www.jatit.org

Towards Universally Acceptable m-Com Protocol

Yang Mu
School of Information Science and Technology, USTC
{yangchoun@yahoo.com.cn}

Abstract: Online trust is growing in importance. Consumers and businesses, feeling the pressure of economic downturn and terrorism, increasingly look to buy from and do business with the most trusted Web sites. Companies' perception of customer trust has steadily evolved from being a construct involving security and privacy issues to a multidimensional, complex construct that includes credibility, emotional comfort and quality. Further, trust online spans the end-to-end aspects of e-business rather than being just based on the electronic storefront. The increasing popularity of electronic commerce has necessitated the development of, robust, reliable, efficient and secure e-commerce protocols. These protocols should not only ensure the confidentiality and integrity of information exchanged, but researchers have identified other desirable properties, such as, money atomicity, goods atomicity and validated receipt, that must be satisfied by e-commerce protocols. In the traditional scenario of physical goods, one can order the goods and transfer money over the network, but the goods cannot be delivered over the network. Whereas Informational goods have the special characteristic that both the delivery of goods and money can accomplished over the same network infrastructure. This paper summarizes analyses and compares five of the most emerging ecommerce protocols for micro transactions. These protocols are NetBill, Payword, Micromint, Millicent and MPTP. Analysis of these protocols is done on the basis of eight crucial parameters which are Computational Cost, Communication Cost, Storage Cost, Privacy, Reliability, Repudiation, online/offline scheme and trust.

Keywords: NetBill, MPTP, Micromint, Payword, Millicent, e-commerce protocols, privacy, Reliability, Repudiation, Online/Offline scheme, trust.

1.INTRODUCTION

Ever since it was realized that a commodity has worth, money was invented as an abstract way of representing value, and systems for making payments have been in place. With advances in human civilizations, new and increasingly abstract representations of value were introduced. A corresponding progression of value transfer systems, starting from barter, through bank notes, payment orders, checks, and later credit cards, has finally culminated in "electronic" systems. Mapping between these abstract payments and the transfer of "real value" is still guaranteed by banks through financial clearing systems. The financial clearing systems are built on the closed, strictly controlled networks of financial institutions and hence considered comparatively more secure than open networks.

Since the world has become a global village and access to any Remote location is at our finger tips, now it is the right time for distributed projects for Organizations not necessarily distributed i.e. organizations whose business is running in generic or specialized locations all over the world i.e. the era of global business. Due to this fact there has been a great deal of interest in facilitating commercial transactions over open computer networks, such as the Internet. This is the only means on hand by which large-scale robust transactions can be attended to in a business and customer friendly heterogeneous milieu. More than a few electronic payment systems have been proposed and implemented in the past few years. Currently, many poles-apart, incompatible Internet payment systems compete with one another. Most shops accept, at best, only a subset of them.

There is a long history of attempts to construct systems for transferring information quickly over long distances. But from a protocol designer and analyzer's point of view, the mishaps that can be caused by misinterpreted communications are fascinating.

In this paper we discuss a set of five of the most promisingly emerging e-commerce protocols namely NetBill, Payword, Micromint, Millicent and MPTP briefly described in section 2, 3, 4, 5 and 6, and analyzed on the basis of a set of eight crucial parameters comprising of Computational Cost, Communication Cost, Storage Cost, Privacy, Reliability, Repudiation, online/offline scheme.

2. NETBILL

The NetBill transaction model involves three parties: the customer, the merchant and the NetBill transaction server. Furthermore it involves three phases: price negotiation, goods delivery, and payment. For information goods which can be delivered over the network, the NetBill protocol merges goods delivery and payment into a single atomic transaction.

In this atomic transaction, the customer and merchant interaction occurs in first two phases; the NetBill server is involved only in the payment phase, when the merchant submits a transaction request. The customer directly contacts the NetBill server only in case of communication failure or administrative functions request.

A) Transaction Objectives

Following set of objectives in a NetBill transaction.

1) Only authorized customers can charge against a NetBill account.
2) The customer and merchant must agree on the item to be purchased and the price to be charged.
3) A customer can optionally protect his/her identity from merchants.

4) Customers and merchants are provided with proof of transaction results from NetBill.
5) There is an offer and acceptance negotiation phase between customer and merchant.
6) A customer may present credentials identifying his/her as entitled to special pricing or treatment.
7) A customer receives the information goods she purchases if and only if she is charged (and thus the merchant is paid) for the goods.
8) A customer may need approval from a fourth (access control) party before the NetBill server will allow a transaction.
9) The privacy and integrity of communications is protected from observation or alteration by external parties.

NetBill accounts can either be a debit model (pre-paid) or credit model (post-paid). In case of prepaid model, funds would be transferred to NetBill in advance to cover future purchases. If the user does not have sufficient funds to cover a particular transaction, that transaction would be declined.

In the credit model, transactions would be accumulated with payment to NetBill being triggered by either time (based on a pre-established billing period) or dollar amount (based on a pre-established limit). Because granting credit creates a risk of non-payment, higher transaction fees may be associated with credit, versus prepaid accounts.

B) A NetBill Scenario

Figure 1 shows NetBill in action. A user, represented by a client computer, desires to buy from a merchant's server. A NetBill server maintains accounts for both customers and merchants. These accounts are linked to conventional financial institutions. A NetBill transaction transfers the information goods from merchant to user, and debits the customer's account and credits the merchant's account for the value of the goods. When necessary, funds in a customer's NetBill account can be replenished from a bank or credit card; similarly finances in a merchant's NetBill account are made available by depositing them in the merchant's bank account.

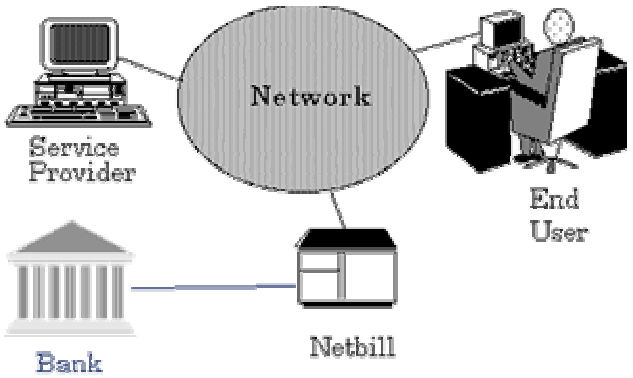

Figure 1: The Netbill Concept

C) NetBill Design

There are a number of operational challenges in making of an electronic commerce systems technically and economically feasible:

I) High transaction volumes at low cost. In this scheme information is sold for a few pennies a page, and an electronic commerce system must handle very large transaction volumes at a marginal cost of a penny or less per transaction.

II) Authentication, privacy and security. In today's infrastructure Internet today provides no universally accepted means for authenticating users, protecting privacy, or providing security.

III) Account management and administration. Customers and merchants must be able to establish and monitor their accounts.

D) NetBill Architecture

NetBill offers a single protocol that implements charging in a wide range of service interactions. NetBill provides transaction support through libraries integrated with different client-server pairs. These libraries use a single transaction-oriented protocol for communication between client, server and NetBill; the normal communications model between client and server is unchanged. Clients and servers can continue to communicate using protocols optimized for the application -- for example, video delivery or database queries -- while the financial-related information is transmitted over protocols optimized for that purpose. This approach allows NetBill to work with information delivery mechanisms ranging from the WWW to FTP and MPEG-2 streams.

The client library -- which is called the *checkbook* -- and the server library -- the *till* -- have a well-defined API allowing easy assimilation with a range of applications. (Underneath we describe how these libraries are integrated with Mosaic clients and HTTP servers.) The libraries incorporate all security and payment protocols, relieving the client/server application developer from having to worry about these issues. All network communications between the checkbook and till are encrypted to protect against adversaries who eavesdrop or inject messages.

E) The NetBill Transaction Protocol

Before a customer begins a typical NetBill transaction, she will usually contact a server to locate information or a service of interest. For example, the customer may request a Table of Contents of a journal showing available articles available, and a list price associated with each article. The transaction begins when the customer requests a formal price quote for a product. This price may be different than the standard list price because, for example, the customer may be part of a site license group, and thus be entitled to a marginal price of zero.

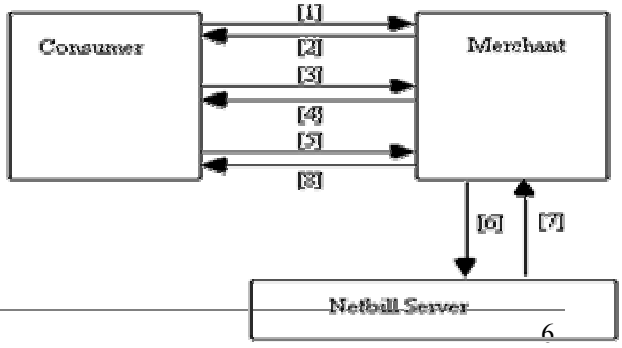

Figure 2: Transaction Protocol

The customer's client application then indicates to the checkbook library that it would like a price quote from a particular merchant for a specified product. The checkbook library sends an authenticated request for a quote to the till library which forwards it to the merchant's application. (Figure 2, Step 1.)

The merchant then must invoke an algorithm to determine a price for the authenticated user [5]. He returns the digitally signed price quote through the till, to the checkbook (Step 2), and on to the customer's application. The customer's application then must make a purchase decision.

Assume the customer's application accepts the price quote. The checkbook then sends (Step 3) a digitally signed purchase request to the merchant's till. The till then requests the information goods from the merchant's application and sends them to the customer's checkbook encrypted in a one-time key (Step 4), and computes a cryptographic checksum (such as MD5) on the encrypted message. As the checkbook receives the bits, it writes them to stable storage. When the transfer is complete, the checkbook computes its own cryptographic checksum on the encrypted goods and returns to the till a digitally signed message specifying the product identifier, the accepted price, the cryptographic checksum, and a timeout stamp: we refer to this information as the *electronic payment order (EPO)* (Step 5). Note that, at this point, the customer can not decrypt the goods; neither has the customer been charged.

Upon receipt of the EPO, the till checks its checksum against the one computed by the checkbook. If they do not match, then the goods can either be retransmitted, or the transaction aborted at this point. This step provides very high assurance that the encrypted goods were received without error. If checksums match, the merchant's application creates a digitally signed invoice consisting of price quote, checksum, and the decryption key for the goods. The application sends both the EPO and the invoice to the NetBill server (Step 6).

The NetBill server verifies that the product identifiers, prices and checksums are all in agreement. If the customer has the necessary funds or credit in his/her account, the NetBill server debits the customer's account and credits the merchant's account, logs the transaction, and saves a copy of the decryption key. The NetBill server then returns to the merchant a digitally signed message containing an approval, or an error code indicating why the transaction failed (Step 7). The merchant's application forwards the NetBill server's reply and (if appropriate) the decryption key to the checkbook (Step 8).

F) Protocol Failure Analysis

The NetBill server is highly reliable and highly available. All transactions at the NetBill server are atomic: they either finish completely or not at all. NetBill is never in doubt about the status of a purchase.

First, consider the protocol from the perspective of the customer's application. Up to step 5, when the customer application acknowledges receipt of the information goods, the customer application knows that no transaction has occurred. That is, the customer does not have access to the product and the merchant does not have the customer's money. Once the application sends the EPO, the customer is *committed* to the transaction and must be prepared to accept the purchase. If the customer's application does not receive a response from the merchant's application, then it is the responsibility of the customer's application to determine what happened: the customer's application can poll either the merchant application or the NetBill server to determine the status of the purchase request. If the merchant's application did not successfully forward the EPO to the NetBill server, then the EPO will have expired and the NetBill server will respond to the customer's application that the purchase has failed. Of course, the customer still does not have the one time key, so while the customer still has his/her money, she also does not have the goods. If, on the other hand, the transaction succeeded before communication failed, then the customer's application can find the status of the purchase and, if appropriate, the decryption key from either the merchant's application or the NetBill server (which has registered the key). If both are unreachable, the customer's application must continue to poll.

Now consider the protocol from the perspective of the merchant's application. Before it forwards the EPO and invoice to the NetBill server, the merchant's application knows that the transaction has not occurred. After it forwards the EPO and invoice, however, the merchant's application is *committed* to the transaction and must obtain the result from the NetBill. If the merchant's application does not receive a response from the NetBill server, the merchant's application must poll the NetBill server.

The protocol is much simpler for the NetBill server than for the other parties. The NetBill server is never in a state in which it depends on a response from another entity to determine the status of a transaction. Until the NetBill server receives the EPO and invoice from the merchant's application, it knows nothing about the purchase. Once it receives the EPO and invoice it has all the information necessary to approve or reject the purchase. The NetBill transaction protocol also exhibits a number of other desirable features:

I) Support for flexible pricing. By including the steps of offer and acceptance, an opportunity for the merchant is provided to calculate a customized quote for an individual customer. In the process we also generate signed messages that can later prove that there was a contract at the quoted price.

II) Scalability. The bottleneck in the NetBill model is the NetBill server which supports many different merchants. Our transaction protocol minimizes the load on the NetBill server and distributes the burden over the many customer and merchant machines. Note that a single interaction with the NetBill server both verifies the availability of funds and records the transaction. It is not possible to have less than one interaction with the NetBill server .

III) Protection of user accounts against unscrupulous merchants. In a conventional credit card transaction, the merchant learns the customer's credit card number and can submit fraudulent invoices in the customer's name. In a NetBill transaction, the customer

digitally signs the EPO using a key that is never revealed to the merchant, thus eliminating this threat.

G) NetBill Account Management

NetBill supports a many-to-many relationship between *customers* and *account*s. A project account at a corporation can have many users authorized to charge against it. Conversely, an individual customer can maintain multiple personal accounts. Every account has a single user who is the account *owner*; and the account owner can grant various forms of access rights on the account to other users. User account administration is provided through WWW forms.

Automating account establishment for both customers and merchants is important for limiting costs. (Account creation is one of the largest costs associated with traditional credit card and bank accounts.) To begin the process, a customer retrieves, perhaps by anonymous FTP, a digitally signed NetBill security module that will work with the user's WWW browser. Once the customer checks the validity of the security module, she puts the module in place. She then fills out a WWW form, including appropriate credit card or bank account information to fund the account, and submits it for processing. The security module encrypts this information to protect it from being observed in transit. The NetBill server must verify that this credit card or banking account number is valid and that the user has the right to access it. There are a variety of techniques for this verification: for example, customers may telephone an automated attendant system and provide a PIN associated with the credit card or bank account to obtain a password.

3. PayWord

PayWord is credit-based protocol. In this scheme the players are brokers, users, and vendors. Brokers authorize users to make micropayments to vendors, and redeem the payments collected by the vendors. While user-vendor relationships are transient, broker-user and broker-vendor relationships are long-term. The user establishes an account with a broker, who issues his/her a digitally-signed PayWord Certificate containing the broker's name, the user's name and IP-address, the user's public key, the expiration date, and other information. The certificate has to be renewed by the broker (e.g. monthly), who will do so if the user's account is in good standing. This certificate authorizes the user to make Payword chains, and assures vendors that the user's PayWords are redeemable by the broker.

The public keys of the broker B, user U, and vendor V are denoted PK_B, PK_U, and PK_V , respectively; their secret keys are denoted SK_B, SK_U, and SK_V . A message M with its digital signature produced by secret key SK is denoted $\{ M \}_{SK}$. This signature can be verified using the corresponding public key PK.

When user U clicks on a link to a vendor V 's non-free web page, his browser determines whether this is the first request to V that day. For a first request, U computes and signs a “commitment" to a new user-specific and vendor-specific chain of paywords w_1, w_2, . . . , w_n. The user creates the payword chain in reverse order by picking the last payword w_n at random, and then computing

$$w_i = h (w_{i+1})$$

for i = n - 1, n - 2, . . . , 0. Here w_0 is the root of the payword chain, and is not a payword itself. The commitment contains the root w_0, but not any payword w_i for i > 0. Then U provides this commitment and his/her certificate to V , who verifies their signatures. The i-th payment (for i = 1, 2, . . .) from U to V consists of the pair (w_i , i), which the vendor can verify using w_{i-1}. Each such payment requires no calculations by U, and only a single hash operation by V .At the end of each day, V reports to B the last (highest-indexed) payment (w_l, l) received from each user that day, together with each corresponding commitment. B charges U's account l cents and pays l cents into V 's account.

PayWord is an “offline" scheme: V does not need to interact with B when U first contacts V, nor does V need to interact with B as each payment is made. Note that B does not even receive every PayWord spent, but only the last PayWord spent by each user each day at each vendor. The public-key operations required by V are only signature verifications, which are relatively efficient.

A) User-Broker relationship and certificates

User U begins a relationship with broker B by requesting an account and a PayWord Certificate. She gives B over a secure authenticated channel, his/her credit-card number, his/her public key PK_U, and his/her delivery address A_U. His/her aggregated PayWord charges will be charged to his/her credit-card account. His/her delivery address is his/her Internet/email or his/her U.S. mail address; his/her certificate will only authorize payments by U for purchases to be delivered to A_U.

The user's certificate has an expiration date E. Certificates might expire monthly. Users who don't pay their bills won't be issued new certificates. The broker may also give other (possibly user-specific) information I_U in the certificate,such as: a certificate serial number, credit limits to be applied per vendor, information on how to contact the broker, broker/vendor terms and conditions, etc. The user's certificate C_U thus has the form:

$$C_U = \{ B , U , A_U , PK_U , E , I_U \} SK_B$$

The PayWord certificate is a statement by B to any vendor that B will redeem authentic PayWord produced by U turned in before the given expiration date

PayWord is not intended to provide user anonymity. Although certificates could contain user account numbers instead of user names, the inclusion of A_U effectively destroys U's anonymity. However, some privacy is provided, since there is no record kept as to which documents were purchased. If U loses his/her secret key she should report it at once to B. His/her liability should be limited in such cases, as it is for credit-card loss. However, if she does so repeatedly the broker may refuse his/her further service. The broker may also keep a “hot list" of certificates whose users have reported lost keys, or which are otherwise problematic.

B) User-Vendor relationships and payments

User Vendor relationships are transient. A user may visit a web site, purchase ten pages, and then move on elsewhere.

C) Commitments

When user U is about to contact a new vendor V, she computes a fresh PayWord chain w_1,. . . , w_n with root w_0. Here n is chosen at the user's convenience; it could be ten or ten thousand. She then computes his/her commitment for that chain:

$$M = \{ V , C_U , w_0 , D , I_M \} SK_U$$

Here V identifies the vendor, C_U is U's certificate, w_0 is the root of the payword chain, D is the current date, and I_M is any additional information that may be desired (such as the length n of the payword chain). M is signed by U and given to V.

This commitment authorizes B to pay V for any of the paywords $w_1, \ldots, w_n$ that V redeems with B before date D (plus a day's grace). Note that paywords are vendor-specific and user-specific, they are of no value to another vendor. The vendor V should cache verified commitments until they expire at the end of the day. Otherwise, if he redeemed and forgot paywords received before the expiration date of the commitment, U could cheat V by replaying earlier commitments and PayWords.

D) Payments

A payment P from U to V consists of a payword and its index, P = (w_i , i). The payment is short, only twenty or thirty bytes long. The payment is not signed by U, since it is self-authenticating. The user spends his/her paywords in order: w_1 first, then w_2, and so on. If each payword is worth one cent, and each web page costs one cent, then she discloses w_i to V when she orders his/her i-th web page from V that day. This leads to the PayWord payment policy for each commitment a vendor V is paid l cents, where (w_l, l) is the corresponding payment received with the largest index. This means that V needs to store only one payment from each user, the one with the highest index. Once a user spends w_i , she can not spend w j for j < i. The broker can confirm the value to be paid for w_l by determining how many applications of h are required to map w_l into w_0.

PayWord supports variable-size payments in a simple and natural manner. If U skips paywords, and gives w_7 after giving w_2, she is giving V a nickel instead of a penny. When U skips paywords, during verification V need only apply h a number of times proportional to the value of the payment made.

A payment does not specify what item it is payment for. The vendor may cheat U by sending him nothing, or the wrong item, in return. The user bears the risk of losing the payment, just as if he had put a penny in the mail. Vendors who so cheat their customers will be shunned. This risk can be moved to V , if V specifies payment after the document has been delivered. If U doesn't pay, V can notify B and/or refuse U further service. For micropayments, users and vendors might find either approach workable.

E) Vendor-Broker relationships and redemption

A vendor V need not have a prior relationship with B, but does need to obtain PK_B in an authenticated manner, so that he can authenticate certificates signed by B. He also needs to establish a way for B to pay V for paywords redeemed. (Brokers pay vendors by means outside the PayWord system.)

At the end of each day (or other suitable period), V sends B a redemption message giving, for each of B's users who have paid V that day

- The commitment C_U received from U.
- The last payment P = (w_l , l) received from U.

The broker then needs to

- Verify each commitment received (he only needs to verify user signatures, since he can recognize his own certificates), including checking of dates, etc.
- Verify each payment (w_l , l) (this requires l hash function applications). We assume that B normally honors all valid redemption requests.

F) Efficiency

Following are the PayWord's computational and storage requirements

- The broker needs to sign each user certificate, verify each user commitment, and perform one hash function application per payment. The broker stores copies of user certificates and maintains accounts for users and vendors.
- The user needs to verify his certificates, sign each of his commitments, and perform one hash function application per PayWord committed to. He needs to store his secret key SK_U, his active commitments, the corresponding payword chains, and his current position in each chain.
- The vendor verifies all certificates and commitments received, and performs one hash function application per payword received or skipped over. (All his computations are on-line.) The vendor needs to store all commitments and the last payment received per commitment each day.

4. MicroMint

MicroMint is designed to provide reasonable security at very low cost, and is optimized for unrelated low-value payments. MicroMint uses no public-key operations at all. MicroMint "coins" are produced by a broker, who sells them to users. Users give these coins to vendors as payments. Vendors return coins to the broker in return for payment by other means.

A coin is a bit-string whose validity can be easily checked by anyone, but which is hard to produce. MicroMint has the property, that generating many coins is very much cheaper, per coin generated, than generating few coins. A large initial investment is required to generate the first coin, but then generating additional coins can be made progressively cheaper.

The broker will typically issue new coins at the beginning of each month, the validity of these coins will expire at the end of the month. Unused coins are returned to the broker at the end of each month, and new coins can be purchased at the beginning of each month. Vendors can return the coins they collect to the broker at their convenience

A) Hash Function Collisions

MicroMint coins are represented by hash function collisions, for some specified one-way hash function h mapping m-bit strings x to n-bit strings y. We say that x is a pre-image of y if h(x) = y. A pair of distinct m-bit strings (x1 , x2) is called a (2-way) collision if h(x1) = h(x2) = y, for some n-bit string y.

If h acts "randomly," the only way to produce even one acceptable 2-way collision is to hash about $2^{n/2}$ x-values and search for repeated outputs.

Hashing c times as many x-values as are needed to produce the first collision results in approximately c2 as many collisions, for $1 \leq c \leq 2^{n/2}$, so producing collisions can be done increasingly efficiently, per coin generated, once the threshold for finding collisions has been passed.

B) Coins as k-way collisions

A k-way collision is a set of k distinct x-values $x_1, x_2, \ldots, x_k$ that have the same hash value y. The number of x-values that must be examined before one expects to see the first k-way collision is then approximately $2^{n(k-1)/k}$. If one examines c times this many x-values, for $1 \leq c \leq 2^{n/k}$, one expects to see about c^k k-way collisions. Choosing $k > 2$ has the dual effect of delaying the threshold where the first collision is seen, and also accelerating the rate of collision generation, once the threshold is passed.

We thus let a k-way collision $(x_1, \ldots, x_k)$ represent a coin. The validity of this coin can be easily verified by anyone by checking that the x_i's are distinct and that

$$h(x_1) = h(x_2) = \ldots = h(x_k) = y$$

for some n-string y.

C) Minting coins

The process of computing h(x) = y is analogous to tossing a ball (x) at random into one of 2n bins; the bin that ball x ends up in is the one with index y. A coin is thus a set of k balls that have been tossed into the same bin. Getting k balls into the same bin requires tossing a substantial number of balls altogether, since balls can not be "aimed" at a particular bin. To mint coins, the broker will create 2n bins, toss approximately k^{2n} balls, and create one coin from each bin that now contains at least k balls. With this choice of parameters each ball has a chance of roughly 1/2 of being part of a coin.

Whenever one of the 2n bins has k or more balls in it, k of those balls can be extracted to form a coin. Note that if a bin has more than k balls in it, the broker can in principle extract k-subsets in multiple ways to produce several coins. However, an adversary who obtains two different coins from the same bin could combine them to produce multiple new coins. Therefore, it is recommended that a MicroMint broker should produce at most one coin from each bin.

D) A detailed scenario

The broker will invest in substantial hardware that gives him a computational advantage over would-be forgers, and run this hardware continuously for a month to compute coins valid for the next month. This hardware is likely to include many special-purpose chips for computing h efficiently.

E) Selling coins

Towards the end of each month, the broker begins selling coins to users for the next month. At the beginning of each month, B reveals the new validity criterion for coins to be used that month. Such sales can either be on a debit basis or a credit basis, since B will be able to recognize coins when they are returned to him for redemption. In a typical purchase, a user might purchase $25.00 worth of coins (2500 coins), and charge the purchase to his credit card. The broker keeps a record of which coins each user bought. Unused coins are returned to the broker at the end of each month.

F) Making payments

Each time a user purchases a web page, he gives the vendor a previously unspent coin $(x_1, x_2, \ldots x_k)$. The vendor verifies that it is indeed a good k-way collision by computing $h(x_i)$ for $1 \leq i \leq k$, and checking that the values are equal and good. Note that while the broker's minting process was intentionally slowed down by a factor of 2t, the vendor's task of verifying a coin remains extremely efficient, requiring only k hash computations and a few comparisons.

G) Redemptions

The vendor returns the coins he has collected to the broker at the end of each day. The broker checks each coin to see if it has been previously returned, and if not, pays the vendor one cent (minus his brokerage fee) for each coin. It is proposed that if the broker receives a specific coin more than once, he does not pay more than once. Which vendor gets paid can be decided arbitrarily or randomly by the broker. This may penalize vendors, but eliminates any financial motivation a vendor might have had to cheat by redistributing coins he has collected to other vendors.

H) Security Properties

Security mechanisms are primarily designed to discourage large-scale attacks, such as massive forgery or persistent double-spending.

I) Forgery

Large-scale forgery can be detected and countered as follows:

1). All forged coins automatically become invalid at the end of the month.

2). Forged coins can not be generated until after the broker announces the new monthly coin validity criterion at the beginning of the month.

3). The use of hidden predicates (described below) gives a finer time resolution for rejecting forged coins without affecting the validity of legal coins already in circulation.

4). The broker can detect the presence of a forger by noting when he receives coins corresponding to bins that he did not produce coins from. This works well in our scenario since only about half of the bins produce coins. To implement this, the broker need only work with a bit-array having one bit per bin.

5). The broker can at any time declare the current period to be over, recall all coins for the current period, and issue new coins using a new validation procedure.

6). The broker can simultaneously generate coins for several future months in a longer computation; this makes it harder for a forger to catch up with the broker.

J) Theft of coins

If theft of coins is judged to be a problem during initial distribution to users or during redemption by vendors, it is easy to

transmit coins in encrypted form during these operations. User/broker and vendor/broker relationships are relatively stable, and long-term encryption keys can be arranged between them. To protect coins as they are being transferred over the Internet from user to vendor, one can of course use public-key techniques to provide secure communication. However, we can also make the coins user specific.

K) Double-spending

Since the MicroMint scheme is not anonymous, the broker can detect a doubly-spent coin, and can identify which vendors he received the two instances from. He also knows which user the coin was issued to. With the vendors' honest cooperation, he can also identify which users spent each instance of that coin. Based on all this information, the broker can keep track of how many doubly-spent coins are asssociated with each user and vendor. A large-scale cheater (either user or vendor) can be identified by the large number of duplicate coins associated with his purchases or redemptions; the broker can then drop a large-scale cheater from the system.

5.MILLICENT

The key features of Millicent are its use of *brokers* and of *scrip*. Brokers take care of account management, billing, connection maintenance, and establishing accounts with vendors. Scrip is digital cash that is only valid for a specific vendor. The vendor locally validates the scrip to prevent customer fraud, such as double spending.

In Millicent Brokers serve as accounting intermediaries between customers and vendors. Customers enter into long-term relationships with brokers, in much the same way as they would enter into an agreement with a bank, credit card company, or Internet service provider. Brokers acquire and sell vendor scrip as a service to customers and vendors. Broker scrip serves as a common currency for customers to use when buying vendor scrip, and for vendors to give as a refund for unspent scrip.

A) Security and Trust

In Millicent, it is imagined that people treat scrip as they would treat change in their pocket. Since people don't need a receipt when buying candy from a vending machine, they don't need a receipt when buying an item using scrip. If they don't get what they paid for, they complain and get a refund. It is expected that users have a few dollars of scrip at a time. It is not expected that they have hundreds, or even tens, of dollars of scrip. As a result, scrip is not worth stealing unless you can steal lots of it; and if you steal lots, you will get caught.

B) Trust model

Millicent assumes asymmetric trust relationships among the three entities - customers, brokers, and vendors. Brokers are assumed to be the most trustworthy, then vendors, and, finally, customers. The only time customers need to be trusted is when they complain about service problems.

Three factors make broker fraud unprofitable. First, customer and vendor software can independently check the scrip and maintain account balances, so any fraud by the broker can be detected. Second, customers do not hold much scrip at any one time, so a broker would have to commit *many* fraudulent transactions to make much of a gain and this makes them likelier to be caught. Finally, the reputation of a broker is important for attracting customers and a broker would quickly lose its reputation if customers have troubles with the broker. The repeat business of active customers is more valuable to a broker than the scrip that it could steal.

Vendor fraud consists of not providing goods for valid scrip. If this happens, customers will complain to their broker, and brokers will drop vendors who cause too many complaints.

As a result, the Millicent protocol is skewed to prevent customer fraud (forgery and double spending) while providing indirect detection of broker and vendor fraud.

C) Security

The security of Millicent transactions comes from several aspects.

1) All transactions are protected
2) Every Millicent transaction requires that the customer knows the secret associated with the scrip. The protocol never sends the secret in the clear, so there is no risk due to eavesdropping. No piece of scrip can be reused, so a replay attack will fail. Each request is signed with the secret, so there is no way to intercept scrip and use the scrip to make a different request.
3) Inexpensive transactions limit the value of fraud
4) Inexpensive transactions can rely on inexpensive security: it's not worth using expensive computer resources to steal inexpensive scrip. In addition, it would take many illegal uses of scrip to acquire much money, and that raises the probability of getting caught.
5) Fraud is detectable and eventually traceable
6) Fraud is detected when the customer doesn't obtain the desired goods from the vendor, or when the balance returned to the customer doesn't match the balance due. If the customer is cheating, then the vendor's only loss is the cost of detecting the bad scrip and denying service. If the vendor is cheating, the customer will report a problem to the broker. notices a When a broker pattern of complaints from many customers against a vendor, it can pinpoint the fraud and cut off all dealings with the vendor. If a broker is cheating, the vendor will notice bad scrip coming from many customers, all originating from a single broker. The vendor can then publicize its complaint in an appropriate venue.

D) Scrip

The main properties of scrip are:

1) It has value at a specific vendor.
2) It can be spent only once.
3) It is tamper resistant and hard to counterfeit.
4) It can be spent only by its rightful owner.
5) It can be efficiently produced and validated.

Following are the basic techniques to achieve these properties are outlined here:

1) The text of the scrip gives its value and identifies the vendor.
2) The scrip has a serial number to prevent double spending.
3) There is a digital signature to prevent tampering and counterfeiting.
4) The customer signs each use of scrip with a secret that is associated with the scrip.
5) The signatures can be efficiently created and checked using a fast one-way hash function (like MD5 or SHA).

E) Scrip structure

There are three secrets involved in producing, validating, and spending scrip. The customer is sent one secret, the customer_secret, to prove ownership of the scrip. The vendor uses one secret, the master_customer_secret, to derive the customer_secret from customer information in the scrip. The third secret, the master_scrip_secret, is used by the vendor to prevent tampering and counterfeiting.

The secrets are all used in a way that shows knowledge of the secret without revealing the secret. To attest to a message, the secret is appended to the message, and the result is hashed to produce a signature. The message (without the secret) and the signature prove - due to the one-way nature of the hash function - knowledge of the secret, because the correct signature can only be derived if you know the secret.

Scrip has the following fields :

1) Vendor identifies the vendor for the scrip.
2) Value gives the value of the scrip.
3) ID# is the unique identifier of the scrip. Some portion of it is used to select the master_scrip_secret used for the certificate.
4) Cust_ID# is used to produce the customer secret. A portion of Cust_ID# is used to select the master_customer_secret which is also used in producing the customer secret.
5) `Expires` is the expiration time for the scrip.
6) Props are extra data describing customer properties (age, state of residence, etc.) to the vendor.
7) Certificate is the signature of the scrip.

Scrip is validated in two steps. First, the certificate is recomputed and checked against the certificate sent with the scrip. If the scrip has been tampered with, then the two certificates will not match. Second, there is a unique identifier (ID#) included in the scrip body and the vendor can check for double spending by seeing if it has recorded that identifier as already spent. Generating and validating scrip each require a little text manipulation and one hash operation. Unless the secret is known, scrip cannot be counterfeited or altered.

F) Brokers

Brokers maintain the accounts of customers and vendors, and they handle all real-money transactions. The customer establishes an account with a broker by using some other method (like a credit card, or a higher-security electronic commerce system) to purchase some broker scrip. The customer then uses the broker scrip to buy vendor scrip.

The vendor and the broker have a long-term business relationship. The broker sells vendor scrip to customers and pays the vendor.

When a customer wants to make a purchase, the customer contacts the broker to obtain the necessary vendor scrip. The customer uses his broker scrip to pay for the vendor scrip using the Millicent protocol. The broker returns the new vendor scrip along with change in broker scrip.

There are three ways in which the broker gets the vendor scrip. The "scrip warehouse" model assumes a casual relationship between the broker and vendor. The "licensed scrip producer" model assumes a substantial and long-lasting relationship between the broker and vendor. The "multiple broker" model assumes a relationship between brokers, but requires no relationship between the vendor and broker.

G) Scrip warehouse

1) When the broker is acting as a scrip warehouse, the broker buys multiple pieces of scrip from a vendor. The broker stores the scrip and sells the pieces one at a time to customers

2) The broker uses the Millicent protocol to buy the scrip from the vendor in the same way a customer would. Selling scrip in large blocks is more efficient for the vendor since the communication and financial transaction costs are amortized over all the pieces of scrip.

H) Licensed scrip production

If a broker's customers buy a lot of scrip for a specific vendor, it may be desirable for a vendor to "license" the broker to produce vendor scrip. This means that the broker generates scrip that the vendor can validate and accept. The vendor sells the the broker the right to generate scrip using a given master_scrip_secret, series of scrip ID#'s, master_customer_secret, and series of customer identifiers. The vendor can validate the licensed scrip because the master_scrip_secret is known from the series of the scrip ID# and the master_customer_secret is known from the series of the customer identifier.

A license covers a specific series (unique range of identifiers - ID#'s) of scrip for a given period of time, and the secrets shared between the broker and vendor only apply to that series. A vendor can issue licenses to different brokers by giving out different series and secrets to each one. Of course, a vendor can produce its own scrip using its own private series and secrets.

There is less communication because the license is smaller to transmit than a few pieces of scrip. The vendor does less computation since it does not have to generate the scrip itself. The broker does not have to store large blocks of scrip, since it can generate the scrip on demand. Additionally, it allows the broker to encode specific user properties into each piece of scrip it generates.

I) Multiple brokers

In an environment where there are multiple brokers, a customer of one broker may want to make a purchase from a vendor associated with another broker. If the vendor only wants to have an account with its own broker (perhaps to simplify accounting), the customer will have to go through the vendor's broker to buy vendor scrip.

The entire transaction will go like this:

1) The customer asks his broker for vendor scrip.
2) The customer's broker tries to set up account with the vendor.
3) The vendor tells the customer's broker his broker's name.
4) The customer's broker buys broker scrip from the vendor's broker.
5) The customer's broker returns the vendor's broker's scrip to the customer.
6) The customer buys vendor scrip from the vendor's broker.
7) The customer uses the vendor scrip at the vendor.

J) Customer, Broker, and Vendor Interactions

The following diagrams (Figures a-e) present the steps for a complete Millicent session (including the broker buying scrip from the vendor). The initial step (Figure a) happens only once per session. The second step (Figure b) happens each time the customer has no stored scrip for a vendor. Step three (Figure c) happens only if the broker must contact the vendor to buy the scrip. It is not needed for licensed scrip production. The fourth step (Figure d) shows the broker returning the vendor scrip to the customer. The fifth step (Figure e) shows the customer using the scrip to make a purchase from the vendor. The last step (Figure f) shows a typical Millicent transaction. The customer already has vendor scrip and uses it to make a purchase. There are no extra messages or interactions with the broker.

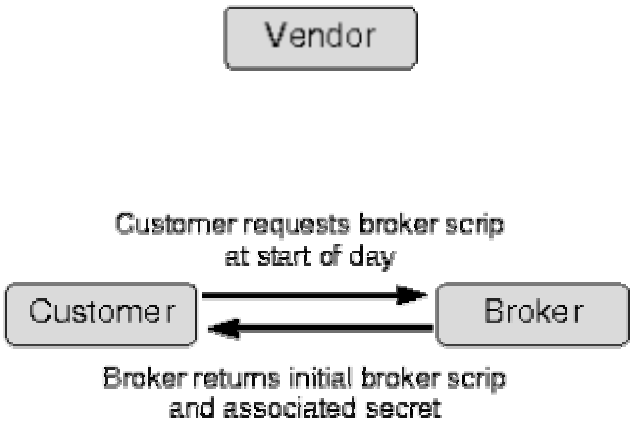

Figure a: The client makes a secure connection to the broker to get some broker scrip.

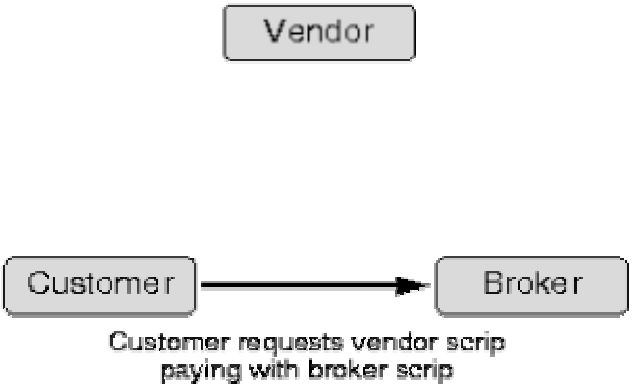

Figure b: If the client doesn't already have scrip for a particular vendor, he contacts the broker to buy some using his broker scrip.

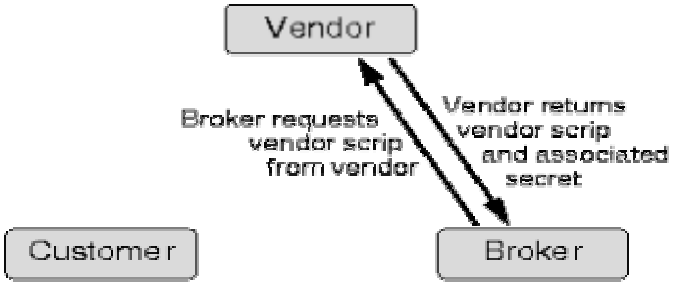

Figure c: If the broker doesn't already have scrip for that vendor, he buys some from the vendor.

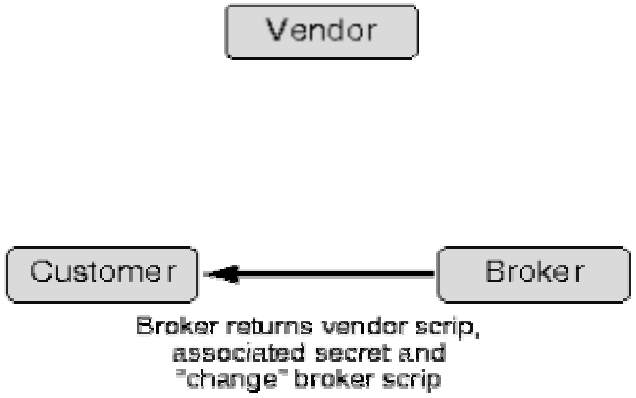

Figure d: The broker returns vendor scrip and change (in broker scrip) to the client.

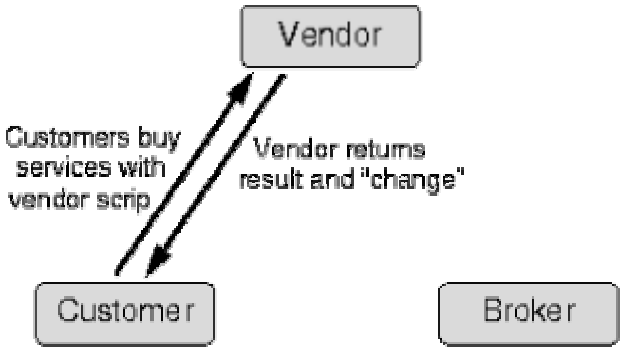

Figure e: The customer uses the vendor scrip to make a purchase from the vendor. The vendor returns change (in vendor scrip) to the client.

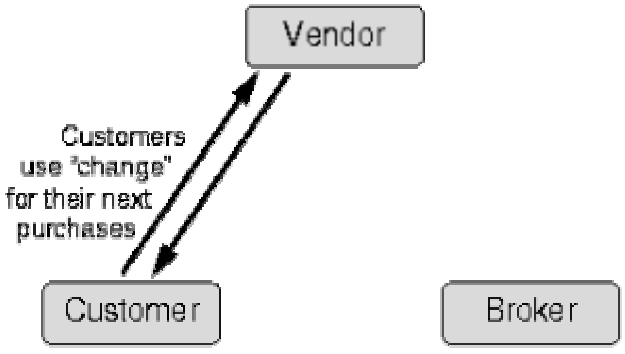

Figure f: The customer continues using the change to make more purchases.

6. MPTP

MPTP is an asynchronous protocol. Much of the processing required for the transaction may be done offline. In particular payment does not require an online communication with the broker. MPTP is also symmetric, there is no distinction between customer and vendor accounts except in relation to specific transactions where the flow of payment is generally in a single direction. MPTP may be layered on a variety of Internet protocols including HTTP and SMTP/MIME.

MPTP involves three parties, a customer *C* who makes the payment, a vendor *V* who receives the payment and a broker *B* who keeps accounts for the parties concerned. At present only a single broker model is considered, this means that both customer and vendor must share the same broker. Note however that the protocol does not restrict the broker to use of a single server. Support for Inter-Broker transfers will be required in the long term to permit the system to be scaled effectively.

A) Policy

MPTP is designed to be policy neutral, permitting a broker to offer a wide number of policy options. Individual vendors may thus choose the precise terms on which they offer goods. Customers may choose the terms they are willing to accept on a vendor by vendor basis

MPTP permits a considerable degree of flexibility in establishing a payments policy. A vendor may permit a customer a certain amount of trade before requiring a firm payment commitment or require all purchases to be paid for in advance. The first policy may be applicable where the goods offered cannot be evaluated by the customer in advance. A Vendor who has established a reputation with the customer may be in a position to insist on prior payment.

B) Mechanism

In the MPTP a payment order consists of two parts, a digitally signed payment authority and a separate payment token which determines the amount. A chained hash function is used to authenticate the token. To create the payment authority the customer first chooses a value w_n at random. The customer then calculates a chain of payment tokens (or *paychain*) $w_0, w_1, \ldots w_n$ by computing

$$w_i = h\,(w_i+1)$$

Where h is a cryptographically secure one way has function such as MD5 or SHA .

The signed payment authority contains w_0, the root of the payment chain and defines a value for each link in the chain. Payments are made by revealing successive paychain tokens. Once the vendor or broker has authenticated a payment authority an arbitrary payment token may be authenticated by performing successive hash functions and comparing against the root value. It should be noted however that the broker is only presented with the final payment order. It is therefore unnecessary for the broker to maintain large online databases.

MPTP permits use of double payment chains. This allows implementation of a broker mediated satisfaction guarantee scheme. The pair of payment chains represent the high and low

watermarks for the payment order. The low watermark chain represents the amount that the customer has fully committed to pay. The high watermark chain represents partial commitments. The vendor exposure is the difference between the counter values. MPTP also supports use of multiple payment counters denoting different units of currency. This allows some optimization of processing time through shortening of the payment chains.

MPTP provides protection against double spending through vendor and broker checking of authority identifiers. The size of required Vendor authority identifier matching tables (th *double spending buffer*may be controlled by checking that the authority timestamp is within bounds. An alternative approach would incorporate challenge/response sequence into the session establishment protocol. This could be used to simplify broker double spending prevention measures if constraints were placed on the challenge identifiers. The reduction in vendor resource requirements do not appear to justify an additional round trip delay however.

The mechanism could be modified to use a collection of payment tokens as opposed to a chain. Each token would consist of a the hash of a shared secret which would be revealed to make a payment. This might provide a solution to possible patent difficulties concerning the use of the Lamport hash chain mechanism. It would also permit payments to take place in parallel.

C) Signature

MPTP permits use of both shared secret and public key based signature schemes. Schneier describes a wide variety of public key signatures schemes and one way hash functions suitable for constructing Message Authentication Codes *(MAC*s). Choice of algorithm, key length etc. is left to the parties involved. It is desirable to minimize the latency introduced in the signing of the initial payment order and also to minimize computational needs of the vendor and broker.

D) Certificates and establishment of trust.

Certificates bind a public key to an account number under the public key of the broker. It is assumed that the broker public key is known to all parties. Each party generates their own public-private key pair locally. The public key certificate is communicated to the broker at account establishment.

Account revocation lists are supported to enable credit risks to be prevented from engaging in further abuse. Separate certificate revocation lists are not supported since compromise of public key certificates may be dealt with through the same mechanism.

The following certificate attributes are supported:

I) IP-Address mask, value

Specifies a set of internet addresses for which the certificate is valid. Only payment requests originating from IP addresses which equal the specified value after being logically ANDed with the mask. If more than one IP-Address attribute is specified a single match is sufficient.

II) Not Guaranteed amount

The broker will not guarantee that payment will be made for amounts exceeding the specified amount.

III) Guaranteed amount

The broker guarantees payment up to the specified amount without separate authorization.

IV) Authorization-Required *amount*

Payments above the specified amount require separate authorization to be guaranteed.

E) CREDIT LIABILITY

If a broker chooses to act as guarantor for a payment a credit liability risk may be incurred. Note that MPTP supports an option for the broker to transfer this risk to the vendor by refusing a guarantee of payment.

In either case a credit liability is incurred. Such liabilities are a familiar consideration in the financial industry. A similar risk is accepted in parts of the publishing industry where newspapers are sold from unattended vending machines which cannot control the number of copies taken by each customer. In certain countries no precautions are taken to prevent a copy being taken without any payment at all.

F) Credit Abuse

The problem of credit abuse is linked to but distinct from that of credit liability risks. For example an account might be created in a false name and its authentication information widely published with the intention of permitting general access to charged material for free.

Credit abuse might be discovered through broker tracing of payment patterns to detect sudden increases in payment activity and then terminated through the revocation list mechanism. The case of widespread use of a single connection may be controlled through checking of the certificate IP-Address attribute if specified. If no IP address attribute is specified a vendor might employ code to detect accesses from multiple IP addresses within a suspiciously short interval.

G) Counterfeiting

MPTP payment orders are vendor specific and digitally signed. Provided the signature scheme is secure it is not possible for a party to construct a payment order without having access to the secret information corresponding to the key.

H) Unauthorized Withdrawal

Unauthorized withdrawal is not possible without detection by the account holder who may require an audit trail from the broker for each transaction. Note that this requires the broker to maintain a substantial quantity of online logging information.

I) PURCHASE ORDER MODIFICATION

In a purchase order modification attack an external party modifies a purchase request in order to cause different goods to be delivered. This risk is not directly addressed in the MPTP scheme although the satisfaction guaranteed policy might be used to protect the customer. Without authentication of the purchase order there is no method of avoiding this attack. The costof this authentication might be reduced by establishing a shared key between vendor and customer during the session establishment protocol. Such shared keys might have a lifetime spanning several payment orders.

J) Double Spending

Payment orders are specific to a particular vendor and carry a unique authority identifier. A broker is required to detect an attempt to deposit the same payment order more than once and act accordingly. In some cases this may mean increasing the amount of payment authorized.

K) Failure to Credit Payment

Currently MPTP does not address this risk. A Broker may deliberately deduct a payment amount from the account of one party without making a corresponding credit to another party.
One approach to this problem is to make information concerning bad debts available for scrutiny. A broker might be required to issue a frequent list of bad debts signed under the broker's public key. Such debts might be rendered unlikable through a use of a one way hash function on the authority identifier. The proportion of bad debts might be concealed through addition of padding. In this way both customer and vendor could ensure that the broker acted in good faith.

L) Denial of Service

Denial of service is a significant risk, unfortunately it is one that the underlying infrastructure of the Internet does not protect against. Consequently any application protocol level protection against a denial of service attack can at best provide limited protection against this risk. Use of Shamir's signature screening algorithm substantially reduces the risk of a denial of service attack against a vendor or broker through construction of bogus payment orders.

M) Repudiation

MPTP payment orders are non-repudiable in the sense that the customer cannot deny having made a payment authorization. This is distinct from the option for a vendor or broker to permit a customer the right to refuse payment after receiving the goods.

N) Failure to deliver

Failure to deliver may occur for many reasons including vendor fraud. The Internet is an unreliable transport medium and a customer may in good faith offer to buy an article and a vendor in good faith may intend to supply but delivery fail nevertheless. The HTTP protocol in particular does not currently provide for customer acknowledgment of receipt. One solution to the failure to deliver risk is to permit the customer to refuse payment through the “satisfaction guaranteed" policy.

O) FRAMING

The vendor has the opportunity to frame a customer, albeit at a direct monetary loss to himself. In this scenario a vendor receives a valid payment chain from a customer but chooses not to deliver the authorization paychain token, instead delivering only the promissory paychain token. The vendor is thus able to frame the customer, albeit at the cost of the payment.

P) Payment Flow

I) Session Establishment by Customer

The customer performs the following steps to create an Authority:
1). Calculates paychains, stores head, may additionally store all or part of paychain.
2.) Creates unique authority identifier. Alternatively the paychain root might be used for this purpose.
3). Fills remaining slots in Authority structure.
4). The authority is sent to the vendor.

II) Session Establishment by Vendor

On receipt of an Authority the vendor performs the following steps:
1). The date of the authority is checked to ensure that it is within the vendor determined permitted timeframe.

2.) The authority identifier is checked against those in the double spending check buffer.
3). The authority identifier is added to the double spending check buffer.
4). If public key signatures are used the signature of the customer certificate validated.
5). If public key signatures are used: The signature of the authority is validated.
6). If the account certificate does not offer the required payment guarantees or symmetric signatures are used: A validation request is performed.
7). The Authority is appended to the online file.

III) Session Establishment with Validation Request by Vendor

If the vendor determines that an account enquiry is required an account enquiry is created:

1). The account enquiry packet is created
2). The account enquiry is authenticated using a MAC and a shared secret established between vendor and broker.
3). The account enquiry is sent to the broker.

IV) Session Establishment with Validation Request by Broker

1). A Validation Request is received.
2). The validation request is authenticated.
3). The account information corresponding to the customer id is retrieved.

4). A decision is made to accept or reject the authorization.

5). The Validation Response is sent to the Vendor .

V) Session Establishment with Validation Request by Vendor
On receipt of an account enquiry response a vendor:

1). Checks to see that the response is genuine.
2). Checks to see that the account is authenticated and the required payments guarantees provided.

The Customer may then send a sequence of PayWords which are processed as follows:

VI) Payment Transfer by Customer

The Customer prepares a Charge message as follows:
1). The Authority information corresponding to the vendor id is retrieved.
2). The Payword(s) corresponding to the desired payment amount is determined.
3). The Charge message is sent to the vendor.

VII) Payment Transfer by Vendor

The Vendor processes the Charge message as follows:

1). The vendor receives the charge message.
2). The session record is retrieved using the authority-id.
3). The PayWord is validated using the paychain root.
4). The PayWord information and increment are updated in the session record.

7. Parameters

A) Computational Cost

For computational cost, a rough guide is that hash functions are about 100 times faster than RSA signature verification, and about 10,000 times faster than 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.

PayWord's computational requirements are that the broker needs to sign each user certificate, verify each user commitment, and perform one hash function application per payment, however all these computations are offline. The user needs to verify his certificates, sign each of his commitments, and perform one hash function application per PayWord committed to. The vendor verifies all certificates and commitments received, and perform one hash function application per PayWord received.

In Micromint the broker has to mint coins, for this purpose the broker will create 2n bins, toss approximately k^{2n} balls, and create one coin from each bin that now contains at least k balls. With this choice of parameters each ball has a chance of roughly 1/2 of being part of a coin. The vendor's task of verifying a coin remains extremely efficient, requiring only k hash computations and a few comparisons.

In NetBill each transaction requires seven symmetric encryption/decryption operations and three Hashing operations.

In Millicent the scrip is digitally signed by the broker to prevent tampering and counterfeiting. The customer signs each use of scrip with a secret that is associated with the scrip. The Vendor can efficiently check the script using a fast one-way hash function like MD5 or SHA.

In MPTP the customer bears the most substantial processing costs. Establishment requires the creation of a pay-chain and digital signature. The vendor must process two signature verifications per establishment of a payment session and one hash operation per PayWord transferred. The broker must perform one signature verification per collection, plus one hash calculation per PayWord transferred. All broker calculations may be performed offline.

B) Communication Cost

Emerging applications in electronic commerce of today involve very low-cost transactions, which execute in the context of ongoing, extended client-server relationships. For example, consider a web-site (server) which uses repeated authenticated personalized stock quotes to each of its subscribers (clients). The value of a single transaction (e.g., de-livery of a web-page with a customized set of quotes) does not warrant the cost of executing a handshake and key distribution protocol.
Using the following noteworthy techni-cal aspects we have performed the comparison.

- Client-side shared key computation
- Client-side shared key management
- Server-side shared key management
- Modular structure

Most prominent factors identified that affect the computation cost are Secrecy, Consistency, Efficiency, Modular Security and Impersonation resistance.

In Payword scheme vendor has to communicate to broker once a day for the payment of money in return of PayWords which he earned that day. User has to communicate to the Vendor when the purchase of goods takes place. User has to communicate to the vendor once a month for the exchange of PayWords.

In Micromint scheme vendor has to communicate to broker once a day for the payment of money in return of coins. User has to communicate to the Vendor when the purchase of goods takes place. User has to communicate to the broker once a month for the purchase of new coins and returning the unspent coins of last month.

In NetBill each transaction requires eight (8) interactions between customer, merchant and NetBill server. The customer sends an authenticated request for a quote to the merchant. The merchant then determine a price for the authenticated user and returns the digitally signed price quote to the customer. The customer sends a digitally signed purchase request to the merchant. The merchant then sends the encrypted goods to the customer. Now customer sends the digitally signed electronic payment order to the merchant. The merchant sends this EPO along with the decryption key to the net bill server. The NetBill server returns merchant a digitally signed message containing an approval, or an error code indicating why the transaction failed. The merchant then forwards the NetBill server's reply to the customer along with the decryption key.

Millicent requires four (4) interactions between customer, vendor and broker for each transaction. First customer requests broker the scrip of vendor for payment to the vendor against broker scrip.

Broker returns the vendor scrip to customer. Customer then requests the vendor for goods against vendor scrip. Vendor then provides the goods to customer.

In MPTP scheme vendor has to communicate to broker once a day for the payment of money in return of PayWords which he earned that day. User has to communicate to the Vendor when the purchase of goods takes place. User has to communicate to the vendor once a month for the exchange of PayWords.

C) Storage Cost

In PayWord the Broker should store all the certificates issued by him to the users .The vendor should store verified commitments until they expire at the end of the day. Otherwise, if he redeemed (and forgot) PayWords received before the expiration date of the commitment, user could cheat vendor by replay attack. The user should preferably also store his commitment until he believes that he is finished ordering information from vendor.

In Micromint scheme the broker keeps a record of coins that user has bought from him. Vendor also stores the coins which he earned during the day and at the end of day he returns them to broker for money. User stores all his coins which he bought at the start of the month till the end of month, he purchased goods by spending these coins and at the end of month he returns the unspent coins.

NetBill server stores each transaction's complete information. This information all resides in Electronic Payment Order. In EPO following important identities are stored.
These are User Identity , Product ID , Price , Merchant ID, Cryptographic check sum of goods etc.

In Millicent broker stores the scrip of all the vendors with which his customers interacts and give these scrip to users for purchase of goods, and similarly vendor stores the scrip of all brokers collected from the customers in return of goods against which brokers paid him. User only stores only the scrip of vendors which he obtained from the broker.

In MPTP Customer stores the computed paychains in complete or partial form. The vendor must maintain an online record for each open session. This record is fixed length consisting of the authority identifier and payer identifier from the authority, and the paychain root or most recent valid pay-word plus the currency unit. If the symmetric signature option is not provided the broker may perform almost all operations offline in batch. Incoming collection requests from vendors may be pre-processed to optimize access to secondary storage such as disk. Detection of double spending requires a record of all transactions to be available at the time when a record is added. This need not involve the expense of online memory however.

D) Privacy

Privacy is a necessary concern in electronic commerce. It is difficult, if not impossible, to complete a transaction without revealing some personal data – a shipping address, billing information, or product preference. Users may be unwilling to provide this necessary information or even to browse online if they believe their privacy is invaded or threatened. People are concerned about privacy, particularly on the Internet.

Some parties involved may wish confidentiality of transactions. Confidentiality in this context means the restriction of the knowledge about various pieces of information related to a transaction: the identity of payer/payee, purchase content, amount, and so on. Where anonymity or un-traceability are desired, the requirement may be to limit this knowledge to certain subsets of the participants only.

In Payword scheme privacy is lost because of the use of email address in each transaction. Privacy is achieved in Micromint since no user data is sent to vendor. However if the user specific coins are used than privacy is lost. In NetBill privacy is achieved since no user data is sent to vendor. In Millicent the secrets are used in a way that shows knowledge of the secret without revealing the secret. To attest to a message, the secret is appended to the message, and the result is hashed to produce a signature. In MPTP scheme privacy is achieved since (certificate) authority is used and no user information is received.

E) Repudiation

Repudiation is that the originator of a message falsely denies later that they were the party that sent the message. It is much easier to repudiate an electronic business transaction then in the Cash based system. Thus the protocol should prevent the denial of previous commitments or actions. This can be achieved through digital signatures. Digital Signatures are bit patterns that depend upon the message being signed and use some information unique of the sender.

In Payword the user cannot repudiate since the PayWord is digitally signed by private key of user and the Broker also cannot repudiate since the PayWord is digitally signed by private key of Broker. Micromint transactions are non repudiable since the coins are user specific and Broker cannot repudiate since the coins are generated by broker and can be checked. NetBill transactions are also non repudiable since digital signature of user are used. Millicent ensures non repudiability by enforcing the customer to sign each use of scrip with a secret that is associated with the scrip. MPTP payment orders are non-repudiable in the sense that the customer cannot deny payment authorization made by him.

F) Reliability

Fault tolerance and reliability is a must in any enterprise class deployment. Commercial web applications in fields such as shopping, financial and stock markets constitute the most important market segment for high-performance servers today. As the popularity of such commercial applications on web servers increase, there is an increasing amount of dynamic pages using server side scripts, as opposed to static html pages that dominated internet traffic until a few years ago. This increases the workload on servers that host these sites significantly, such as increased CPU time, memory usage and response time spent servicing requests. Such on-line transaction processing (OLTP) workloads provide a challenging set of requirements for Fault Tolerance, such as High Availability and Reliability. Even a slight reduction of availability and reliability in such Servers could lead to tremendous business losses. Studies have shown that service downtime, even for relatively brief periods can cause major losses

of revenue, staff hours, and customer confidence and on customer side can also cause both financial and psychological agony. It is therefore no longer acceptable to be running mission critical or revenue- generating services without some kind of fault tolerant and reliability features.

A reliable payment system is the one that allows no money to be taken from a user without explicit authorization by that user. It may also disallow the receipt of payment without explicit consent, to prevent occurrences of things like unsolicited bribery. Payment transactions must be atomic: They occur entirely or not at all, but they never hang in an unknown or inconsistent state. Moreover the system should be available at all time. Denial of service is a significant risk, unfortunately it is one that the underlying infrastructure of the Internet does not protect against.

Payword and Micromint are highly reliable due to the use of multi-Broker and multi-Vendor schemes. Thus the failure of one of the broker or the vendor does not affect the overall system. NetBill is reliable since all the transactions are atomic they occur entirely or not at all, but they never hang in an unknown or inconsistent state however it can be badly effected by the Dos/ DDos Attacks resulting in total failure of the system. Millicent and MPTP also provide high reliability by using scheme of multi-Broker and multi-Vendor scheme.

G) Offline / Online

Concern in online transactions is always about how the online medium influences satisfaction and loyalty and the relationship between satisfaction and loyalty. Typically, online customers can more easily compare alternatives than offline customers, especially for functional products and services. A competing offer is just a few clicks away on the Internet.

Because of this potential for "frictionless commerce," many managers fear that the online environment might raise customers' expectations about the service, making them less satisfied and also more prone to switching to, or among, competing services. In other words, the online medium may induce lower customer satisfaction and loyalty compared to the offline medium, and that increased satisfaction with a service may not lead to higher loyalty when that service is chosen online. [9].
But research survey have showed that people care more about the actual service received, which apparently is no different whether the service is chosen online or offline [9]

In an online scheme, when a vendor receives digital cash, he must contact the issuer to see if it is valid and not already spent. This extra communication makes the central site a bottleneck and adds cost to the transaction. In an off-line scheme, the vendor authenticates the digital cash during the transaction and then later transmits it to the issuer. This scheme adds computational costs to the vendor for authenticating the digital cash, and adds messages and encryption to the protocol for pinpointing the source of the double spending.

Payword, Micromint, MPTP and Millicent are the offline schemes while the NetBill is the online scheme.

G)Trust

An important barrier to the widespread diffusion of electronic commerce among consumers is "the fundamental lack of faith between most businesses and consumers on the web today. In essence, consumers simply do not trust most Web providers enough to engage in relationship exchanges (financial / commercial).

None of these protocols discusses trust as a separate criteria and it is a highly under explored topic needing much research. In simple terms, trust can be defined as the belief by one party about another party that the other party will behave in a predictable manner (Luhmann 1979) [11].

Two important elements of trust by a focal party about the other party are: (1) the perception of risk and vulnerability by the focal party in dealing with the other party and (2) the expectation that the other party will behave in the interest of the focal party (Rousseau et al. 1998).[12] Trust has been extensively studied in communication, computer science, information systems, management, marketing, philosophy, psychology, and political science since the 1950's. Although each field has its own definition(s), they all have contributed to a better understanding of trust in general. According to our understanding two types of trust exist:
(1) offline trust that involves the offline activities of the firm and its relationships with its customers and other stakeholders. (2) online trust that involves the firm's business activities in the e-medium.
Although online trust is similar to offline trust in many ways, there are some important distinctions. In offline trust, the object of trust is typically a human or an entity (organization). In online trust, typically, the technology (mainly the Internet) itself is a proper object of trust (Marcella 1999). In a sense, a firm's Web site is its salesperson that needs to build trust with her/his customers (Jarvenpaa et al. 1999). There is, however, some degree of overlap or transfer of trust between the online and offline environments.

It is obvious form the above discussion that online trust is to be the main concern when having a technical peek at the E-commerce. With the emergence of multiple touch point or multi channel marketing, consistency in online trust and in trust across the multiple touch points is becoming important. Dayal et al. (1999) discuss security, merchant legitimacy and fulfillment as important determinants of online trust and claims that trustworthiness affects credibility [13].

We can also use virtual-advisor technology to gain customer confidence and belief, provide unbiased and complete information and design protocols to keep tag and include information on competitive products and tag promises to promote trust. Reliability in fulfillment is a key aspect of trust, so reliability issue concern is a good step towards next generation trustworthy protocols. Furthermore disclosed patterns of past performance, references from past and current users, get third-party certifications, and make locality assurance easy, awareness and enforcing of policies involving privacy and security must be there. Trust can be enhanced by credit card loss assurance, warranty and merchandise return policies, availability of escrow service, ability to schedule human customer service, and availability of user friendly

interfaces. Privacy statements and third party involvement can improve trust (Palmer et al. 2000). Because different organizations such as retailer, shipping courier, and bank are involved in an online transaction, online trust may increase if these organizations are properly rated and compiled result accessible to the user.

The consequences of above include intention to act, satisfaction, loyalty, traffic, price, revenues and profitability. Online trust is a relatively under explored topic that offers several promising avenues for future research including the roles of multiple stakeholders, the impact of strategic alliances etc.

Keeping in view the above discussion no protocol of today fulfills these quality attributes of trust thus is not a candidate of being fully trustworthy but can claim to be partially trustworthy.

8. CONCLUSION

For users to adopt e-commerce, it is imperative that the benefits of using this new commercial medium significantly outweigh potential risks and inconveniences. Indeed, difficulty of use and lack of trust with respect to online payment, privacy and consumer service have been found to constitute a real psychological barrier to e-commerce. Although it is not up to interaction designers and usability engineers to solve issues linked to legislation or cryptography, it is argued that they can nevertheless play an important role in ensuring the trustworthiness.

We discussed the fundamental problems of designing and analyzing protocols specialized towards secure E-commerce which formalizes interactions in this highly sophisticated and intricate environment. We have discussed and analyzed five emerging micro commerce protocols namely NetBill, Payword, Micromint, Millicent and Micro payment transfer protocol. The analysis is done the basis of eight parameters which are Computational cost, Communication Cost, Storage cost, Repudiation, Reliability, Privacy and online/offline. It has been established in the research that protocol design is a mounting problem since so many tradeoffs have to be considered. These issues are as old as communication itself. Only when the interpretation of the protocol rules had to be automated in technically diverse environment, was it discovered that protocol design in itself can be a challenging problem. The protocols being developed for e-commerce are larger and more sophisticated than ever before. They try to offer more functionality and reliability, but as a result they have increased in size and in complexity. The problem that a designer now faces is fundamental, that is, how to design large sets of rules for information exchange that are minimal, logically consistent, complete, and efficiently implemented.

The problem in all such systems is to come up with an unambiguous set of rules that allows one to initiate, maintain, and complete information exchanges reliably.

During this research venture it is concluded by the study of architectural designs that none of these five protocols is a whole solution, but there exist tradeoffs. The main trade off is between security and cost. If we increase the security then the overall cost also increases as in the case of NetBill. By addressing these issues we hope to converge towards a unanimously adoptable scheme in this regard. But due to such diversity in issues there is little chance that the world will agree on a single scheme for electronic payments in near future. However, the world needs one card holder scheme, not one per brand or one per country.

Future work

A Forrester survey found that 51% of companies would not trade with parties they do not trust over the Web. Lack of trust is one of the greatest barriers inhibiting online trade between buyers and sellers who are unfamiliar with one another [10].

On basis of this research we intend to provide a conceptual architectural framework for single ecommerce scheme to get rid of most of the shortfalls of the current ecommerce protocols and give a research bed for ecommerce on basis of which world will agree on a single scheme for electronic payments in near future. As mentioned above the world needs one card holder scheme, not one per brand or one per country.

REFERENCES

[1] M. Sirbu and J. D. Tygar. " NetBill: An Internet Commerce System Optimized for Network Delivered Services ". In *IEEE Personal Communications*, 2(4) pages 34Ð39, August 1995.

[2] M. Sirbu and J. D. Tygar. " NetBill Security and Transaction Protocol ". Carnegie Mellon University Pittsburgh, January 1996.

[3] Ronald L. Rivest, *Addi Shamir*. "Payword and Micromint". In *Proceedings of the International Symposium on Ecommerce*, October 2002.

[4] Steve Glassman et all. The Millicent protocol for inexpensive electronic commerce

[5] MicroPayment Transfer Protocol (MPTP) Version 0.1, W3C Working Draft, November 1995

[6] Mark E. Peters. Emerging eCommerce Credit and Debit Protocols**,** In p*roceedings of the third International Symposium on Ecommerce*, IBM Corporation, October 2002.

[7] Randy C. Marchany and Joseph G. Tront. E-Commerce Security Issues, Proceedings of the 35th Hawaii International Conference on System Sciences, 2002

[8] N. Asokan and Michael Waidner. The State of the Art in Electronic Payment Systems, IBM Research Laboratory, 1997

[9] Customer Satisfaction and Loyalty in Online and Offline Environments Venkatesh Shankar, Amy K. Smith, Arvind Rangaswamy. October 2000, Revised, May 2002

[10]Beyond "web of trust": Enabling P2P E-commerce Anwitaman Datta, Manfred Hauswirth, Karl Aberer

Distributed Information Systems Lab ´ Ecole Polytechnique F´ed´erale de Lausanne (EPFL) CH-1015 Lausanne, Switzerland

[11] LUHMANN, N. (1979), TRUST AND POWER, JOHN WILEY AND SONS, LONDON.

[12] Rousseau, Denise, M. Sitkin, Sim B. Ronals S. (1998). "Not so Different after All: A Cross-discipline View of Trust" Academy of Management Review, 23 (3), 393-404.

[13] Dayal, Sandeep, Helene Landesberg, and Michael Zeisser (1999), "How to Build Trust Online," Marketing Management, Fall, 64-69.