Micro-Payments via Efficient Coin-Flipping

(Extended Abstract)

Richard J. Lipton[1] Rafail Ostrovsky[2]

[1] Department of Computer Science, Princeton University, Princeton, NJ and Bellcore, USA Email: rjlv@cs.princeton.edu.
[2] Bell Communications Research, 445 South St., MCC 1C-365B, Morristown, NJ 07960-6438, USA. Email: rafail@bellcore.com.

Abstract. We present an authenticated coin-flipping protocol and its proof of security. We demonstrate the applicability of our scheme for on-line randomized micro-payment protocols. We also review some essential aspects of other micro-payment proposals (including SET, PayWord and MicroMint, PayTree, NetCheque, NetCash, Agora, NetCard, CAFE, Pederson's proposal, micro-iKP, Milicent, proposal of Jarecki-Odlyzko, proposal of Yacobi, SVP, DigiCash, Rivest's "Lottery tickets as Micro-Cash" and Wheeler's proposal) and compare it with our scheme.

1 Design Principles and Parameters

This paper presents another micro-payment scheme, designed for world-wide web applications. It avoids many shortcomings of previous schemes. In particular, our scheme can support tiny transactions (like buying individual web-pages) at (amortized) cost of a fraction of a cent per web-page. We give an overview of other existing proposals and compare it with our scheme. In the heart of our construction is a new, on-line, fair and authenticated coin-flipping protocol, which is of independent interest and could be used in other settings as well. We start by surveying the setting and parameters considered.

The Participants: As in all the payment schemes, the main participants are the User $\mathcal{U}$ who wishes to get some information (i.e., a web page) from some Vendor $\mathcal{V}$ (i.e. from some web site, like on-line Encyclopedia Britannica). Vendor $\mathcal{V}$ wants to get paid for the provided information, while the User $\mathcal{U}$ wishes to pay only for the information that she gets. We operate in the setting where neither Vendors nor Users trust each other, hence, Vendors wish to make sure that they get paid for the provided information, while Users wish to make sure that they are not "over-charged" for the services that they did not get. Additionally, there is the third party, a Broker (or a Bank) $\mathcal{B}$ which assists in various ways for fund transfer between users and vendors and tries to detect/prevent various fraud of dishonest Users and Vendors. In the simplest setting, the Broker (or a Bank) is assumed to be trusted. More generally, some schemes do not assume that Brokers/Banks are trusted, and then introduce one or two more additional participants, such as central authority and/or one or several arbiters in order to resolve disputes and provide checks on other participants.

Design objectives: One of our goals is to minimize the computational requirements of our scheme. For micro-payments, this means minimizing public-key

cryptography in favor of faster private-key— and hash-function— based schemes. For example, as [27] point out: "as a rough guide, hash functions are about 100 times faster than RSA signature verification, and about 10,000 times faster then RSA signature generation: on a typical workstation, one can sign two messages per second, verify 200 signatures per second and compute 20,000 hash function values per second." (We remark that cryptographic hash-functions are in many of the above applications used solely as one-way functions which are one-way on their iterates – technically a weaker property then collision-resistance.) We also remark that private-key cryptography (for example, pseudo-random generators) is often even more efficient. Summarizing, one of our design objectives is to make use of efficient one-way functions (like MD5 [26]) and/or private-key cryptography and to minimize the use of digital signatures. Additionally, we wish to minimize the communication (both the number of rounds and the number of bits transmitted) per transaction, between all the parties, as well as computational requirements of our scheme and memory requirements for all the participants. We also wish to minimize potential fraud, which we elaborate upon further after reviewing previous proposals. To summarize, we wish to optimize the following parameters:

- minimize the number of rounds of interaction per transaction between users and vendors and between the bank/broker and users and vendors;
- minimize the number of total bits transmitted per transaction between users, vendors and the bank/broker;
- minimize the computational demands needed per transaction for all the participants (i.e. minimizing the use of digital signatures in favor of less-expensive means — see above);
- minimize hardware requirements for all the participants (i.e. eliminate and/or minimize the use of large databases of revocation lists or other "per-transaction" lists; and/or the need of smart-cards; and/or expensive hardware for pre-processing);
- minimize fraud (to be discussed in below after surveying other schemes.)

Additionally, we discuss the issue of anonymity, which plays a role in some micro-payment schemes, such as DigiCash [8]. Informally, the goal of anonymity is to minimize the user identification (both to the vendor and to the bank) when purchases are made. We remark that our schemes can be made anonymous as well.

OUR RESULT: In the heart of our construction is an authenticated coin-flipping protocol. The protocol requires evaluation (and transmission) of only two hash functions per coin-flip after the initial setup. The *authentication* in our protocols guarantees (among other things) that the vendor can request a third party (such as bank/arbiter) to verify what the outcome of the coin-flip should be, and if the protocol is aborted in the middle, to prove (to arbiter/bank) what the outcome is and insist on the resumption from the right point in the protocol execution.

We show how our protocols can be utilized to efficiently implement a coin flipping protocol of [31] where with some small probability (say, $\frac{1}{200}$) the user pays a larger (say, 1$ dollar) amount. Notice that the *expected* cost per transaction in the above example is half a cent, while the *expected* overhead of handling

the payment is now two hundred times smaller, thus allowing us to use (in the event of the "payment"-outcome coin-flip) an alternative, slow but secure payment mechanism. Our scheme is related to that of [24, 25], but with some important differences, especially in the setup stage. We discuss these differences and why they seem to be essential for the proof of security. In summary, the main technical contribution of this paper is the design of *fair* and *authenticated* coin-flipping protocol together with its proof of security.

2 Previous schemes and Techniques

Credit-card setting and On-Line Schemes: In the current credit-card setting, every transaction is on-line, where whenever some customer wishes to pay a vendor, the Bank is always contacted. In particular, the Bank gets a request to transfer money from user's account to vendor's account. In order to do so, the bank first verifies that the customer's account is in good standing (which requires a database lookup) and then gives to the vendor a validation number for the transaction. If Vendor's account is with a different Bank, this contact is also made at some point in time, different for different schemes. The cost per such transaction is about 10 cents, and hence is not financially viable for tiny-cost transactions. Moreover, since the Bank must maintain 99.99% availability, even during peak traffic time (Anderson at. all. [1] mention that typically 1pm on the Saturday before Christmas is such a peak-time) this requires additional cost in order to maintain capability for additional throughput and backup systems.

The main source of fraud in the current credit-card practice is from stolen [credit-card number, expiration-date] information, which can (and is) used to impersonate users.

SET: Recently, Visa and Master-card developed a SET on-line analog of the credit-card setting, where digital signatures are used to authenticate all three parties (i.e. users, vendors and banks), so that an adversary who wire-taps all the communications, still can not impersonate users, since he can not forge signatures (for further details and features, see [30]). However, since signature generation and verification is required for all the parties, and since the Bank must be present on-line for every transaction, SET is clearly not suitable for tiny web-related transactions.

NetBill: NetBill [6] is another on-line protocol (it has additional features like atomicity – i.e. customer pays only for messages that he gets; and anonymity via pseudonyms). It requires eight messages for each transaction and on-line communication with the intermediary NetBill server for each transaction.

Electronic currency: DigiCash; NetCash. In electronic currency schemes, a user deposits some amount of money into the bank, that in return gives some digital data representing "Electronic currency" (also called an electronic "coin"). In its simplest form, electronic currency is an authenticated (by the bank) serial number. Of course, the danger with any such scheme is double-spending (i.e. where a user or someone else "spends" the same "coin" more the once). There are several ways to combat this: one is to insist on the on-line check (with the

Bank) to verify if the coin have been already spent; the second is for the Bank to pay for each coin only once [27]; the third is to incorporate the identity of the user (who bought an electronic coin) into the coin itself, so that if it is spent twice, the known user will be prosecuted. A twist proposed by Chaum is to also have anonymity, where again Bank keeps track which "coins" have been spent, but where DigiCash [8] in case of double-spending reveals the identity of the User (for later prosecution – see also [9]). The issue, of course, is when to check for double-spending. One possibility (which *is* what Chaum's [8] current implementation does) is to do the check on-line, the other possibility is to check off-line, running the risk that the user will spend huge sums of money and then disappear, when discovering fraudulent activity is too late. Another "electronic currency" scheme is that of Medvinsky and Newman "NetCash" scheme [19]. The twist there is that they keep track only of the outstanding "tokens" (i.e. those that have been issued but have not been deposited), compared to Chaum's scheme where comparison with all tokens ever issued must be made. Never-the-less, all these schemes must either be on-line, or stand the risk of "huge-spending-with-quick-gateway" attack.

NetCheque: University of Southern California NetCheque (TM) project [21] is another on-line scheme, where users issue checks for vendors using (as a certificates of the validity of a check) a private-key cryptography (shared between a user and a bank.) This scheme is more efficient then public-key signatures, but requires registration of users at the banks and then subsequent clearance (i.e. checking by the bank of the validity of the private-key authentication) of checks by the bank to verify both correctness of the check and the availability of funds in user's account. Neuman and Medvinsky [21] argue that such on-line verification can be in some cases done off-line, but then the fraud (of bad checks) becomes as issue.

Hardware-Based Schemes In hardware-based schemes, one assumes the existence of temper-resistant smart-cards, which contain private-keys, but such that this private-keys can not be extracted from the card without destroying its contents. This, in principle, allows to use faster private-key cryptographic means. One such example is the proposal of Stern and Vaudeny [29]. They offer a scheme where every smart card contains a master private key for MAC (i.e., private-key Message Authentication Codes). Every Vendor is given such a tamper-resistant smart card. Users buy from the bank "tokens" which are authenticated with banks private-key authentication mechanism. When a user wishes to make a payment, it gives the token that was provided by the bank to the Vendor's tamper-resistant devise which then verifies that it is a valid payment, and then gets paid from the bank. The idea is that the private key is known only by the bank, yet all vendors can verify that users provide correct private-key authentication tags, since they all have smart-cards with the banks private key. Again, clearance of this payments can be done either on-line (to prevent double-spending) or off-line (with black-listing double-spending users) as before. The drawback of this scheme is that if the unique bank's key on each of the vendors smart cards is recovered by an adversary, the security is lost. That is, the single private-key of the bank is distributed to every device, hence breaking even a single tamper-resistant device completely compromises the scheme. They suggest an extension which uses logarithmic (in the number of users) keys,

and authenticating with all of them, but this, while it prevents the above threat makes the proposal less efficient.

Various "electronic-wallet" hardware-based solutions are proposed by Mondex [18] and others, such as Yacobi e-war [32]. There are basically two different approaches. The first approach is to keep the actual counter on the card – which increases/decreases during transactions, where the counter denotes the actual cash value (up to a certain limit). Since the counter is on the tamper-proof device it can not be increased. Of course, if one can artificially increase the counter, this leads to money forgery. Another approach, is to keep digitally signed (by the bank) coins, where the only goal of the tamper-resistant device is to prevent double-spending [32] (where various methods (either on-line or off-line and/or probabilistic) are made to detect if some coin is spent twice and by what smartcard – the scheme requires revocation lists and if not done on-line has some over-spending threat.)

Rivest and Shamir's Micro-Mint [27] propose to run (using huge off-line computations) schemes which find collisions of (properly tuned, only somewhat hard to find) collisions of collision-free hash-functions, to be used instead of signatures. This guarantees that forging is hard (basically using birthday-paradox type argument) but still duplication easy. They argue that duplication is not an issue since every coin will be paid only once (which requires storage of all the coins as well).

SUBSCRIPTION SCHEMES: Many large vendors, sell *subscriptions* for certain websites. Examples include $ 40 year-based subscription to on-line Wall-Street journal and $ 150 year-based subscription to Encyclopedia Britannica. The subscription payment is performed only once a year (by various means). Clearly, one of the drawbacks is that infrequent customers are not willing to pay a relatively high subscription cost. Moreover, the subscription-method is not suitable for "infrequently used" vendors of specialized information (like consumer reports information on how to purchase a new car). Besides, there is only so many subscriptions any user will sign-on, even if the cost of subscription is falling. In summary, while subscription-based approaches are and will be in use, an additional micro-payment approaches are needed as well.

Another variant on the subscription scheme is a *registration scheme* where first, customers *register* with the Vendor and prove its identity and then Vendors regularly *charge* them for transactions made. One such scheme is the "Chrg-http' protocol [7]. The drawback is the cost of registration, and, of subsequent unpaid charges.

In light of the above, the development for tiny "per-use" payments schemes received considerable attention in the last two years. Below, we review various proposals.

COUPON-BASED SCHEMES DIGITAL's *Milicent* [11] is basically a private-key solution, where there is a "broker" which sells "vendor-specific" coins. Vendor-specific coin can only be authenticated by this vendor using vendor's private key (recall that the private-key authentication is much cheaper than public-key). Of course this solution requires "brokers" who must be trusted, and who should have agreements with vendors.

Another coupon-based family of (similar to each other) solutions is Rivest and Shamir's *PayWord*, [27], Anderson's *NetCard* [1], Pederson's at. al. scheme [23], Jutla and Yung *PayTree* [22] and Hauser at. al *micro-iKP* [15]. The top-level idea of all this schemes is basically one of Lamport's [17] (also used for S/key [14]), and it is as follows: take a one-way permutation f (or a hash-function or a one-way function which is one-way on its iterates), pick a random input x, and iterate it some sufficiently large number of times (say a 1000) (i.e. compute $y = f(f(f \ldots (f(x))))$), then authenticate y (i.e. sign y and perhaps user's ID with banks public key signature). Now we have a *chain* of values of the form $f^{-1}(y), f^{-1}(f^{-1}(y)) \ldots x$ with the property that given any prefix of this chain, it is hard to compute the next pre-image (since it involves inverting a one-way permutation) but easy to verify that this chain leads back to an authenticated y. The idea is for the bank to issue such $(x, y, \textit{bank's signature}(y))$ triple to the user (for the appropriate fee), where every inverse is a single micro-payment. The user, when he wishes to make a payment to the Vendor, gives y (with appropriate banks public-key authentication of y – this is a one-time setup operation) to the Vendor, but then for each subsequent payment just gives the next inverse in the above chain. Jutla and Yung [22] generalize this chains to trees in a natural way. The drawback of all this schemes is double-spending, where to combat this the two approaches being taken are either to check on-line (which is expensive) or to black-list users (which is somewhat expensive too, and may not be sufficient if user's identity can be easily changed/forged).

Probabilistic Schemes The probabilistic schemes can be divided into two categories: probabilistic checking and probabilistic payment. We first outline the probabilistic checking schemes and then describe the two previous probabilistic payment schemes.

The first two probabilistic checking protocols are **probabilistic audit** Agora protocol of Gabber and Silberschatz [10], and Jareski and Odlyzko probabilistic polling [16]. The idea there is basically as follows: the user gives (signed) promisory notes to the vendor, which the vendor later "cashes" to the bank, but when exactly this happens is done probabilistically, in order to limit the amount of over-spending. This approach combines (expensive) on-line approach of always verifying that the user has the money in his account (witch is communication-expensive) and the off-line credit-based solution (which leads to over-spending/ black-listing solution.) Here the over-spending (by tuning the rate with which vendor talks to the bank to be a probabilistic function depending of the transaction size) can be limited. The drawback is similar to coupon-based schemes, namely the requirement that in case of detected over-spending the vendors/banks must "black-list" users (and, hence keep such databases) and to inform all vendors of bad users [16], as well as keeping, by each Vendor the list of revoked users [10].

The combination of software-based and **hardware-based solution with probabilistic audit** was suggested by Yacobi's e-war [32] project at Microsoft. In [32] Yacobi proposes smart-card id-based wallets that keep signed by the bank coins. Notice that the new coins can not be forged since only bank can sign, and the duplication is controlled using hardware where probabilistic checking is used to prevent double-spending. This solution requires both software and hardware,

and can still take some amount of over-spending, though the amount can (as in the previous scheme) be made limited. The drawback is the need to black-list users/smart-cards and keep this databases around as above.

The final category of the probabilistic schemes is the so-called **probabilistic payment** category. Our scheme belongs in this category as well. The two other scheme in this category is Rivest's "lottery tickets as Micro-Cash" [24, 25] and Wheeler's "Transactions Using Bets" [31].

The [24] and [25] differ, and we review both. The idea of [24] scheme is for the bank to issue for each user a book of "lottery tickets" as follows: As in coupon-based schemes, the bank, picks a random x, computes $y = f(f(f \ldots (f(x))))$ for f a one-way permutation or a cryptographically-strong hash function, then authenticates y (i.e. signs y and perhaps user's ID with banks public key signature). Now we have a *chain* of values of the form $f^{-1}(y), f^{-1}(f^{-1}(y)) \ldots x$ which is a "lottery book" of tickets (for each user and each vendor that the user wishes to talk to), where for each micro-payment transaction, the user pays with the next pre-image from this book (just like the coupon-based scheme.) The twist here is that the bank, later on announces one of the tickets from each book as a "winning ticket". If the user did not give this winning ticket to a Vendor (since it stopped early and did not use the entire book), it does not have to pay anything, if it did, it is responsible for the winning ticket (i.e. the bank will pay the amount to the Vendor upon Vendor's presentation of the winning ticket and will subtract it from user's account). It is important to note that the "lottery" is held *after* the book in question (say for this day) is no longer in use, otherwise the user could always avoid giving out the winning ticket. The advantage of the scheme is that if, say, half of the lottery tickets from a book have been used, then with probability one-half the user will not have to pay, thus making the amortized cost of transactions less costly. The drawback is the Bank's overhead of holding lotteries and having to check the results as well as the issue of timing (since the payments to the Vendor can be made only after the lottery is announced, at which time the user may not have sufficient funds in its account.) In cite [25] (independently of our work) Rivest extends [24] suggestion using [31] approach and two chains, similar to our approach, but with some important differences. We first describe Wheeler's suggestion [31]:

The second probabilistic micro-payment scheme is the protocol of Wheeler "Transactions Using Bets" [31], where he suggests, similar to our scheme to decide probabilistically whether the payment should be made. In particular, he suggests for the user and vendor to execute a standard coin-flipping protocol [2] (vendor commits a random number to the user, then user sends a guess of this number to the vendor, then vendor de-commits) in order for the user and vendor to decide if the user should pay. One of the aspects not addressed by the paper, is that the user should not be able to deny that the coin-flip protocol execution took place, and hence that he has to pay the agreed-upon amount in case of an unfavorable coin-flip. A natural way of dealing with this problem is to introduce digital signatures into the protocol, so that the vendor can prove (to a bank/arbiter) that this interaction took place. However the use of signatures makes the protocol inefficient. Another drawback of the [31] protocol is the danger that the protocol is aborted in the middle of the execution. Indeed this is a serious problem, since if the seller/user are allowed to abort the coin-flipping protocols and re-try again, the probabilities can be altered. This problem is

indeed mentioned in the [31] paper, but no solution how to resolve this problem is given. In the current paper, we show how both problems can be resolved in an efficient manner.

The most related (to our scheme) is the work of Rivest [25], which suggests for the user and vendor to exchange roots of two chains, and show inverses in order to define coin-flips. However, there are several crucial differences in the two approaches, especially in the setup stage. We discuss specific differences in two schemes (and why they seem to be essential for the proof of security) after we preset our scheme.

3 Our Scheme

Our scheme is a probabilistic payment scheme. It involves probabilistic polynomial-time user, vendor and the bank. It is probabilistic in the same sense as [31] and [25]: User and Vendor are going to flip (appropriately biased) coin flips, so that with small probability (for example, with probability $\frac{1}{200}$ the user will have to pay a larger amount (for example, 1$ dollar charge) and the rest of the time it has a free access. Notice that the expected price per page in above example is thus half a cent.

Now, we need to define the properties needed from our coin-flipping protocol. Of course, one of the properties is efficiency (i.e. we should try to avoid costly digital signatures as much as possible.) Additionally, we need *fairness* and *authentication* properties.

Our coin-flip protocol mainly involves two probabilistic polynomially bounded players (i.e. algorithms) – a Vendor and a User (both polynomially bounded by a security parameter). We operate in the public-key setting, where both User and Vendor have public/private signature key pairs [12] (authenticated by the trusted third party, such as a Bank). The protocol proceeds in rounds. The rounds are divided into a pre-processing stage and polynomially-bounded subsequent on-line "coin-flip" rounds. After the initial pre-processing stage, if the Vendor and the User do not abort during this pre-processing stage, the sequence of future output "coin-flips" is uniquely defined. More specifically, every additional round reveals one (or several) coin-flips which was defined in the pre-processing stage (where a round consists of two messages one from User to Vendor and another from Vendor to User). Of course, within each round, one of the players (who already received a message of this round but did not yet send his message of this round) can efficiently compute the outcome of this round coin-flip before the other player. We are not trying to prevent this asymmetry, but rather we require that all the coin-flips associated with *future* rounds are pseudo-random for both players (for definitions of pseudo-randomness, see [3, 33].) More specifically, we say that the coin-flipping protocol is *fair* if the following three conditions are satisfied:

- If both players follow the protocol then there are no aborts.
- For all probabilistic polynomial-time Adversary-User algorithms, if the Vendor follows the protocol and does not abort in the pre-processing stage, then all the coin-flips are uniquely defined and for any non-aborting prefix of the protocol execution, the coin-flips of future rounds are pseudo-random for the Adversary-User.

– For all probabilistic polynomial-time Adversary-Vendor algorithms, if the User follows the protocol and does not abort in the pre-processing stage, then all the coin-flips are uniquely defined and for any non-aborting prefix of the protocol execution, the coin-flips of future rounds are pseudo-random for the Adversary-Vendor.

Additionally, we say that the coin-flipping protocol is Vendor-*authenticated* if it allows the Vendor to convince a third party what the outcome of the coin-flip is, given the transcript of the protocol execution (and an authenticated public key).

We satisfy these properties in the following protocol. First, the bank issues to the user certified public/private key pair for digital signatures. Then every time the user wishes to start making micro-payments to some Vendor, User and Vendor participate in the following two-stage coin-flipping process, assuming the existence of a one-way permutation f:

– SETUP
First, user and vendor run the following setup protocol:

s1. Vendor: The Vendor picks a random x and computes a *chain* of values (just as in coupon-based scheme) to produce a $y = f(f(f \ldots(x)))$. The Vendor sends y to the User.
After y is sent, the Vendor gives to the User a zero-knowledge proof of knowledge of x (using standard cut-and-choose methods – for definitions and further references see [5]).

s2. User: The User checks a zero-knowledge proof of knowledge of the Vendor (and if rejecting, aborts.) If the proof is accepting, then the User picks a random x' and computes a *chain* of values $y' = f(f(f \ldots(x')))$. The User signs and sends (y, y') (together with it's signature) to the Vendor.
After (y, y') is sent, the User gives to the Vendor a zero-knowledge proof of knowledge of x'.

s3. Vendor: The Vendor verifies user's proof of knowledge, user's signature and user's public key (if incorrect aborts.)

– COIN-FLIP
Now we are ready for the efficient coin-flip stage. Recall that both y and y' define roots of the two chains. To make the next coin-flip the user and the vendor execute the following protocol round:

c1. User: The User reveals to the Vendor its next pre-image in the y' chain.

c2. Vendor: the Vendor reveals to the user its next pre-image of the y chain. For both chains, one can associate hard-core bits with each pre-image (in fact up to logarithmically-many hard-core bits [13]) The xor of hard-core bits from y and y' chains define the coin-flip output for this round.

Before we proceed to describe the properties of the protocol, let us answer several frequently asked questions regarding our protocol:

REMARKS:

- One of the frequently asked questions regarding the design of the above protocol is: why is it necessary for both the user and the vendor to give zero-knowledge proofs of knowledge? (In fact, Rivest's scheme [25] omits the proofs of knowledge and does not use the hard-core bits [13].) The actual reason comes from a formal proof, but let us briefly mention the technical problem: in order to show that the scheme is fair for the User/Vendor, we must show that if the User/Vendor can predict future coin-flips (and, say, abort the protocol if the coin-flips are extremely unfavorable), then we can use such a predicting User/Vendor to invert a one-way function, thus reaching a contradiction. Now, the problem of using such an algorithm is that the prediction of the future coin-flips *does not* directly give us information regarding hard-core bits of the individual chains, but rather of the xor of two hard-core bits of both chains. Since one of this two hard-core bits does in fact belong to the predicting User/Vendor (and he does not need to disclose this hard-core bit) the prediction of the future coin-flips does not seem to help in predicting the corresponding hard-core bits in the other chain.
- Another problem of eliminating proofs of knowledge is that the user can set $y' = y$, which will certainly not make the future coin-flips pseudo-random. One can try to play with definitions, and say that only *revealed* coin-flips should be pseudo-random, but since the proofs of knowledge seem to be needed for the security proof anyway, we do not see any advantage of working with this less natural definition.
- One possible criticism of our scheme is that while the on-line stage is extremely efficient, a pre-processing stage is somewhat expensive. Indeed, zero-knowledge proofs of knowledge are the main source of inefficiency in our scheme. Yet, we do not know how to make the proof of security go through without such a pre-processing stage due to the reasons indicated above.
- We should also compare our coin-flipping protocol and the coupon-based schemes, such as PayWord [27]. The main advantage of our scheme is that the actual payments can be done very infrequently. Hence, in the (infrequent) case that the user has to pay, we can afford expensive on-line processing, including the on-line secure payment and receipts, thus eliminating the need for black-lists of credit-based approaches that were needed to prevent double-spending. Note that if the user refuses to pay the necessary amount the Vendor has a proof that the user has to pay which it can take to the bank/arbiter (since it has signed by the user (y, y') pair as well as the necessary inverses for both chains with the right properties.) Thus, we stress that in our scheme the (infrequent) payment can be done on-line, avoiding drawbacks of credit-based approaches and double-spending. We should point out that our proposal also differs from [25] in this regard, namely the scheme of [25] is proposed to be used as a credit-based scheme. In contrast, we suggest that in case of the Vendor-favorable coin-flip the payment will be made on-line, thus avoiding the drawbacks of credit-based approach. Since the payment is very infrequent we can in this case afford to do expensive on-line processing.

- There is another asymmetry in our protocol, namely that the user signs the value (y, y') while the vendor does not sign anything. The reason is that the Vendor in any event has a lot of control which information to provide to the user (and in fact can always provide bad, incomplete or incorrect information) and the way this is combatted in the business world is that the vendor gets bad publicity, loses customers, etc. (In particular, if the customer does not get the desired "free" information, it will simply stop interacting with the current Vendor.) We stress, though, that in our scheme, even a cheating vendor can not influence the outcome of the coin-flip and make the customer pay more "frequently".
- Notice that in the coin-flipping stage, the Vendor learns the value of the coin first. This is important, since otherwise, the user can stop the interaction if he discovers that he has to pay. Additionally, note that it is strait-forward to make biased coin-flips by combining several unbiased bits. Further, note that each iteration of the permutation can produce many (in fact, up to logarithmically many) hard-core bits [13] leading to further savings.
- Analogous to coupon-based PayTree scheme of Jutla and Yung [22], our scheme can be made more efficient by using tree-based construction for coin-flips.

We now list some of the properties of our protocol.

Claim 1 *If both players did not abort in the setup stage, then the coin-flips are uniquely defined.*

Proof: Since f is a one-way permutation, and y, y' are fixed, their pre-images and hard-core bits are uniquely defined. ∎

Claim 2 *Assume that one-way permutations exist and that the Vendor follows the protocol. Then, for any polynomially-bounded Adversary-User if the pre-processing stage is not aborted by the Vendor then for any prefix of the protocol execution the coin-flips of subsequent rounds are pseudo-random for the Adversary-User.*

Proof: Assume not. Then there exists probabilistic polynomial-time Adversary-User algorithm which after the (non-aborting) pre-processing stage and some prefix of the protocol execution can distinguish future coin-flips from a random sequence. The distinguishability implies that there is some the next-bit test that is not passed [3, 33]. Thus, there exists some future coin-flip which a probabilistic polynomial-time Adversary-User algorithm can predict with non-negligible probability. Now, we will show how this prediction can be used to invert a one-way permutation f on a random input z. Given a z for which we wish to find $f^{-1}(z)$, we put z in a random place in the Vendor's chain, and compute (by iterating f) the corresponding y, then run a zero-knowledge *simulator* to simulate the proof of knowledge of x (if the adversary can distinguish a simulation and the actual proof we reach a contradiction of zero-knowledge). Then, after the Adversary-User provides its y' and gives a zero-knowledge proof of knowledge of x' we use a *knowledge extractor* to get from the Adversary-User x'. Now, since we assume that the adversary can predict a coin of some future round with non-negligible

probability, and since we know x', we can now compute all hard-core bits associated with y' chain and predict with polynomial probability the hard-core bit of $f^{-1}(z)$. Using [13] this leads to inversion of f — a contradiction. ■

Now, the proof of the claim in the opposite direction mimics the previous proof:

Claim 3 *Assume that one-way permutations exist and that the User follows the protocol. Then, for any polynomially-bounded Adversary-Vendor if the preprocessing stage is not aborted by the User then for any prefix of the protocol execution the coin-flips of subsequent rounds are pseudo-random for the Adversary-Vendor.*

Proof: Similar to the previous claim, where we now use *knowledge extractor* for the Adversary-Vendor's proof of knowledge and *zero-knowledge simulator* for the User's proof of knowledge. ■

Thus, we have

Theorem 4 *In section 3, we presented* fair, authenticated *coin-flipping protocol.*

Finally it it worth-while to point out some of the the advantages of our scheme:

- Unlike the lottery solution [24], the bank does not have to participate in the coin-flip, and the number of bank-related transactions (which can be done using any other, more expensive payment scheme) is greatly reduced.
- Unlike the coin-flipping solution for [25], for which we do not know how to prove its security, our scheme is secure according to a strong pseudo-random definition for all the future rounds.
- After the setup stage, if the coin-flip is favorable to the Vendor, than it has the proof that the user must pay certain amount, which it can show to the bank/arbiter in case of dispute/nonpayment.
- If some site is used infrequently by some user, our coin-flip solution can be viewed as offering a "free-trial" service, where if tuned appropriately it can attract additional customers.
- Our solution can (and should) be combined with other more expensive schemes when the coin-flip is for payment, thus providing overall high security of the system.

4 Conclusions

The initial setup price involves checking one signature during connection to a new site and two proofs of knowledge, which can be done using standard cut and choose methods. After the setup stage, our solution is similar (up to a factor of two) in efficiency to the coupon-based schemes, but our scheme avoids "overspending" issue and the need to black-list users. Moreover,

- after the pre-processing stage is done, the subsequent number of rounds is minimized – our scheme does not add *any* additional rounds when there is no payment necessary (since we can "piggy-back" our coin-flip messages with standard "get-page/here-it-is" interaction), and bank it not involved at all, thus saving the overall round complexity between users, vendors and banks.
- we minimize the (amortized) number of total bits transmitted per transaction between users, vendors and the bank/broker, since most of the time the "for-free" coin-flip avoids expensive payment protocol, and the on-line coin-flip price is similar to coupon-based solutions.
- we minimize (amortized) computational demands needed per transaction for all the participants (i.e. both users and vendors as well as banks);
- we eliminate tamper-proof hardware requirements for all the participants (i.e. we do not need smart-cards) or other "per-transaction" lists or expensive hardware for pre-processing);
- we minimize fraud since this is not credit-based solution and the payment (if the coin-flip is favorable) must be made immediately, avoiding the problems of over-charging the accounts, having to wait for bank-sponsored lotteries, and risking non-payments.

Additionally, our scheme can be made anonymous, with the use of *pseudonyms*, similar to Chaum's scheme. We postpone this discussion to the full version of the paper.

A possible criticism of our scheme (as well as Wheeler's [31] and Rivest's [24, 25]) is that probabilistic payments is some weak form of *gambling* which is forbidden by U.S. laws. We do not address this issue here, but rather, say that in our view this may not constitute gambling since the expected profit of the Vendor is very close (by the law of large numbers) to a deterministic (but more expensive) schemes. Of course, the legal ramifications of the proposed scheme are beyond the scope of this paper.

Acknowledgements

The authors are grateful to William Aiello, Sanjoy Dasgupta, Stuart Haber, Ashwin Nyack, Ron Rivest and Victor Shoup for several valuable discussions regarding coin-flipping protocols. We also wish to thank an anonymous referee of the FC98 conference for some useful remarks.

References

1. R. Anderson, C. Manifavas, C. Sutherland "Netcard - a practical electronic cash system" In Fourth Cambridge Workshop on Security Protocols. Springer Verlag, Lecture Notes in Computer Science, April 1996.
   available online URL http://www.cl.cam.ac.uk/users/rja14/
2. Blum, M., "Coin Flipping over the Telephone," IEEE COMPCON 1982, pp. 133-137.
3. M. Blum, and S. Micali "How to Generate Cryptographically Strong Sequences Of Pseudo-Random Bits" *SIAM J. on Computing,* Vol 13, 1984, pp. 850-864, FOCS 82.

4. J.P. Boly, A. Bosselaers, R. Cramer, R. Michelsen, S. Mjølsnes, F. Muller, T. Pedersen, B. Pfitzmann, P. de Rooij, B. Schoenmakers, L. Vallée, and M. Waidner.
5. M. Bellare and O. Goldreich. "On defining proofs of knowledge." Extended abstract in Advances in Cryptology - Crypto 92 Proceedings, Lecture Notes in Computer Science Vol. 740, E. Brickell ed, Springer-Verlag, 1993.
6. B. Cox, D. Tygar, M. Sirbu "NetBill security and transaction proptocol" First USENIX Workshop on Electronic Commerce, New York, July 1995.
available online URL http://www.ini.cmu/NETBILL/home.html
7. L. Tang and S. Low "Chrg-http: A Tool for Micropayments on the World Wide Web" 6th USENIX Security Symposium, San Jose, CA July 1996.
8. D. Chaum "Achieving Electronic Privacy" Scientific American, pp. 96-101, August 1992.
9. D. Chaum, A. Fiat, M. Naor "Untracable electronic Cash" Crypto-89.
10. E. Gabber and A. Silberschatz "Agora: A Minimal Distributed Protocol for Electronic Commerce" USENIX Workshop on E-Commerce, Oakland CA Nov. 1996.
11. S. Glassman, M. Manasse, M. Abadai, P. Gauthier, and P. Sobalvarro "The milicent protocol for inexpensive electronic commerce" In Proc. of the forth International World Wide Web Conference", 1995.
available online URL http://www.research.digital.com/SRC/milicent
12. S. Goldwasser, S. Micali, and R. Rivest "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks". *SIAM Journal of Computing* vol 17, No 2, (April 1988), pp. 281-308.
13. O. Goldreich, L. Levin "A hard-core predicate for all one-way functions." In Porc. of 21st STOC, pp. 25-32 ACM 1989.
14. N. M. Haller "The S/KEY one-time password system. In ISOC 94.
15. R. Hauser, M. Steiner, M. Waidner "Micro-payments based on ikp" In 14th Worldwide Congress on Computer and Communication Security Protection, CNIT Paris -La defense France, June 1996.
available online URL http://www.zurich.ibm.com./Technology/Security/publications/1996/HSW96-new.ps.gz
16. S. Jarecki, A. Adlyzko "An efficient micropayment system based on probabilistic polling" Conference proceedings of *Financial Cryptography'97*, February 1997, Anguilla, BWI.
17. L. Lamport "Password authentication with insecure communication" Communications of the ACVM, 24(11):770-771, November 1981.
18. Mondex USA
available online URL http://www2.mondexusa.com/
19. G. Medvinsky, C. Neuman NetCash: A design for practical electronic currency on the internet. In *Proceeding sof the Second ACM Conference on Computer and Communcation Security* Novemeber 1994.
20. M. Naor "Bit Commitment using Pseudo-Ranomness" Proc. CRYPTO 89.
21. C. Neuman, G. Medvinsky Requirements for network payment: The Netcheque prospective. In Proc. of IEEE COMCON, March 95.
available online FTP ftp://prospero.isi/edu/pub/papers/security/
22. C. Jutla and M. Yung "Paytree: amortized signature for flexible micropayments" In Second USENIX workshop on Electronic Commerce, November 1996.
23. T. Pedersen "Electronic payments of small amounts" Technical Report DAIMI PB-495, Aarhus University, Computer Science Department, Århus, Denmark, August 1995.
24. R. Rivest "Lottery tickets as Micro-Cash" rump session talk at *Financial Cryptography'97*, February 1997, Anguilla, BWI.

25. R. Rivest "Electronic Lottery tickets as Micropayments" Proceedings of *Financial Cryptography'97*, LNCS series 1318, pp. 306-314.
26. R. Rivest "The MD5 message-digest algorithm" Internet Request for Comments, April 1992. RFC 1321.
available online URL http://theory.lcs.mit.edu/rivest/publications.html
27. R. Rivest and A. Shamir "Payword and micromint: Two simple micropayment schemes" In fourth Cambridge workshop on security protocols. Springer Verlag, lecture Notes in Computer Science, April 1996.
available online URL http://theory.lcs.mit.edu/rivest/publications.html
28. V. Shoup "On Fast and Provaaably Secure Message Authentication Based On Universal Hashing" CRYPTO-96.
29. J. Stern, S. Vaudenay "Small-Value-Payment: a Flexible Micropayment Scheme" Conference proceedings of *Financial Cryptography'97*, February 1997, Anguilla, BWI.
30. Visa and MasterCard "Secure Electronic Transactions (SET) specification, *available online* URL http://www.mastercard.com/set
31. D. Wheeler "Transactions using bets" in security protocols Int. Workshop, cambridge, UK April 1996. In LNCS 1189 pp. 89-92
available online URL http://www.cl.cam.ac.uk/users/cm213/Project/project_publ.html
32. Y. Yacobi "On the continuum between on-line and off-line e-cash systems - I" Conference proceedings of *Financial Cryptography'97*, February 1997, Anguilla,
33. A.C. Yao "Theory and Applications of Trapdoor Functions" *FOCS 82.*

This article was processed using the LaTeX macro package with LLNCS style