NetBill Security and Transaction Protocol

Benjamin Cox, J. D. Tygar, and Marvin Sirbu
Carnegie Mellon University
Pittsburgh, PA

NetBill Security and Transaction Protocol

Benjamin Cox — *thoth+@cmu.edu*
J. D. Tygar — *tygar@cmu.edu*
Marvin Sirbu — *sirbu+@cmu.edu*

Carnegie Mellon University
Pittsburgh, PA 15213–3890

Abstract

NetBill is a system for micropayments for information goods on the Internet. This paper presents the NetBill protocol and describes its security and transactional features. Among our key innovations are:

- *An atomic certified delivery method so that a customer pays if and only if she receives her information goods intact.*
- *Outsourcing access control: different users can use different access control servers.*
- *A credential mechanism allowing users to prove membership in groups. This supports discounts.*
- *A structure for constructing pseudonyms to protect the identities of consumers.*

1. Introduction and Overview

Buyers and sellers increasingly want to use the Internet to conduct their business electronically. As a base for commerce, the Internet poses special challenges due to its lack of standard security mechanisms. At the same time, the ease with which buyers can peruse catalogs published via the World Wide Web makes the Internet attractive for commerce. Consumers will want to use the Internet as a means for multiple phases of the purchase process: searching for suppliers, price negotiation, ordering, and payment for goods. In the case of information items, such as software or journal pages, the Internet can deliver the items.

Using the Internet for commerce poses new variations on traditional issues. Transactions occur in cyberspace with no easily identifiable place of business for the merchant or physical delivery site for the customer. Transactions are subject to observation by third parties sharing the network. And the use of computers to support transactions makes record keeping easier, exacerbating privacy problems arising from transaction data collection by merchants.

Supporting transactions in cyberspace requires electronic analogs for many familiar procedures from face-to-face transactions. Parties need to know with whom they are dealing, or at least verify their creditworthiness. They need to be able to negotiate prices, perhaps providing credentials entitling them to special discounts, such as a student ID. Parents need methods to control where their children shop in cyberspace. In the case of information goods, the value of an item may be as low as a few cents, requiring transaction mechanisms which impose per-transaction overheads much smaller than those for typical check and credit card purchases. Merchants need to restrict the class of customers they support based on their credentials, to restrict distribution of sensitive materials.

We are building a system called NetBill which is optimized for the selling and delivery of low-priced network goods. A customer, represented by a client computer, wishes to buy information from a merchant's server. An account server (the NetBill server), maintains accounts for both customers and merchants, linked to conventional financial institutions. A NetBill transaction transfers information goods from merchant to customer, debiting the customer's NetBill account and crediting 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, funds in a merchant's NetBill account are made available by depositing them in the merchant's bank account. NetBill acts as an aggregator to combine many small transactions into larger conventional transactions, amortizing conventional overhead fees.

The transfer of an information good consists of delivering bits to the customer. Users may be charged on a per item basis, by a subscription allowing unlimited

Sponsored by the Air Force Materiel Command, under Advanced Research Projects Agency contract No. F19628–95–C–0018, "Electronic Commerce: The NetBill Project." Additional support was received from the National Science Foundation under Cooperative Agreement No. IRI–9411299, and from Visa International.

access, or by a number of other pricing models. A more detailed examination of the NetBill model may be found in [10].

NetBill requires an efficient set of protocols to support price negotiation, goods delivery and payment. This paper outlines our protocols.

Among our key innovations are:

- A method of (atomic) certified delivery so that a customer pays if and only if she receives her information goods intact.
- A system for allowing access control to be outsourced—so that different users may use different access control servers. (For example, some children's accesses may be governed by a PTA access control server, while other children may be under the domain of access control servers set up by a church group.)
- A credential mechanism for allowing users to easily prove membership in groups, to qualify for discounts or for other purposes.
- A structure for easily constructing pseudonyms so that buyers of information can protect their identities.

2. The NetBill Transaction Model

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

In a NetBill transaction, the customer and merchant interact with each other in the first two phases; the NetBill server is not involved until the payment phase, when the merchant submits a transaction request. The customer contacts the NetBill server directly only in the case of communications failure or when requesting administrative functions. Figure 1 shows the relationships among parties in a NetBill transaction.

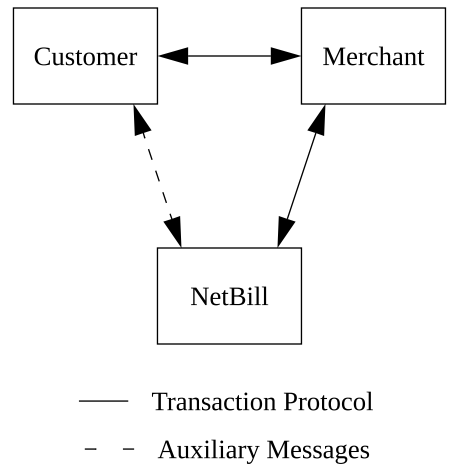

Figure 1: Parties in a NetBill Transaction.

2.1. Transaction Objectives

For a NetBill transaction, we have the following set of objectives. (Similar versions of objectives (a)–(d) below can be found in [2].)

a) Only authorized customers can charge against a NetBill account.

b) The customer and merchant must agree on the item to be purchased and the price to be charged.

c) A customer can optionally protect her identity from merchants.

d) Customers and merchants are provided with proof of transaction results from NetBill.

In addition, we have the following objectives to support price negotiation and goods delivery.

e) There is an offer and acceptance negotiation phase between customer and merchant.

f) A customer may present credentials identifying her as entitled to special pricing or treatment.

g) A customer receives the information goods she purchases if and only if she is charged (and thus the merchant is paid) for the goods.

h) A customer may need approval from a fourth (access control) party before the NetBill server will allow a transaction.

Finally, we add as a general objective for all phases of the purchase process:

i) The privacy and integrity of communications is protected from observation or alteration by external parties.

To achieve these goals, the NetBill protocol provides for strong authentication and privacy, atomic payment and delivery protocols, and a flexible access control system.

In the price negotiation phase, the customer presents evidence of her identity, and (optionally) supplemental credentials, and requests a price quote on an item. The customer may also include a bid for the item. The merchant responds with a price offer.

In the second phase, the customer accepts or declines the offer. In the case of information goods, acceptance constitutes an order for network delivery. The merchant provisionally delivers the goods, under encryption, but withholds the key.

Key delivery is linked to completion of the third phase, the payment protocol. In this phase, the customer constructs, and digitally signs, an electronic payment order (or *EPO*) and sends it to the merchant. The merchant appends the key to the EPO and *endorses* (digitally signs) the EPO, forwarding it to the NetBill server. The NetBill server returns a digitally signed receipt, which includes the key, to the merchant, who forwards a copy to the customer.

3. The Transaction Protocol

We use the notation "X $\Longrightarrow$ Y" to indicate that *X* sends the specified message to *Y.* The basic protocol consists of eight steps, which are:

1. C $\Longrightarrow$ M   Price request
2. M $\Longrightarrow$ C   Price quote
3. C $\Longrightarrow$ M   Goods request
4. M $\Longrightarrow$ C   Goods, encrypted with a key *K*
5. C $\Longrightarrow$ M   Signed Electronic Payment Order
6. M $\Longrightarrow$ N   Endorsed EPO (including *K*)
7. N $\Longrightarrow$ M   Signed result (including *K*)
8. M $\Longrightarrow$ C   Signed result (including *K*)

Objective (a) from Section 2.1 is realized because the customer must be authenticated to NetBill before the EPO (generated in step 5) will be accepted (in step 6).

Objective (b) is achieved because the relevant information is included in the EPO which must be signed by the customer in step 5 and endorsed by the merchant in step 6.

Section 4.2 presents a mechanism implementing objective (c).

Objective (d) is realized through the digitally signed receipt from NetBill in step 7.

Objective (e) is achieved by the exchange in steps 1 and 2 of the protocol.

Section 5.1 presents a solution implementing objective (f).

Objective (g) is realized by the exchange in steps 4–8, which we call *certified delivery.* The customer first gets a version of the goods encrypted with a key *K*. The goods are also cryptographically checksummed. In this way, the customer uses the checksum to confirm that she received the goods without transmission error. The customer returns the checksum to the merchant together with other information, and that message is forwarded to the NetBill server. The key *K* that is needed to decrypt the goods is registered with the NetBill server and also sent to the customer (step 8). The exchange in steps 4–8 provides protection to the customer against fraud by the merchant. For example, suppose there is a discrepancy between what the customer ordered and what the merchant delivered. The customer can easily demonstrate the discrepancy to a third party (such as a judge). The customer has NetBill's receipt (step 7, forwarded in step 8) indicating what was ordered, the amount charged and the key *K* reported to NetBill by the merchant. The customer also has registered with NetBill the checksum of the encrypted goods. Thus if the goods are faulty (*e.g.,* purchased software doesn't run), it is easy to demonstrate that the problem lies with the goods as sent and not with any subsequent alteration. This certified delivery technique also protects the merchant by requiring the customer to pay and the payment to clear through the NetBill server before the customer gets the use of the goods. (A more general discussion of the role of certified delivery and atomicity in general for electronic commerce can be found in [13].)

Section 5.2 presents a solution for objective (h).

Objective (i) is realized by encrypting communications between all pairs of parties and providing integrity checks on those messages.

3.1. Notation

We use the following notation to denote cryptographic operations. *X* and *Y* always represent communicating parties. *K* always represents a cipher key. The protocol is a sequence of messages exchanged among three parties: *C*, the customer; *M*, the merchant; and *N*, the NetBill server.

T_{XY}(Identity)	A Kerberos ticket proving to *Y* that *X* is named by *Identity*, and establishing a session key *XY* shared between them.
CC(*Message*)	A cryptographic checksum of *Message*, using an algorithm such as the Secure Hash Algorithm (SHA) hash function presented in [5].

E_K(*Message*)	*Message*, encrypted by a symmetric cipher using key *K*. The key *K* may be denoted as *XY*, meaning that it is known only to parties *X* and *Y*. The encrypted message implicitly includes a nonce.
$E_{X\text{-}PUB}$(*Message*)	*Message*, encrypted in party *X*'s public key using the RSA cryptosystem as presented in [8].
$E_{X\text{-}PRI}$(*Message*)	*Message*, encrypted in party *Y*'s private key using RSA.
[*Message*]$_X$	*Message*, clearsigned by *X* using RSA public key cryptography. Clearsigning is implemented by forming *Message*, *Timestamp*, $E_{X\text{-}PRI}$(CC(*Message*, *Timestamp*)). This is computationally efficient and allows any party to read the *Message* text without needing *X*'s public key. The clearsigned item implicitly includes a nonce.
[*Message*]$_{X\text{-}DSA}$	*Message*, signed and timestamped by *X* using the Digital Signature Algorithm (DSA) as described in [6].
{*Message*}^{X}	*Message*, encrypted for *X* using RSA public key cryptography. For computational efficiency, this is implemented by forming E_K(*Message*), $E_{X\text{-}PUB}$(*K*). The encrypted message implicitly includes a nonce.

3.2. The Price Request Phase

This section assumes possession of tickets; the method for obtaining tickets is shown in Section 4. The *Identity* item may actually be a pseudonym, as shown in Section 4.2.

1.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2.	M $\Rightarrow$ C	E_{CM}(ProductID, Price, RequestFlags, TID)

These two steps represent a request/response message pair in which the customer requests a price quote of the merchant. The customer presents an identifying ticket (the identity presented may be a pseudonym; see Section 4.2 for details on pseudonyms) to the merchant, along with some optional *credentials* establishing her membership in groups which may make her eligible for a discount. (We discuss these credentials in Section 5.1.)

The customer passes parameters indicating Product Request Data (*PRD*, an arbitrary stream of application-specific data which the customer and merchant use to specify the goods) and some flags. These *RequestFlags* are the customer's indication of her request for the disposition of the transaction (*i.e.*, delivery instructions; see Section 3.6 for different transaction options).

The customer may also optionally include a Bid, indicating to the merchant a price she may be willing to pay for the item.

The Transaction ID, *TID*, is optional in step 1. Steps 1 and 2 may be repeated as needed until the customer and merchant can agree on a price; the TID is present to indicate to the merchant that this is a repeated request.

The merchant stores the *PRD* for later use in delivering the goods, generates a new set of *RequestFlags* based on its response to the customer's *RequestFlags*, and generates a price quote and a TID (if one was not supplied in step 1) to identify this transaction in later steps. The TID is not globally unique; it is used only by the customer and merchant to maintain context between them.

The *ProductID* returned by the merchant in step 2 is a human-readable product description that will appear on the customer's NetBill statement.

3.3. The Goods Delivery Phase

Once the customer and merchant have negotiated a price for the goods in question, the customer directs the merchant to deliver the goods by supplying the TID that was used in the price request phase:

3.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(TID)
4.	M $\Rightarrow$ C	E_K(Goods), E_{CM}(CC(E_K(Goods)), EPOID)

The merchant generates a unique symmetric cipher key *K*, encrypts the goods using this key and sends the encrypted goods to the customer, along with a cryptographic checksum computed on the encrypted goods, so that the customer will immediately detect any discrepancy before proceeding. The merchant also sends an Electronic Payment Order ID, or *EPOID*, with the goods.

The EPOID is a globally unique identifier which will be used in the NetBill server's database to uniquely identify this transaction. It consists of three fields: a field identifying the merchant, a timestamp marking the time at the end of goods delivery, and a serial number to guarantee uniqueness.

The specification that the EPOID must be globally unique is used to prevent replay attacks, in which unscrupulous merchants reuse customers' old signed payment instructions. The timestamp portion of the EPOID is used to expire stale transactions; it must be generated at the end of goods delivery because the delivery (especially for very large goods) may take longer than the payment expiration time.

Because the goods are delivered encrypted in step 4, the customer cannot use them. The key *K* needed to decrypt the goods will be delivered in the payment phase, which follows.

3.4. The Payment Phase

After the encrypted goods are delivered, the customer submits payment to the merchant in the form of a signed Electronic Payment Order, or *EPO*. At any time before the signed EPO is submitted, a customer may abort the transaction and be in no danger of its being completed against her will. The submission of the signed EPO marks the "point of no return" for the customer.

An EPO consists of two sections, a clear part containing transaction information which is readable by the merchant and the NetBill server, and an encrypted part containing payment instructions which is readable only by the NetBill server. The clear part of the EPO includes:

- The customer's (possibly pseudonymous) identity
- The human-readable Product ID (from step 2)
- The negotiated price (from step 2)
- The merchant's identity
- The cryptographic checksum of the encrypted goods, to forestall debate over the content of the goods or whether they were received completely and correctly
- The cryptographic checksum of the Product Request Data (from step 1), to forestall debate over the details of the order
- The cryptographic checksum of the customer's account number with an account verification nonce, so that the merchant may verify that any supplied credentials (see Section 5.1) were used correctly
- The globally-unique EPOID

The encrypted part of the EPO includes:

- A ticket proving the customer's true identity
- Any required authorization tokens (see Section 5.2)
- The customer's NetBill account number
- The account verification nonce
- A customer memo field

The EPO is a tuple:

Identity,
ProductID,
Price,
M,
$CC(E_K(Goods))$,
CC(PRD),
CC(CAcct, AcctVN),
EPOID,
T_{CN}(TrueIdentity),
E_{CN}(Authorization,
  CAcct,
  AcctVN,
  CMemo)

Henceforth, we use the terminology *EPO* to denote this tuple.

After the customer presents the signed EPO to the merchant, the merchant endorses it and forwards the endorsed EPO to the NetBill server. The endorsed EPO adds the merchant's account number, the merchant's memo field, and the goods decryption key, as well as the merchant's signature:

$[[EPO]_C, MAcct, MMemo, K]_M$

At any time before the endorsed EPO is submitted to the NetBill server, the merchant may abort the transaction and be in no danger of its being completed against his will. The submission of the endorsed EPO marks the "point of no return" for the merchant.

The phase containing the submission and endorsement of the EPO is denoted:

5. $C \Longrightarrow M$ $T_{CM}(Identity), E_{CM}([EPO]_C)$
6. $M \Longrightarrow N$ $T_{MN}(M), E_{MN}([[EPO]_C, MAcct, MMemo, K]_M)$

Upon receipt of the signed and endorsed EPO, the NetBill server makes a decision about the transaction and returns the result to the merchant, who in turn forwards it to the customer.

The NetBill server makes its decision based on verification of the signatures, the privileges of the users involved, the customer's account balance, and the uniqueness and freshness of the EPOID. It then issues a receipt containing the result code, the identities of the parties, the price and description of the goods, the EPOID, and the key *K* needed to decrypt the goods. The receipt is digitally signed by the NetBill server, using the Digital Signature Algorithm (DSA). The receipt is denoted:

Result, Identity, Price, ProductID, M, K, EPOID

For this step, DSA is used rather than RSA because of its relative performance. While RSA signatures may be verified quickly, they are time-consuming to create; DSA signatures, on the other hand, may be created quickly. By using RSA for customer and merchant signatures and DSA for NetBill signatures, we may shift some computing load away from the NetBill server.

Some of the resulting burden on the merchant can be lifted by recognizing that, from a business risk perspective, it may be sufficient for a merchant to verify only a random sample of receipts signed by the NetBill server. Since integrity and authenticity are assured by the symmetric key encryption protocol, only accountability is at stake.

This receipt is returned to the merchant, along with an indication of the customer's new account balance (encrypted so that only she may read it). The EPO ID is repeated in the customer-specific data to ensure that the merchant cannot replay data from an earlier transaction.

7. $N \Rightarrow M$ $E_{MN}([\text{Receipt}]_{N\text{-}DSA},$
$E_{CN}(\text{EPOID, CAcct, Bal, Flags}))$

The *Flags* included in the customer-specific data indicate simple messages from the NetBill server to the customer; for example, that the account balance has reached a "low-water mark" and should be replenished soon.

8. $M \Rightarrow C$ $E_{CM}([\text{Receipt}]_{N\text{-}DSA},$
$E_{CN}(\text{EPOID, CAcct, Bal, Flags}))$

In step 8, the merchant responds to the request from the customer in step 5, forwarding the messages returned by the NetBill server in step 7.

3.5. Status Query Exchange

In the event of communications failure after step 5 of the protocol, the customer or merchant may be unaware of the transaction's status. (Before step 5, the transaction may be aborted with no difficulty, as no parties have yet signaled their commitment.) The system supports a status query exchange for delivery of this information.

The request and response proceed as one of the following exchanges, assuming the information is available. In each case, an alternate response is possible, indicating that the queried party does not have the requested information (possibly indicating why).

- The merchant requests the transaction status from the NetBill server:

1. $M \Rightarrow N$ $T_{MN}(M), E_{MN}(\text{EPOID})$
2. $N \Rightarrow M$ $E_{MN}([\text{Receipt}]_{N\text{-}DSA},$
$E_{CN}(\text{EPOID, CAcct, Bal, Flags}))$

- The customer requests the transaction status from the merchant:

1. $C \Rightarrow M$ $T_{CM}(\text{Identity}), E_{CM}(\text{EPOID})$
2. $M \Rightarrow C$ $E_{CM}([\text{Receipt}]_{N\text{-}DSA},$
$E_{CN}(\text{EPOID, CAcct, Bal, Flags}))$

- The customer requests the transaction status from the NetBill server:

1. $C \Rightarrow N$ $T_{CN}(\text{TrueIdentity}), E_{CN}(\text{EPOID})$
2. $N \Rightarrow C$ $E_{CN}([\text{Receipt}]_{N\text{-}DSA},$
$\text{EPOID, CAcct, Bal, Flags})$

- The customer requests the transaction status from the merchant for a Non-NetBill transaction (see Section 3.6):

1. $C \Rightarrow M$ $T_{CM}(\text{Identity}), E_{CM}(\text{EPOID})$
2. $M \Rightarrow C$ $E_{CM}(\text{Result, K})$

3.6. Handling Zero-Priced Goods

We anticipate that many NetBill transactions will be for *subscription goods*; i.e., goods for which the customer's marginal price is zero. With zero-priced goods, fraud is less important, and so we make several refinements to make our protocol more efficient in these cases.

First, we include a flag in the *RequestFlags* field of the price request (step 1) informing the merchant "If the price for this product is zero, treat this message as an automatic request for the goods."

Zero-priced transactions do not need to go through the NetBill server, as long as both parties agree. We can put another flag in the *RequestFlags* that informs the merchant "I require a NetBill receipt for this transaction." If this flag is present, the merchant must submit the transaction to the NetBill server, even if the price is zero. (The merchant may also decide to submit a zero-price transaction to the NetBill server.)

A customer or merchant may want to use the NetBill server on a zero-priced transaction if they require a signed receipt from a third party confirming the transaction. While subscription goods may not require a receipt, a merchant may decide to put a zero-priced purchase through NetBill in a "buy ten, get one free" situation so as to be able to prove that he actually provided the free item.

The merchant may change his price quote depending on this flag; if NetBill charges the merchant for billing services, the merchant may want to recover this cost if the customer requests a NetBill receipt for what might otherwise be a zero-priced transaction.

Combinations of these flags allow us to support four basic types of zero-price delivery:

3.6.1. Type I: Zero-Price Certified Delivery

This protocol eliminates the separate product request phase. Because steps 2 and 4 from the original protocol are combined, we indicate that by making steps 2 and 4 into a single step labeled 2/4.

1.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2/4.	M $\Rightarrow$ C	E_{CM}(ProductID, Price(=0), RequestFlags, TID), E_K(Goods), E_{CM}(CC(E_K(Goods)), EPOID)
5.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}([EPO]$_C$)
6.	M $\Rightarrow$ N	T_{MN}(M), E_{MN}([[EPO]$_C$, MAcct, MMemo, K]$_M$)
7.	N $\Rightarrow$ M	E_{MN}([Receipt]$_{N\text{-}DSA}$, E_{CN}(EPOID, CAcct, Bal, Flags))
8.	M $\Rightarrow$ C	E_{CM}([Receipt]$_{N\text{-}DSA}$, E_{CN}(EPOID, CAcct, Bal, Flags))

3.6.2. Type II: Certified Delivery without NetBill Server

This protocol improves on Type I by further eliminating the call to the NetBill server. With this modification, the payment phase becomes little more than an acknowledgment.

1.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2/4.	M $\Rightarrow$ C	E_{CM}(ProductID, Price(=0), RequestFlags, TID), E_K(Goods), ECM(CC(E_K(Goods)), EPOID)
5.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(EPOID, CC(E_K(Goods)))
8.	M $\Rightarrow$ C	E_{CM}(Result, K)

3.6.3. Type III: Verified Delivery

This protocol is nearly the same as the Type II protocol, except that the goods are encrypted in the shared session key *CM*. This bypasses the certified delivery mechanism, allowing the customer's software to begin streaming the goods to a viewer rather than being obliged to wait until all the goods have been delivered before receiving the key.

1.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2/4.	M $\Rightarrow$ C	E_{CM}(ProductID, Price(=0), RequestFlags, TID, Goods, CC(Goods), EPOID)
5.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(EPOID, CC(Goods))
8.	M $\Rightarrow$ C	E_{CM}(Result)

3.6.4. Type IV: Unverified Delivery

This protocol improves on Type III by eliminating the acknowledgment of goods delivery in the payment phase if the merchant does not need it.

1.	C $\Rightarrow$ M	T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2/4.	M $\Rightarrow$ C	E_{CM}(ProductID, Price(=0), RequestFlags, TID, Goods, CC(Goods))

4. Identities and Authentication

When a customer creates a NetBill account, she receives a unique User ID and generates the RSA public key pair associated with that User ID. This key pair is certified by NetBill, and is used for signatures and authentication within the system. (See [4] for a discussion of public key certification.)

Section 3.4 showed how these signatures will be used in the payment phase of the protocol. However, our protocol also uses Kerberos tickets. The NetBill transaction protocol involves several phases, for price negotiation, goods delivery, and payment; only the last of these phases requires nonrepudiable signatures. Instead of using public key cryptography for message authentication and encryption throughout the NetBill system, we use symmetric cryptography because it offers significant performance advantages.

We use the public key cryptography infrastructure to alleviate problems with traditional symmetric-key Kerberos (see [12]):

Kerberos uses a two-level ticket scheme; to authenticate oneself to a Kerberos service, one must obtain a *service ticket*, which establishes a shared symmetric session key between the client and server, and establishes that the Kerberos Ticket Granting Server believes the client's identity. To obtain a service ticket, a client must first obtain a *ticket-granting ticket* (or TGT), which proves the client's identity to the Ticket Granting Server. A client obtains a TGT via request from a Key Distribution Center, or KDC.

The Kerberos KDC/TGT arrangement introduces two significant problems that we may alleviate using public key cryptography. First, because it maintains a shared symmetric cipher key with every principal in the system, it is an attractive target for attack; recovering from compromise of the KDC requires establishing new shared keys with all users of the system. Second, a KDC and TGT will be a communications or processing bottleneck if a large number of users present a heavy traffic load.

To eliminate the Ticket Granting Server, we replace the TGT with a public key certificate, allowing each service to act as its own Ticket Granting Server. That is, a user presents a service ticket request encrypted with a certified public key (we call this a *Public Key-based TGT*, or *PKTGT*), and receives in response a symmetric-cipher-based service ticket. This service ticket is identical in form to a Kerberos service ticket. The Key Distribution Center is replaced by a key repository. The protocol for a customer to obtain a service ticket for a merchant *M* is as follows (before this step occurs, the customer requests the merchant's public key certificate over any available channel—such as an unsecured remote procedure call):

1. $C \Rightarrow M$ $[\{\text{Identity, M, Timestamp, K}\}^{M}]_{C}$
2. $M \Rightarrow C$ $E_K(T_{CM}(\text{Identity}), \text{CM})$

This model preserves the efficiency of symmetric ciphers for most communication and repeated authentication, and isolates the computational expense of public key cryptography to initial authentication between parties. We refer to this model as "Public Key Kerberos," or "PK Kerberos."

In the NetBill system, a customer obtains Kerberos tickets for the NetBill transaction server at the beginning of a session and obtains Kerberos tickets for merchants as she needs them. Merchant servers will continually maintain their own tickets for the NetBill transaction server.

4.1. Key Repository

Private keys are large, so users cannot be expected to remember them. Permanently storing private keys at a user's workstation poses security risks and restricts the user's electronic commerce activities to a single workstation. NetBill uses a key repository to optionally store customers' private keys. These keys are encrypted by a symmetric key derived from a passphrase known only to the customer.

4.1.1. Key Validation and Revocation Certificates

We use a public key certificate scheme (like that presented in [4]) to bind User IDs to keys, with NetBill as the certifying authority. NetBill generates a certificate when a customer first proves her identity and begins using NetBill.

However, allowing merchants, as services, to grant their own tickets based on these certificates poses a problem: NetBill is no longer involved in ticket-granting, and cannot prevent a ticket from being issued to a user with a compromised key. NetBill needs to invalidate compromised keys as quickly as possible.

NetBill maintains a Certificate Revocation List (CRL) at its server. When a key is compromised, the owner creates a Revocation Certificate and places it in the key repository along with her key. Any party can check that a given key has not been compromised by examining the revocation list.

Initially, it would seem that it is necessary for the customer and merchant to contact the server to check CRLs on each transaction. However, it is possible to eliminate this check by allowing the NetBill transaction server to do it when it processes the payment transaction. By delaying the CRL check to late in the protocol, we introduce some minor risks. Customers and merchants may disclose information such as their preference for particular items or special prices to bogus peers, but there is no financial risk.

4.2. Pseudonyms

Some customers want to disguise their identities. NetBill provides two pseudonym methods to protect the privacy of the customer's identity: a per-transaction method that uses a unique pseudonym for each transaction, and a per-merchant method that uses a unique pseudonym for each customer-merchant pair. (See [3] for a full discussion of privacy protection with pseudonyms.) The per-merchant pseudonym is useful for customers who wish to maintain a consistent pseudonymous identity to qualify for frequent-buyer discounts.

These pseudonym schemes are implemented by introducing a pseudonym-granting server, *P*, to create pseudonymous PKTGTs for the customer. The protocol for obtaining and using a pseudonymous PKTGT is as follows. The customer obtains the pseudonymous PKTGT in steps 1–2, and uses it with a merchant in steps 3–4 exactly as she would use a normal PKTGT:

1. $C \Longrightarrow P$ $\quad$ $[\{\text{TrueIdentity, M, Timestamp, K1, Type}\}^{P}]_{C}$
2. $P \Longrightarrow C$ $\quad$ $E_{K1}(\text{K2}, [\{\text{Pseudonym, M, Timestamp, K2}\}^{M}]_{P}, [\text{TrueIdentity, M, Pseudonym, Timestamp}]_{P})$
3. $C \Longrightarrow M$ $\quad$ $[\{\text{Pseudonym, M, Timestamp, K2}\}^{M}]_{P}$
4. $M \Longrightarrow C$ $\quad$ $E_{K2}(T_{CM}(\text{Pseudonym}), \text{CM})$

The protocol is the same for both kinds of pseudonyms; the desired type of pseudonym (per-merchant or per-transaction) is indicated in the *Type* field in step 1. The extra message *[TrueIdentity, M, Pseudonym, Timestamp]*$_P$ in step 2 is the customer's receipt proving that she was using the pseudonym *Pseudonym* with the named merchant at the time indicated. This may be useful to the customer in conjunction with the receipt received in step 8 of the transaction (which contains only the pseudonym) to later prove that she was involved in the transaction.

5. Credentials and Authorizations

In [7], Neuman presents a system of using *restricted proxies* for authorization. A restricted proxy is a ticket giving the bearer authority to perform certain operations named in the ticket. NetBill uses a similar construct to implement *credentials* to prove group membership (to allow merchants to provide discounts to special groups) and to implement access control mechanisms.

5.1. Credentials for Group Membership

An organization can provide a credential server which issues credential proxies proving membership in a group. In this case, the credential server is asserting a fact (membership in a group) about which it is authoritative. For example, an auto club may provide a credential server which issues credentials to the members of the club; merchants who offer discounts to the club's members will accept these credentials as proof of membership. The protocol for obtaining a credential (assuming the customer has already obtained a service ticket for the credential server) from a credential server, *G*, is as follows:

1. $C \Longrightarrow G$ $\quad$ $T_{CG}(\text{Identity}), E_{CG}(\text{Group, CAcct})$
2. $G \Longrightarrow C$ $\quad$ $E_{CG}([\text{Group, Detail, Identity, CC(CAcct, AcctVN), Timestamp}]_{G}, \text{AcctVN})$

Credentials obtained in this manner are presented to merchants in the price request phase of the transaction protocol, step 1.

A credential issued to a customer may be unrestricted, or it may optionally be restricted for use on a specific account (for example, in order to prevent corporate employees from taking advantage of corporate discounts for personal purchases). This is accomplished by passing the account number to the group server as part of the request. If the account number is appropriate for this group, the credential will be issued. The credential contains a cryptographic checksum of the account number and an "Account Verification Nonce," which is also returned to the customer along with the credential.

This nonce is a pseudorandom number ensuring that merchants can neither determine which different customers (or the same customer in repeated sessions) are using the same account nor easily verify guesses of the customer's account number. The nonce is passed along to the NetBill server in the encrypted part of the EPO so that the NetBill server can verify that the checksum passed to the merchant (for his comparison to the credential) corresponds to the account number actually being used.

The *Detail* field allows a credential server to include additional information in a format specific to the credential server. This would allow, for example, a multiple-journal subscription credential server to issue a single credential for all subscribers, using the *Detail* field to specify which journal subscriptions the customer holds.

Credentials can also be used by cooperating merchants to restrict information access. In this way, merchants only sell to approved customers: those who can present a certain credential. This offers a solution for merchants who, for example, can restrict distribution of sensitive documents only to individuals whose credentials verify a need-to-know.

5.2. Access Control Mechanism

As noted in [7], proxies can implement access control. An account owner (such as a parent) may have a restriction on the account such that no purchases can be completed by a given customer (such as a child) without approval from an access control server. This allows different organizations to provide access control services. For example, both the PTA and a church group could offer competing access control services.

To obtain an access control authorization, a customer *C* must present details of a specific transaction to the access control server *A*, who grants a single-use proxy authorizing the given transaction. The protocol is as follows:

1. $C \Rightarrow A$ T_{CA}(Identity), E_{CA}(M, ProductID, Price, CC(E_K(Goods)), EPOID, CAcct)
2. $A \Rightarrow C$ E_{CA}($E_{A\text{-}PRI}$(CC(Identity, M, ProductID, Price, CC(E_K(Goods), EPOID, CAcct)))

The item returned in step 2 is the *Authorization* item used in step 5 of the transaction protocol (see Section 3.4).

6. Complaints and Failure Analysis

The NetBill protocols are robust against failures, and retain essential information to protect customers and merchants against fraud. Our system can respond to complaints made by either the customer or the merchant. In this section, we examine those complaints and discuss how they are handled. First, we look at potential customer complaints, and then at potential merchant complaints.

6.1. Customer Complaints

6.1.1. Incorrect or Damaged Goods

- “This isn’t the product I specified.”
- “The goods arrived broken or incomplete.”
- “The decryption key I was given was wrong.”

In the event that the decrypted goods do not match the product description as given by the merchant, the dispute must be brought to the attention of a human arbitrator, who will determine the validity of the customer’s complaint and, if appropriate, direct the merchant to provide a refund.

The arbitrator uses the registered copies at NetBill of the customer’s signed EPO containing a cryptographic checksum of the encrypted goods, and the merchant’s signed endorsement indicating his agreement with that cryptographic checksum and attesting to the decryption key. The arbitrator compares these registered values against the copy of the encrypted goods and decryption key provided by the customer in her complaint. The arbitrator can easily determine whether the purported problem with the goods is the fault of the merchant or an error by the customer.

- “The goods are not as advertised.”

The protocol can be used to demonstrate whether the goods delivered are the goods ordered, as shown above. However, if the customer was induced to buy the goods by false advertising claims, this protocol provides no help. The customer must lodge a complaint with the Federal Trade Commission or other appropriate agency. It is important for billing servers to monitor these charges and assist with their resolution.

- “I bought this but never got the decryption key.”

This complaint may be answered by directing the customer to perform a status query (see Section 3.5) to retrieve the key. In the event that the decryption key does not yield a satisfactory decryption, the dispute will change to one of the other complaints listed.

6.1.2. Transaction Disputes

- “I agreed to pay $X, but was charged $Y instead.”
- “I’ve only bought $X worth of goods, but my balance has gone down by $Y.”

Because the NetBill server has a signed EPO from the customer, it can prove that the customer approved the purchase(s) for $Y. In the event that the NetBill server cannot provide the signed EPO(s), the customer's money is refunded. This protects customers against fraud by the operators of the NetBill server.

- "I never bought this, but it appears on my statement."
- "I told the merchant no, but he put it through anyway."

Because the NetBill server has signed EPOs from the customer, it can prove that the customer approved the purchases. In the event that the NetBill server cannot provide the signed EPOs, the customer's money is refunded.

- "You told me this transaction didn't go through, but I got charged anyway."

Because the NetBill server provides signed receipts even for failed transactions, the customer can present these receipts to prove that the transactions were declined. If the customer cannot produce these receipts and the NetBill server claims to have approved the transactions, it must provide the decryption keys for the information goods (via status query exchange).

6.2. Merchant Complaints

6.2.1. Insufficient Payment

- "I sold $X worth of goods but only received $Y."
- "You told me this transaction went through, but I never got paid for it."

In all transactions, the NetBill server provides a signed receipt indicating the success or failure of a transaction. In the event that a merchant is not properly credited, he can prove the error by presenting these signed receipts.

7. Conclusion

This paper has presented the NetBill protocols. These protocols have introduced new methods for certified delivery, access control, user certificates, pseudonyms, and their integration. These protocols are designed to provide very high degrees of security and flexibility while still providing good efficiency. However, this paper does not represent final work; it is a snapshot of our current design. We plan to test our design in a major wide-scale pre-commercial beta test of the NetBill system beginning in 1996.

The NetBill project is committed to open protocols. We are eager to work with others to make our protocols as widely applicable and interoperable as possible, and welcome comments.

For more information on the current state of the NetBill project, we invite readers to consult our WWW page at `http://www.ini.cmu.edu/netbill/`.

Acknowledgements

We received valuable contributions in technical discussions of the protocol from Thomas Wagner.

References

[1] Alireza Bahreman and J.D. Tygar. "Certified Electronic Mail." In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 3–19, San Diego, CA, February 1994.

[2] M. Bellare, *et al*. *iKP Family of Secure Electronic Payment Protocols*. `http://www.zurich.ibm.com/Technology/security/extern/ecommerce`

[3] Benjamin Cox. *Maintaining Privacy in Electronic Transactions.* Information Networking Institute Technical Report TR 1994–8, Fall 1994.

[4] Stephen Kent. *RFC 1422: Privacy Enhancement for Electronic Mail: Part II: Certificate-Based Key Management*. Internet Activities Board Request For Comments 1422, February 1993.

[5] National Institute of Standards and Technology. *FIPS 180: Federal Information Processing Standard: Secure Hash Standard (SHS).* April 1993.

[6] National Institute of Standards and Technology. *FIPS 186: Federal Information Processing Standard: Digital Signature Standard (DSS)*. May 1994.

[7] B. Clifford Neuman. "Proxy-Based Authorization and Accounting for Distributed Systems." In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283–291, May 1993.

[8] R. Rivest, A. Shamir, L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." In *Communications of the ACM,* 21(2), February 1978.

[9] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York: John Wiley & Sons, 1994.

[10] Marvin Sirbu and J.D. Tygar. "NetBill: An Internet Commerce System Optimized for Network Delivered Services." In *IEEE Personal Communications*, pages 6–11, August 1995.

[11] Alexander Somogyi, Thomas Wagner, *et al.* *NetBill*. Information Networking Institute Technical Report TR 1994–11, Fall 1994.

[12] Jennifer G. Steiner, B. Clifford Neuman and Jeffrey I. Schiller. "Kerberos: An Authentication Service for Open Network Systems." In *USENIX Winter Conference*, pages 191–202, February 1988.

[13] J. D. Tygar. "Atomicity in Electronic Commerce" (invited paper), to appear in *ACM/IEEE 21st Conference on Principles of Distributed Computation*, 1996.