1/1000$ and each micropayment is for 1 cent, then on the average, out of 1000 micropayments, 999 will be discarded and 1 will be deposited for 1000 cents (i.e., for $10), thus incurring only a single processing cost. This approach minimizes processing costs, while on the average every one pays and receives what he should.

A fast implementation of the lottery scheme also uses H-chains as in PayWord. In a first step, the merchant gives the user the root $w = w_0$ of a H-chain

$$w_0, w_1, w_2, \ldots, w_n \ .$$

where

$$w_i = H(w_{i+1}) \quad \text{for } i = 0, 1, \ldots, n-1,$$

In a second step, the user includes w into his commitment to x_0, and thus into his commitment to his own H-chain. Assuming that the selection rate is 1/1000, then the user's i-th is payment is selected if (x_i mod 1000) is the same as (w_i mod 1000). (A mathematically rigorous version of this approach can be found in [9].)

Note that, at the end of the selection protocol, the merchant learns whether a given micropayment has been selected. In principle therefore, he might deny giving the user the expected merchandise if the payment was not selected. But this cheating possibility is not worrysome: since we are dealing with a micropayment, the merchandise involved may be worth 1 cent; while the merchant's reputation and his ability to continue to conduct business through electronic micropayments is worth much more.

THE LOTTERY SCHEME'S PROBLEMS. The scheme suffers from two main problems:

1. *Interaction.* The user and the merchant must interact to select micropayments. This interaction slows down the whole process and makes it impractical for some applications.
2. *User risk.* The scheme burdens the user with the risk that he may have to pay more than he should. For instance, though the selection rate is 1/1000, 10 (rather than 1) of his first 1000 micropayments may be selected for a macropayment! This may be a rare problem, because the probability of its happening is small, and its relative impact decreases dramatically with the number of micropayments made. Nonetheless, it may constitute a strong psychological obstacle to the wide acceptance of the scheme, because ordinary users are not accustomed to managing risk.

The lottery scheme can also be implemented using an external entity. For instance, in the example above, the w_i may not be committed in advance by the merchant, but may consist of the winning numbers of the state lottery. This too, however, poses problems. Namely, the merchant may have to store a large number of payments, because he must wait for the next lottery outcome to determine which of them will be selected. In addition, the merchant may conspire with the external entity against the user. For instance, he may arrange for certain values w_i to "pop-up" so as to guarantee that a given user's payments are always selected. Alternatively, the user may conspire with the external entity against the merchant.

1.2 Three New Micropayment Schemes

We put forward three micropayment schemes that solve the above mentioned problems.

MR1 Scheme "MR1", presented in Section 2, improves on Rivest's lottery scheme by making it *non-interactive* while *allowing the merchant to learn immediately whether or not a check is selected for payment.* We achieve non

interaction by (among other ways) properly utilizing the merchant's public key in the payment protocol.

MR2 Scheme "MR2", presented in Section 3, solves the problem that probabilistic schemes such as Rivest's lottery scheme and MR1 may —by bad luck— charge the user more than the total value of the checks he has written. We solve this problem by modifying the charging protocol to depend properly on the serial numbers of the user's checks.

MR3 Scheme "MR3", presented in Section 4, is a variant which trades off the immediacy (for the merchant) of finding out which checks are payable in favor of giving the bank greater control and flexibility over the deposit process.

In all our three schemes, each micropayment requires the computation of a digital signature, just as for an electronic check. A few years ago, this computational requirement was significant. But digital signature technology has continued to improve, and signature schemes that are both more secure and significantly faster are currently available. Moreover, the computational cost of a public-key signature has continued to decrease due to the deployment of more powerful processors. Furthermore, we note that the use of "on-line/off-line" digital signatures (as proposed in [3] and recently improved in [17]) may be a good choice for micropayment schemes.[3] In sum, we now feel free to utilize public-key computations even in most micropayment schemes.

2 The MR1 Scheme

In this section we improve Rivest's lottery scheme. As before, payments will be selected to be deposited according to a selection rate s, but the MR1 scheme *does not require interaction* between user and merchant, and yet *the merchant learns right away which payments are selected.*

These features make the MR1 scheme eligible for new uses. For example, it could be used by a packet to pay for its bandwidth as it travels along a precomputed route, where the intermediate routers have known public keys. (Note that having the router interact with the user sending the packet is not a realistic option!)

The idea is that when a user sends a (micro) check C to a merchant, C will be selected for (a macro) deposit if a given property holds between C and a unique quantity dependent on C that is easily computable by the merchant, but unpredictable to the user. Therefore, when the user sends a check C to the merchant, he has no idea whether C will be selected for deposit (and thus the user cannot avoid a possible deposit).

[3] Such digital signature are in fact computed in two steps: first an "off-line" step, computationally more demanding but performed before the message to be signed is chosen or available, and then a light-weight "on-line" step, performed once the message to be signed becomes available. Thus, for example, we imagine that the off-line step could be performed while a human user is deciding what to do next; once she decides, the signature for the micropayment can be completed quickly.

Notice that our proposal differs significantly from Rivest's lottery scheme not only at an implementation level, but also at a high level. In Rivest's case, a check sent by the user to the merchant is not determined to be payable or unpayable. Rather, its "payability" will be determined by the execution of the selection protocol that ensues, and thus by the random choices that merchant and user will make. In our case, instead, any check sent by the user to the merchant is already "pre-selected" for payability.[4] In some sense, each check carries its own special mark, "payable" or "unpayable", so as to guarantee the following three properties (1) the fraction of checks marked payable is approximately s, (2) the user cannot read the mark of the checks he sends, (3) the merchant can read the mark and can make it visible to others.

Let us first describe our preferred, practical implementation of the MR1 scheme, and then some of its variants, including a theoretical one for which a rigorous proof of correctness could be provided.

2.1 The preferred embodiment of MR1.

PRELIMINARIES.

Let T denote (the encoding of) a transaction. We assume that T specifies all quantities deemed useful—such as the user, the merchant, the bank, the "merchandise", the transaction time, the transaction's (monetary) value, etc.

For simplicity, we assume that each transaction has a fixed value (equal to 1 cent) and that there is a fixed *selection rate*, denoted by s. (I.e., s is the fraction of payments that is expected to be selected for deposit.)

We let $F(\cdot)$ denote a fixed public function that takes arbitrary bit strings as input and returns as output a number between 0 and 1, inclusive. For example, F might operate by taking the input string e and pre-pending a zero and a point, and interpreting the result as a binary number, so that for example "011" becomes "0.011" which is interpreted as the number 3/8. The function F might also apply a standard hash function to the input as an additional first step.

If X is a party and Y a message, we denote by $SIG_X(Y)$ X's digital signature of Y. For simplicity, we assume that each message is one-way hashed prior to being signed and is explicitly included in its own signature.[5].

THE PREFERRED MR1 SCHEME

Set up: Each user and each merchant establishes his own public key (with the corresponding secret key) for a secure digital signature scheme. The merchant's digital signature scheme must be *deterministic*.

[4] If you want, a check's probability of being selected depends on choices made by the merchant during a set-up phase —e.g., on which public signature key the merchant actually chose.

[5] For example, $SIG_X(Y)$ could be implemented as $(Y, SIG_X(H(Y))$, where H is a fixed public one-way hash function.

Payment: A user U pays a merchant M for a transaction T (with selection rate s) by sending M the check $C = SIG_U(T)$.
Check C is actually *payable* if $F(SIG_M(C)) < s$. If C is payable, then M sends bank B both C and $SIG_M(C)$ for deposit. (Out of courtesy, M may inform U whether C was payable.)

Selective Deposit: If U's and M's signatures are correct, and C is a previously undeposited payable check, then B credits M's account with $1/s$ cents and debits U's account with the same amount (and may justify its action by providing U with $SIG_M(C)$).

Basic Properties.

- *The set-up phase is simple and general.* There is no need of a separate set-up for each user-merchant pair as in PayWord.
- *The payment phase is non-interactive.* The user simply sends a signed message to the merchant, to which the merchant needs not to respond.
- *The selection rate is s.* In fact, $SIG_M(C)$ is a quantity unpredictable to U, because U does not know M's secret signing key. Thus, practically speaking, even if U may control C in any way he wants (e.g. by choosing the transaction T), $SIG_M(C)$ will essentially be a random number. Therefore, $F(SIG(C))$ is a random and long enough number between 0 and 1, and thus will be less than the selection rate s essentially for a fraction s of the checks C.
(Note that for a reasonable selection rate, such as 1/1024, it would be sufficient for $F(SIG_M(C))$ to be 10-bit long. A typical signature is instead hundreds of bits long, which is an "overkill.")
- *Bank B is called into action only for a fraction s of the micropayments, and once it acts it does so only on macropayments.* The merchant can immediately verify whether a check C is payable, because he can easily evaluate $F(SIG_M(C))$ and compare it to the selection rate. Thus, the every check forwarded by the merchant to the bank is payable, and each such check results in a "macropayment" because it has a selected value of $1/s$ cents. (e.g., if $s = 1000$, it has a value of \$10). Thus, the system generates only relatively negligible transaction costs.
- *No two parties can successfully cheat the third one.* Even with B's help, U cannot write a check that has "less than s chance" of being payable. Indeed, the value $SIG_M(C)$ is unpredictable to both B and U, even if they share information about $SIG_M(C')$ for any prior check C' and even if they jointly choose C. Similarly, M and B together cannot defraud U. Informally, once the public signature key of M is chosen in the set-up stage (by M alone or by M and B together), because the signature scheme is deterministic, there is only one possible value $SIG_M(C)$ for every check C, and thus no amount of conspiracy may change that value. Moreover, when M's public key is chosen, M and B do not know what U's checks look like. Even if B and M were capable of guessing or controlling which transactions U will execute, they cannot choose M's public key so as to guarantee that U's checks will be

payable with probability greater than s. In fact, U's check for a transaction T consists of $SIG_U(T)$, and is thus unpredictable to both M and B. Finally, notice that the bank cannot be defrauded in the sense that, each time that B pays M, B withdraws the same amount from U's account. (The problem that U may be unable to pay his checks arises also for ordinary checks, and is therefore independent of our cryptographic scheme and should be handled as usual.)

2.2 Variants of the preferred embodiment of MR1

THEORETICAL VARIANTS. The basic scheme can be modified a bit so as to formally achieve security.

In our analysis of the MR1 scheme, it is crucial that $F(SIG_M(C))$ be a ("random") number of sufficient precision unpredictable to the user. Whether this security condition holds may depend on the signature scheme and on the definition of F.

For example, consider an F that returns the binary fraction whose representation is the low-order 20 bits of its input. The security condition is immediately true if one models digital signature schemes as random oracles; it is however more traditional to model only one-way hash functions as random oracles. The condition nonetheless holds if the merchant digitally signs using a suitable signature scheme such as RSA. In fact RSA is a deterministic signature scheme and it has been proven in [1] that, relative to a randomly chosen RSA public key of L bits, the last $c \cdot \log L$ bits of the signature of a random message are computationally indistinguishable from a random $c \cdot \log L$-bit string, where c is any constant greater than 1. As a consequence of this result, if merchant M randomly chose his RSA keys as 1024-bit long strings, then, letting $c = 2$, the last 20 bits of $SIG_M(C)$ provide a 20-bit string Y essentially indistinguishable from a random 20-bit string, no matter how C was chosen. (Recall in fact that rather than signing C directly, merchant M actually signs $H(C)$. Modeling the one-way hash function H as a random oracle, $H(C)$ would be random even if C were specially selected.) Notice that using even as few as 20 bits for Y enables one to easily implement a selection rate as low as 2^{-20} and provides sufficient resolution for most purposes; with $s = 2^{-20}$ selected 1-cent micropayments are transformed into macropayments worth $10,000 each, whose processing costs would be quite negligible.

The same crucial point can also be formally solved without recourse to any random oracle model. Namely, it would suffice for the merchant to use a verifiable random function (VRF) rather than an ordinary digital signature scheme. As introduced and exemplified by Micali, Rabin and Vadhan [11], a VRF comprises a pair of keys and a pair of algorithms: a public key PK, a matching secret key SK, an evaluation algorithm E, and a verification algorithm V. Key PK totally specifies a function F $(= F_{PK})$, from arbitrary bit strings to k-bit strings, such that it is hard to compute $F(x)$ on input x and PK. Key SK enables one to

evaluate F easily; that is, on inputs x, PK and SK, E returns $F(x)$ together with a proof P_x that indeed $F(x)$ is the correct value of F at point x. Proof P_x is accepted by V on additional inputs PK and x. The crucial property of a VRF is that $F(x)$ is polynomial-time indistinguishable from a random k-bit string for any input x for which a proof P_x has not been seen. (This remains true even if one is allowed to request and obtain $F(x')$ and $P_{x'}$ for any input $x' \neq x$ of his choice.) Thus, in the MR1 scheme, the merchant can select his own PK and SK and establish PK as his public VRF key, so that a check C becomes payable if $F_{PK}(C) < s$. Note that the merchant can immediately determine whether C is payable, because he knows SK and thus easily evaluates F_{PK}. Moreover he can enable the bank to verify that C is payable by releasing the proof P_C.

As for a different technical point, the user and merchant may choose their public signature keys by means of a mutually independent commitment scheme.

PRACTICAL VARIANTS. Different variants are possible that maintain the same (non-interactive) spirit of the MR1 scheme. In particular,

- *Time.* The basic scheme allows a merchant to deposit a payable check at any time. However, the bank may refuse to credit the merchant's account during the deposit phase unless he presents a payable check which has a sufficiently correct time. (E.g., if the transaction T to which a check C refers happened in day i, then the merchant should deposit C within the end of day i, or by day $i + 1$.) This gives an extra incentive to the merchant to verify the time accuracy of the checks he receives (which he should do anyway). Indeed, if the time is wrong, he could refuse to provide "the merchandise" requested. Timely deposit ensures that the user is not charged "too late," when he has no longer budgeted for that possible expenditure.
- *Functions F and G.* The functions F and G may not be fixed, but vary. For instance, a check or a transaction may specify which F or G should be used with it.
- The check-payability condition, $F(SIG_M(C)) < s$, could be replaced by $F(SIG_M(G(C))) < s$, where G is a given function/algorithm. So, rather than signing C itself, the merchant may sign a quantity dependent on C, denoted by $G(C)$. In particular, because C is U's signature of a transaction T, and because we assume that such a signature also specifies T, $G(C)$ may be a function of T alone, for instance a substring of T, such as T's date/time information. As for another example, T may also specify a user-selected string W, preferably unique to the transaction and selected at random, and $G(C)$ may just consist of W (so that the merchant will sign W, or W together with time information).
- The check-payability condition may be chosen in a rather different way. For instance, a check C may be payable if a given property holds between C and a quantity dependent on C that is computable only by the merchant, such as the property that the last 10 bits of C (or some specific 10 bits of T) equal the last 10 bits of $SIG_M(G(C))$.

- To minimize the merchant's number of signatures, rather than using $F(SIG_M(G(C)))$ to determine check payability, one may use $F(SIG_M(G(V_i)))$, where $\{V_i\}$ is a sequence of values associated to a sequence of times. For instance, V_i is a daily value and specifies the day in question (e.g., $V_i = 02.01.01$, $V_{i+1} = 02.02.01$, etc.) and a check C relative to a transaction T on day i may be payable if $F(SIG_M(V_i)) < s$ (or if some other property holds between C and a quantity computable from V_i only by M, such as whether the last 10 bits of C —or some specific 10 bits in T— equal the last 10 bits of $SIG_M(V_i)$).
  Note that the merchant may evaluate $F(SIG_M(G(V_i)))$ at the beginning of day/time interval i, so that, upon receiving a check C on that day/time interval, M may immediately discard C if it is not payable, and set C aside for proper credit otherwise.
  Note too that it is better in this variant for the merchant to hide all information about which checks he has discarded and which checks he has set aside for credit during a given day/time interval. Else malicious users may predict or infer somewhat $F(SIG_M(G(V_i)))$, and give M checks that are not payable or have less probability of being payable. For this reason, if the merchant uses a V_i-approach, we recommend that he stores all his payable checks of a given day/time interval, and then send all of them to the bank at the end the day/time interval. This way even a malicious bank cannot collude with a user so as to enable him to defraud the merchant.
- A special way to implement the above approach consists of utilizing a hash/one-way function chain. That is, the merchant computes a sequence of values

  $$x_0, x_1, x_2, \ldots, x_n$$

  where

  $$x_i = H(x_{i+1}) \quad \text{for } i = 0, 1, \ldots, n-1,$$

  where H is a one-way function/hash, and puts x_0 in his public file, or otherwise publicizes x_0 (e.g., by steps that include digitally signing it). Then one can use x_i, rather than $F(SIG_M(G(V_i)))$ on day/time interval i.
- It is easy to extend the basic MR1 scheme to handle checks of different values; everything is simply scaled appropriately for each check.

3 The MR2 Scheme

Recall that Rivest's lottery scheme suffered from two problems: (1) interaction in the payment process, and (2) the possibility of user's excessive payments. The MR1 scheme solved the first problem, but did not address the second one. Of the two problems, we regard the first one to be a real one, and the second to be mostly a "psychological" one. Indeed, the possibility that the user may be debited substantially more than the micropayments he makes is very small, and will decrease with the number of micropayments made. Nonetheless, user acceptance is key to making micropayments widely used.

Accordingly, in this section we present a selective-deposit micropayment scheme that solves both problems. In particular, it guarantees that a honest

user is never charged more than he actually spends. The small risk of excessive payment is shifted from the user to the bank. Note that this is much preferable for two reasons. First, as we said, excessive payment occurred only rarely (i.e., for few users) and in moderate amounts. Now if this may have bothered users, it will not bother banks who are actually accustomed to managing substantial risks, never mind the rare risk of a small excessive payment! Second, the relative risk becomes less and less probable in the long run, and thus is less probable for the bank, given that it will experience much higher volumes than a single user.

Another main attraction of the scheme is its extreme simplicity. Accordingly, rather than trying hard to prevent cheating, it simply punishes cheating parties, or purges them from the system before they can create any substantial damage.

PRELIMINARIES. We adopt the same simplifying assumptions and notations (about transactions, fixed monetary value, fixed selection rate, and digital signatures, etc.) as in the MR1 scheme.

THE BASIC SCHEME

Set up: Each user and each merchant establishes his own public key (with the corresponding secret key) for a secure digital signature scheme; the merchant's signature scheme must be deterministic.

Payment: A user U pays a merchant M for a transaction T (with selection rate s) by sending M the check $C = SIG_U(T)$. The user includes the time and a serial number SN in every check/transaction. (The serial numbers should start at 1 and be assigned sequentially.)

Check C is actually *payable* if $F(SIG_M(C)) < s$. If C is payable, then M sends bank B both C and $SIG_M(C)$ for deposit.

Selective Deposit: Let $maxSN_U$ denote the maximum serial number of a payable check of U processed by B so far (initially $maxSN_U = 0$). Assume that C is a new, payable check, and that U's and M's signatures are correct. Then the bank B credits M's account with $1/s$ cents. Furthermore, if the serial number SN of the check is greater than $maxSN_U$, the bank B debits U's account by $SN - maxSN$ cents, and sets $MaxSN_U \leftarrow SN$ —and may justify its action by providing U with $SIG_M(C)$.

(An exception to the above rules is made if the bank notices that the new check has the same serial number as a previously processed check, or if the new check's serial number and time are "out of order" somehow with respect to previously processed checks, or if the amount of the check is excessive, or if other bank-defined conditions occur. In such exceptional cases the bank may fine the user and/or take other actions as it deems appropriate.)

Selective Discharge: The bank may keep statistics and throw out of the system (e.g., by revoking their certificates) users whose payable checks cause exceptions (as noted above) because they are inconsistently numbered and/or dated, or whose checks are "more frequently payable than expected." It may similarly throw out merchants with whom such problematic or "more frequently payable" checks are spent.

BASIC PROPERTIES. As for the MR1 scheme, the set-up phase is simple and general, the payment phase is non-interactive, and the selection rate is s.

Let us now argue that the scheme is fair for the honest user. At any time t, an honest user U has been charged $maxSN$ cents if $maxSN$ is the highest serial number of U's successfully deposited checks. Assume now that, by time t, U has made n transactions. Then, because an honest user numbers his checks sequentially starting with 1, n will also be the highest serial number of any check that U has written, and thus $maxSN \leq n$. That is, U will have been charged at most (rather than exactly) 1 cent per transaction.

Out of courtesy, M may inform U when a check was payable, but in this scheme it is preferable that M does not so inform the user, and certainly unnecessary for him to do so. It is better to keep the user ignorant of which serial numbers have turned out not to be payable. Note that the user's cumulative charges do *not* depend much on which checks turned out to be payable, but only on the number of checks he has written.

To incur lesser charges, a malicious user U' may try to lower artificially $maxSN$ by using twice at least one serial number SN. Thus U' can be caught by B in at least two ways: (1) two checks of U' are deposited whose serial numbers and times are inconsistent, or (2) two checks of U' with the same serial number are deposited.[6] Thus, if a suitably high fine or punishment is imposed on users caught cheating (something that is preferably agreed-upon beforehand), then cheating would be counterproductive.

A malicious user U', however, may collude with a malicious merchant M', so as to ensure that a check of U' spent with M' is always payable. Indeed, for each potential check C, M' can tell U' the value of $SIG_{M'}(C)$, so it is no longer unpredictable to U' whether the check would be payable. With a little trial and error, U' only writes payable checks. This way, U' will always pay just 1 cent to B, while B will always pay $1/s$ cents (i.e., \$10 if $s = 1/1000$) to M'. U' and M' may then share their illegal proceeds: indeed, U' may coincide with M' if he sets himself up as a merchant! Nonetheless, U' and M' may only make a modest illegal gain: if they try to boost it by repeating it several times, they are likely to be thrown out of the system. (This is a high price to pay, particularly if M' also has legitimate gains in the system.) If it is not easy for thrown-out users and merchants to come back in the system (e.g., under a new identity), or if the price to get into the system (e.g., that of getting an initial certificate) is sufficiently high, this illegal game pays little or even has negative returns to the user, and its cost may be easily absorbed by the bank.

A small probability exist that a honest user may look malicious because he makes n checks and significantly more than n/s of them become payable. In this case, he may be thrown out. With appropriate parameter settings, there will be very few such users. In addition, they can be convinced that they unintentionally

[6] Note that way 1 can occur even if honest merchants check for the time accuracy of the checks they receive: there cannot be perfect accuracy. Note that way 2 may occur because U' does not control which of his checks spent with honest merchants become payable.

caused losses to the bank (e.g., because the bank presents them with the relevant $SIG_M(U)$ values for their checks). Therefore, they may accept being kept on the system under different conditions—for instance, as users of an MR1 system; that is, they may agree to be debited 1000 cents, from then on, for each payable check. (Such a transition to an MR1 system might even be an automatic feature of the original agreement between the user and the bank.)

VARIANTS. In general, variants of the MR1 scheme apply here too.

We note that to handle checks of different values needs a bit of care. In general, a check worth v cents should be treated as a bundle of v one-cent checks (with consecutive serial numbers). A bit more efficiently, the user may write a single check that, rather than having a traditional serial number has a serial-number interval, $[SN, SN + v]$. We leave as an exercise how to modify the MR2 scheme (and its "penalty system") so t handle properly such checks.

4 The MR3 Scheme

This scheme differs from both MR1 and MR2 in that the bank determines, probabilistically and fairly, which checks are payable. Again, the small risk of excessive payment is shifted from the user to the bank, which is accustomed to risk management. And again simplicity is a main attraction: rather than trying hard to prevent cheating, the bank simply punishes/eliminates cheating parties before they can create any substantial damage.

THE BASIC SCHEME

Set up: Each user and each merchant establishes his own public key for a secure digital signature scheme.

Payment: A user U pays a merchant M for a transaction T by sending M the check $C = SIG_U(T)$. The user includes in every check/transaction a progressive serial number SN.

Selective Deposit: Let t' and t denote, respectively, the time of M's last and current deposit. M groups all checks dated between t' and t into n lists, L_1,...,L_n. Denote by V_i the total value of the checks in L_i, and by V the sum of the V_i's. M computes a commitment C_i to list L_i, preferably together with V_i (e.g., practically speaking by one-way hashing them so that $C_i = H(L_i, V_i)$), and then sends $C_1, \ldots, C_n$ to B, preferably signed and with an indication of deposit time. For instance, M sends $SIG_M(t, n, V, H(L_1, V_1), \ldots, H(L_n, V_n))$ to B.

B verifies M's latest deposit time, and selects k indices, $i_1, i_2, ..., i_k$, and sends them to the merchant.

M responds by de-committing $C_{i_1}, \ldots, C_{i_k}$.

B credits M's account with V cents, and debits the users whose checks belong to $L_{i_1}, \ldots, L_{i_k}$ according to the serial numbers used —e.g., as in the MR2 scheme.

(An exception to the above rules is made if the bank notices that something is wrong. For instance, if the sum of the checks in L_{i_j} is not V_{i_j}, if one check in L_{i_j} has the wrong time, if a newly processed check has the same serial number as a previously processed check, or if the new check's serial number and time are "out of order" somehow with respect to previously processed checks, or if the amount of the check is excessive, or if other bank-defined conditions occur. In such exceptional cases the bank may fine and/or throw out of the system the merchant and/or the user, or take other actions as it deems appropriate.)

Selective Discharge: B may keep statistics and throw out of the system (e.g., by revoking their certificates) users or merchants who misbehave, those users U whose checks cause B to pay merchants more than it is entitled to receive from U, and the merchants with whom those users spend their checks.

BASIC PROPERTIES. As for the MR1 and MR2 schemes, the set-up phase is simple and general and the payment phase is non-interactive. Moreover, the present scheme is very understandable and looks very fair to the merchants.

Notice that the value of k is arbitrary and up to the bank. When there is more attempted fraud, or there is suspicion of a particular merchant, a larger value of k may be used. Indeed, B may ask the merchant to de-commit *all* of his commitments. (Failure to de-commit, in particular, may trigger a fine or a discharge of the merchant.) Choosing $k > 1$ is recommendable in order to having a chance to catch two checks from the same user with the same serial number (rather than throwing out such a user later on "statistical evidence").

Notice that the merchant may deposit at prescribed times, $t_1, t_2, \ldots$, or at times of his choice. For instance, at a time t in which he has new checks totaling a given value (so that he does not want to delay payment any further), or when he has sufficiently many new checks (and does not want to store them any more).

As in MR1 and MR2, users and merchant may collude, but again with little or no benefit, since the bank may adopt the same defense mechanisms. Honest users who look suspicious may be treated similarly too.

VARIANTS. Variants of the MR1/MR2 schemes may also be applied here. In addition, the merchant may make use of Merkle trees to commit to $L_1, \ldots, L_n$. Check value information may be authenticated within the tree or alongside with it. In particular, the Merkle tree may authenticate at each node the total value of the checks "stored below it," as well as the total value of the check stored below each child. The same holds for check time information. The root of the Merkle tree may be digitally signed by the merchant—possibly together with other data, such as (partial or total) check value and time information.

The value of n may be variable or fixed. The bank may choose k out the n lists to have the merchant decommit, and pay the merchant an amount that depends on k, n and the value in the checks contained in the k de-committed lists. For instance, if the total value of these checks is TV, the bank may pay nTV/k.

Banks and merchant may agree not to process deposits whose checks total more than a given value, or deposits containing a list totaling more than a given value. (This discourages a single cheating attempt with the goal of getting either a high payoff or going bust.)

5 Conclusions

We believe that the schemes presented here provide effective solutions to the micropayments problem. Malicious behavior of a single player is generally prevented by design, while malicious behavior of a coalition of players is dealt with by a penalty system backed up by hard evidence. From a user's point of view, the interface is beautifully simple: it is just like writing (small) checks. From the bank's point of view, it is just like processing (large) checks. And the merchant is happy, because he can efficiently aggregate small payments from many users.

References

1. W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. Rsa/rabin functions: Certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, June 13 1988.
2. Ross Anderson, Harry Manifavas, and Chris Sutherland. A practical electronic cash system. In *Proceedings Fourth Cambridge Workshop on Security Protocols*, volume ?? of *Lecture Notes in Computer Science*. Springer, 1996.
3. Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. In Gilles Brassard, editor, *Advances in Cryptology - Crypto '89*, pages 263–277, Berlin, 1989. Springer-Verlag. Lecture Notes in Computer Science Volume 435.
4. Phillip Hallam-Baker. W3C payments resources, 1995. `http://www.w3.org/hypertext/WWW/Payments/overview.html`.
5. Ralf Hauser, Michael Steiner, and Michael Waidner. Micro-payments based on iKP. Technical Report 2791 (# 89269), June 1996.
6. Stanislaw Jarecki and Andrew Odlyzko. An efficient micropayment scheme based on probabilistic polling. In *Proceedings 1997 Financial Cryptography Conference*, volume ??? of *Lecture Notes in Computer Science*, page ??? Springer, 1997.
7. Charanjit Jutla and Moti Yung. PayTree: "amortized-signature" for flexible MicroPayments. In *Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 213–221. USENIX, 1996.
8. Laurie Law, Susan Sabett, and Jerry Solinas. How to make a mint: the cryptography of anonymous electronic cash. National Security Agency, Office of Information Security Research and Technology, Cryptology Division, June 1996.
9. Richard J. Lipton and Rafail Ostrovsky. Micro-payments via efficient coin-flipping. In *Proceedings of Second Financial Cryptography Conference, '98*, volume 1465 of *Lecture Notes in Computer Science LNCS*, pages ??–??, February 1998.
10. Mark S. Manasse. Millicent (electronic microcommerce), 1995. `http://www.research.digital.com/SRC/personal/Mark_Manasse/uncommon/ucom.html`.
11. S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *Proc. 40th Symp. on Foundations of Computer Science*, pages 120–130, October 1999.

12. Silvio Micali. Certified e-mail with invisible post offices. In Proceedings RSA97, San Francisco, CA, January 1997. Also, U.S. Patent No. 5,666,420.
13. Silvio Micali. Efficient certificate revocation. In Proceedings RSA97, San Francisco, CA, January 1997. Also U.S. Patent No. 5,666,416.
14. Torben P. Pedersen. Electronic payments of small amounts. Technical Report DAIMI PB-495, Aarhus University, Computer Science Department, Århus, Denmark, August 1995.
15. Ronald L. Rivest. Electronic lottery tickets as micropayments. In *Proceedings of Financial Cryptography '97*, volume 1318 of *Lecture Notes in Computer Science*, pages 307–314. Springer, 1997. (Available as `http://theory.lcs.mit.edu/~rivest/lottery.pdf`).
16. Ronald L. Rivest and Adi Shamir. PayWord and MicroMint–two simple micropayment schemes. In Mark Lomas, editor, *Proceedings of 1996 International Workshop on Security Protocols*, volume 1189 of *Lecture Notes in Computer Science*, pages 69–87. Springer, 1997. (Also available in CryptoBytes, volume 2, number 1 (RSA Laboratories, Spring 1996), 7–11, and at `http://theory.lcs.mit.edu/~rivest/RivestShamir-mpay.pdf`).
17. Adi Shamir, 2001. Personal communication.
18. W3C. Micropayments overview. texttthttp://www.w3.org/ECommerce/Micropayments/.
19. David Wheeler. Transactions using bets. In Mark Lomas, editor, *Security Protocols*, volume 1189 of *Lecture Notes in Computer Science*, pages 89–92. Springer, 1996. (Also available by ftp from the server `ftp.cl.cam.ac.uk` as `/users/djw3/tub.ps`.).