Atomicity in Electronic Commerce

J. D. Tygar

January 1996
CMU-CS-96-112

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was partially supported by the Department of Defense (ARPA contracts F33615-C-1465, F19628-93-C-0913), IBM, the Information Networking Institute, Motorola, National Science Foundation (under Presidential Young Investigator Grant CCR-8858087 and Cooperative Agreement No. IRI-9411299), TRW, the US Postal Service, and Visa International. Appendix A is drawn from a report that was additionally supported by ARPA contract F19628-95-C-0018.

Keywords: electronic commerce, atomicity, NetBill, IBIP, cryptography, transaction processing, ACID, franking, electronic currency, mail, postage, stamps, electronic stamps, secure coprocessors, signatures

Abstract

This paper will appear as an invited paper in the May 1996 proceedings of the ACM Symposium on Principles of Distributed Computing.

There is a tremendous demand for the ability to electronically buy and sell goods over networks. Electronic commerce has inspired a large variety of work. Unfortunately, much of that work ignores traditional transaction processing concerns — chiefly atomicity. This paper discusses the role of atomicity in electronic commerce. It then briefly surveys some major types of electronic commerce pointing out flaws in atomicity. We pay special attention to the atomicity problems of proposals for digital cash.

The paper presents two examples of highly atomic electronic commerce systems: NetBill and Cryptographic Postage Indicia.

Atomicity in Electronic Commerce

J. D. Tygar
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891 USA
tygar@cs.cmu.edu

Abstract

There is tremendous demand for the ability to be able to electronically buy and sell goods over networks. This field is called electronic commerce, and it has inspired a large variety of work. Unfortunately, much of that work ignores traditional transaction processing concerns — chiefly atomicity. This paper discusses the role of atomicity in electronic commerce. It then briefly surveys some major types of electronic commerce pointing out flaws in atomicity. We pay special attention to the atomicity problems of proposals for digital cash.

The paper present two examples of highly atomic electronic cash systems: NetBill and Cryptographic Postage Indicia.

1. Electronic Commerce

If you regularly use the World Wide Web, you have probably noticed that much of the information on it is worth what you pay for it. To improve the quality of available electronic information, we must create mechanisms to conveniently compensate the creators and owners of network information. If we want to put the Library of Congress on-line, we will have to first find a way to compensate copyright owners.

Electronic commerce is an attempt to address these problems. The idea is to build mechanisms that make it simple to buy and sell goods on-line. These mechanisms have attracted wide interest in the last year. Besides enabling a new type of commerce, they appear to offer a variety of benefits, including increasing the range of information easily available to most people, making automatic search and retrieval of that information easy, and reducing costs by simplifying or eliminating human involvement in processing and fulfilling orders.

Here is one measure of the excitement over electronic commerce: the 12 June 1995 issue of *Business Week* includes the following projection of the role of electronic commerce. This projection is probably overly optimistic, but it indicates that electronic commerce is being taken seriously in some quarters.

Year	Traditional Commerce (billion $)	Electronic Commerce (billion $)
1994	5150	245
2000	8500	1650
2005	12000	2950

Here is another measure: in 1994, J. C. Penney, a well-known American retailer with a reputation for not being especially high-tech, sold directly to customers $17 million worth of goods over computer networks

I gratefully acknowledge support from various sources for this work: Department of Defense (ARPA contracts F33615-90-C-1465, F19628-93-C-0193), IBM, the Information Networking Institute, Motorola, National Science Foundation (under Presidential Young Investigator Grant CCR-8858087 and Cooperative Agreement No. IRI-9411299), TRW, the US Postal Service, and Visa International. Appendix A is drawn from a report that was additionally supported by ARPA contract F19628-95-C-0018. The views and conclusions contained in this document are those of the author and should not be interpreted as reflecting the official policies, either expressed or implied, of the Advanced Research Projects Agency, the National Science Foundation, the U.S. Government or any part thereof, or any other research sponsor.

(including both Internet and private services such as Compuserve, America On Line, Prodigy, etc.)

For yet another measure, visit the WWW site `http://www.yahoo.com`, and see the tens of thousands of electronic storefronts available.

There are many efforts to build electronic commerce systems. Here are some representative organizations making major electronic commerce efforts (by no means comprehensive):

- CMU (NetBill)
- Cybercash
- Digicash
- DEC (Millicent)
- First Virtual
- FSTC (E-check)
- IBM (iKP)
- Mastercard (SEPP)
- Microsoft and Visa (STT)
- Open Market
- Netscape (SSL)
- US Postal Service

(This list is very fluid. For example, at the time that this paper is being written [December 1995] there are discussions about developing a merged protocol from STT and SEPP; statements that I make about the systems being developed today are likely to become outdated as the systems evolve. Any bibliographic listing of references is will rapidly become dated, but [30] is a nice summary of most of this work.)

Concepts from the PODC community are used heavily in electronic commerce. Concepts that need to be mastered to properly handle electronic commerce include:

- atomic transactions
- cryptographically secure protocols
- secure computation
- safe voting
- high reliability

This paper is concerned with the first property, atomic transactions. I will discuss a variety of types of electronic commerce. After giving a discussion on atomicity, I will consider the atomic properties of several electronic commerce protocols. I will then discuss the development of two highly atomic protocols: the NetBill protocol and cryptographic postage indicia.

Because this paper is based on an expository lecture, my tone throughout is informal. I am afraid that you will not find formal definitions of types of electronic commerce atomicity below; indeed, I consider the formulation of those formal definitions an open problem (see Section 6). For those who crave more details presented in a more formal manner, Section 8 (Appendix A) contains a technical exposition of the NetBill protocol; [10] is the best reference for a formal exposition of cryptographic postage indicia.

Note that throughout the text I use male pronouns to refer to merchants and female pronouns to refer to customers.

2. Electronic Commerce Properties

How can we characterize electronic commerce protocols. There are a variety of properties that we can describe; this paper focuses on atomicity but, because properties often interact in a variety of non-trivial ways, it important to review several properties.

2.1. Atomicity

Atomicity allows us to logically link multiple operations so that either all of them are executed or none of them are. For example, in transaction processing, one may execute a sequence of code as follows:

```
<begin-transaction>
state-changing operation 1;
state-changing operation 2;
. . .
state-changing operation n;
<end-transaction>
```

When this block of operations is executed, all of the state-changing operations from 1 to *n,* inclusive will be executed or the state of the system will be as if none of them had been executed.

Why would atomicity every fail to occur? Well, if the transactions are being executed in a distributed environment on multiple processors, then one of the processors executing a state-changing operation or communication between two processors executing state-changing operations may fail. In that case, it may be impossible to complete the entire block of state-changing operations. In these cases, it is necessary to roll-back the processors to a state consistent with the transaction have never been begun in the first place.

Atomic transactions form the cornerstone of modern transaction processing theory. (Nancy Lynch and her fellow researchers have written an encyclopedic book about atomic transactions [13]; a tremendous resource for those implementing atomic transaction processing systems is the standard textbook [9]; for a thorough review of powerful roll-back methods in the context of computer security and electronic commerce, see [25], [26], and [27].) The "A" in *ACID Transactions* stands for "atomic", no non-atomic distributed transaction system would ever be tolerated by customers of data processing.

However, as we shall see below, the story is quite different in the world of electronic commerce protocols. Most of the proposed protocols are not atomic. For example, if I interrupt a communication between a merchant and a customer, I can often throw an electronic commerce protocol, into an ambiguous state. Money or electronic cash tokens may be copied (with different parties each believing that it has the true, valid copy) or destroyed.

I define three levels of atomicity to protect electronic commerce protocols.

2.1.1. Money Atomicity

Money atomic protocols effect the transfer of funds from one party to another without the possibility of the creation or destruction of money.

For example, a cash transaction is usually money atomic (unless the possibility exists of counterfeiting or destruction of money).

This is a basic level of atomicity that each electronic commerce protocol should satisfy.

2.1.2. Goods Atomicity

Goods-atomic protocols are money atomic, and also effect an exact transfer of goods for money. That is, if I buy a good using a goods-atomic protocol, I will receive the good if and only if the money is transferred. For network protocols, goods atomicity is especially important for information goods. There must be no possibility that I can pay without getting the goods, or get the goods without paying. (Anyone who has had an interrupted file transfer while downloading information on the Internet is aware of the importance of goods atomicity.)

For example, a cash-on-delivery parcel delivery is a good real-world approximation to an electronic commerce protocol. I get the parcel exactly when I have paid the delivery agent.

Goods atomicity is an important property that each electronic commerce protocol intended for information goods should satisfy.

2.1.3. Certified Delivery

Certified delivery protocols are money atomic and goods atomic protocols that also allow both a merchant and a customer to prove exactly which goods were delivered. If I buy a document entitled "How to make a million dollars fast on the Internet" and receive an electronic copy of some unrelated or garbage material, I will want to complain to an authority. To rapidly resolve the question, both the merchant and the customer will want to be able to prove the exact contents of what was delivered.

For example, a certified delivery protocol corresponds to a cash-on-delivery parcel delivery where the contents of the parcel is opened in front of a trusted third-party who immediately records in an indestructible form the exact contents of the parcel.

Certified delivery protocols are helpful for scenarios where merchants and customers may be untrusted. Today, there is no effective way to distinguish a large trusted WWW merchant from a fly-by-night impressive electronic storefront that actually connects to a shop that contains a fraudulent operation.

2.2. Anonymity

Some people want to keep their purchases private. They do not want to have third-parties (or even merchants) know their identity. This concern may arise because the customer is buying a good of questionable social value (e.g., pornography); or because the customer does not want to have his name added to a marketing or mailing list; or for illegal reasons (e.g., to evade taxes); or simply because the customer personally values privacy.

Although most paper money contains serial numbers, cash transactions can often have anonymous properties. Serial numbers are rarely traced and recorded, and if I buy something from a merchant who does not know me or from a vending machine, my purchase is often effectively anonymous.

David Chaum has been the most influential advocate of anonymous electronic commerce protocols. He has written a number of highly influential papers on topics such as "anonymous digital cash", these in turn have inspired all electronic commerce researchers. Modern researchers have improved his protocols; a representative sophisticated example of the current version of his protocols can be found in [4].

Here is the way these protocols work:

a) a customer withdraws money from the bank, receiving a cryptographic token which can be used as money;

b) the customer applies a cryptographic transformation to the money that still allows a merchant to checks its validity, but make it impossible to trace the customer's identity;

c) the customer spends the money with the merchant. (in doing so, she applies a further cryptographic transformation so that the merchant's identity is used in the data);

d) the merchant checks to make sure that he has not received the token previously;

e) the merchant sends the goods to the customer;

f) at a later point, the merchant deposits his electronic tokens at the bank; and

g) the bank checks the tokens for uniqueness; the identities of the customers remain anonymous except in the case when a customer had double-spent a token—if a token was double-spent, the identity of the customer is revealed and the network police are notified of attempted counterfeiting.

Now consider when a communication failure happens around step (c). The customer has no way of knowing if a merchant has received her token or not. The customer has two options:

- The customer can return her electronic token to the bank (or spend it on a different merchant.) If she does this, and the merchant actually received her token, then when the merchant cashes in the token, the customer's anonymity will be revealed. Even worse, the customer will be likely to be accused of fraud.
- The customer can take no action, failing to return her token. If she does this, and the merchant never received her token, then she is in danger of losing her money. She will have never received the good she attempted to purchase, and she will be unable to use her money.

In either case, money atomicity breaks down.

In many countries, most anonymous transactions are illegal. For example, in the United States, the Money Laundering Act (12 USC §1829) requires that electronic commerce systems should both

- promptly report any transaction over $3000 (this is the figure effective January 1, 1996; prior to that, the limit was $10,000)
- store a copy of any transaction over $100.

These requirements have not been tested in court for digital cash systems. However, it is clear that it is arguable that, as currently proposed, digital cash systems may be illegal.

I also note that it is often possible to achieve a limited form of anonymity by having a proxy agent complete purchases for the customer. In this case, the transaction may be easily traced to the proxy agent, which keeps private the identity of the true customer.

2.3. Security

Can we trust anyone in cyberspace? Communications can be easily intercepted, messages can be inserted, and the absolute identity of other parties may be uncertain. Clearly, security will be important for any electronic commerce protocol.

By contemporary standards, the current form of credit cards, which reveal a customer's identity and charge numbers to a merchant and to anyone who can obtain a copy of the receipt, would be unlikely to be accepted if they were introduced newly today.

Many electronic commerce systems depend on some ultimate, trusted authority. For example, NetBill depends on the trustworthiness of a central server. However, even in the case where one uses a trusted server, one can minimize the effects of the security failures of that server. For example, in NetBill, detailed cryptographically-unforgeable records are kept so that if the central server was ever corrupted, it would be possible to unwind all corrupted actions and restore any lost money.

2.4. Transaction size

The average credit card transaction has typically been estimated to be on the order of $50. Depending on the arrangements made with a bank, a merchant payd approximately 30¢ plus 2% of the purchase price for each and every transaction. For many telephone or mail order businesses, the actual rate is closer to 50¢ plus 2.25%.

If one is engaging in a transaction that is only worth 10¢ or even 1¢, the standard credit card rates would dominate the cost of the item. Thus, a number of parties have proposed support for *microtransactions* or transactions less than $1. (By no means is 1¢ the minimum transaction value of interest; Mark Manasse's electronic commerce system is named Millicent [14].)

Both NetBill and cryptographic postage indicia are motivated by the idea of supporting microtransactions. Some of the design decisions made for those systems can only be understood by the microtransaction requirement. However, a detailed discussion of microtransactions is beyond the scope of this paper.

(For those who are curious: the key idea behind most microtransaction protocols is to aggregate many small transactions charged using specially optimized protocols; then charge the aggregated total as a large value transaction. This idea is a beautiful application of protocol nesting. For a discussion of microtransactions in NetBill, see [23]; for a completely different approach, see [14].)

3. Non-atomic Electronic Commerce Protocols

3.1. Digicash

Digicash uses an anonymous digital cash protocol. As discussed in Section 2.2, digital cash protocols are not money atomic; indeed, in the event of communication failure, they can fail to be anonymous as well. Finally, digital cash protocols use several rather computationally intensive cryptographic operations and are thus quite expensive. I estimate that the real cost of processing a single digital cash transaction would be on the order of $1 per transaction; Digicash reportedly has relatively high fees, suggesting that this expectation is correct. Digicash in its current form is not useful for microtransactions.

3.2. First Virtual

First Virtual allows users to freely buy goods. First Virtual then uses e-mail to confirm each and every transaction with the customer. Setting aside the acceptability of flooding a user with e-mail for purchases in this way, this model clearly preserves money atomicity, although it clearly fails goods atomicity (since the customer can buy an item without paying for it.) First Virtual apparently considers goods atomicity to be relatively unimportant. (Indeed, First Virtual takes a dim view of communications security and encryption in any form; in [3], they argue that communications security is "irrelevant" and they dismiss electronic commerce designers who postpone deployment of their systems to perfect strong security guarantees.)

First Virtual's system can easily be a target of fraud and atomicity failures. It is somewhat better than digital cash, but inferior to other electronic commerce systems.

Ultimately, First Virtual translates each electronic commerce transaction into a credit card transaction, making First Virtual in its current form of limited value for microtransactions. (First Virtual suggests using aggregation, but they can not aggregate across different merchants in a single credit card transaction.)

3.3. SSL

The Secure Socket Layer (SSL) approach sets up a secure communication channel (using cryptography) to transfer a customer's credit card number to the merchant. This approach is equivalent to reading your credit card number over the phone to a merchant using a secure telephone connection.

This approach offers money atomicity to the extent that credit card transactions are money atomic. However, its security properties are less clear; for example, since a (potentially unscrupulous) merchant has the customer's credit card number, he can use it to commit fraud. (Merchant fraud is one of the most serious problems facing the credit card industry [31]. Lyndon LaRouche is a well-known example of a person who committed merchant credit card fraud.)

Goods atomicity is not addressed by SSL.

In its current form SSL is clearly of limited value for microtransactions.

3.4. STT/SEPP/iKP

STT (Visa/Microsoft), SEPP (Mastercard), and the iKP family of protocols (IBM) are examples of a secure credit-card based protocols. Each of these protocols are slightly different, but they have a common structure: the customer digitally signs (using, for example, public key cryptography) a purchase request together with a price and then encrypts the request in a bank's public key. Similarly, the merchant prepares and submits a sales request with a price for the bank. The bank compares the purchase and sales request — if the prices match, the bank charges the customer's account and instructs the merchant to complete the sale.

Like SSL, this approach offers money atomicity to the extent that credit card transactions are money atomic. However, the security properties of STT *et al* are superior since they prevent merchant fraud.

Goods atomicity is not addressed by STT *et al*.

In their current form STT *et al* are clearly of limited value for microtransactions.

4. NetBill

My co-researchers and I developed NetBill to provide all three levels of atomic transactions. Here, I give a broad sketch of the NetBill format and some rough arguments of why it satisfies all three atomicity conditions: money atomicity, goods atomicity, and certified delivery. However, to keep my explanation simple, I do not cover the details of the protocol, leaving that for Appendix A (Section 8). For example, I do not discuss here how NetBill protects against message replay, communication security, or various timing attacks.

The NetBill protocol is between three parties: a customer, a merchant, and the NetBill server. Think of a NetBill account held by a customer as equivalent to a virtual electronic credit card account.

Here is the outline of the NetBill protocol

a) The customer requests a price from the merchant for some goods. (This step is necessary because the price of a good may depend on the identity of the customer; for example, a student ACM member may qualify for a discount on some items)

b) The merchant makes an offer to the customer

c) The customer tells the merchant that she accepts the offer.

d) The merchant sends the information goods requested encrypted by key *K*.

e) The customer prepares an electronic purchase order (EPO) containing a digitally signed value for: <price, cryptographic-checksum of encrypted goods, time-out>. The customer sends the signed EPO to the merchant

f) The merchant countersigns the EPO. The merchant also signs the value of *K*. The merchant sends both values to the NetBill server.

g) The NetBill server checks the signature and counter-signature on the EPO. It then checks the customer's account to ensure that sufficient funds exist to approve the transaction, and also checks that the time-out value in the EPO has not expired. Assuming that all is OK, the NetBill server transfers price funds from the customer's account to the merchant's account. It stores *K,* and the cryptographic-checksum of the encrypted goods. It then prepares a signed receipt that includes the value *K*. It sends this receipt to the merchant.

h) The merchant records the receipt, and forwards it to the customer (who can then decrypt her encrypted goods.)

This protocol thus transfers an encrypted copy of the information goods, and records the decryption key in escrow at the NetBill server. Now let us see how this protocol provides various types of atomicity protection.

Money atomicity: all funds transfers occur at the NetBill server, and since the NetBill server uses a local atomic database to store fund values, no money can be created or destroyed.

Goods atomicity: if the protocol fails as a result of communications failure or processor failure before the NetBill server atomically processes the transaction in step (g), then no money changes hands, and the customer never receives the decryption key — he gains no access to the encrypted information goods. On the other hand, if step (g) succeeds, then both the merchant and NetBill server will record the value of *K*. Normally, these values would be forwarded back to the customer as a result of step (h), but if something goes wrong, the customer can obtain *K* from either the merchant or NetBill server at any time.

Certified delivery: since we have goods atomicity, we know that the customer received something in exchange for money. Now, suppose that the customer claims that he receives goods different from what she ordered. Then, since NetBill server has a cryptographic checksum of the encrypted goods that is countersigned by both the customer and the merchant, the customer can present her encrypted goods to a judge and verify that she has not tampered with the goods. Now, since a merchant-signed value of *K* is stored at both the customer and the merchant, the judge can decrypt the goods and determine whether the goods were as advertised as not.

Thus NetBill presents an example of a highly-atomic electronic commerce protocol. We have currently built an alpha version of NetBill at Carnegie Mellon (in conjunction with our development and operations partners, Mellon Bank and Visa International), and we hope to prove that NetBill is not only highly-atomic but that it has the performance, scalability, and efficiency to handle a large number of microtransactions

5. Cryptographic Postage Indicia

Is it possible to achieve money atomicity without using a central server? Yes, one way to do this is to use secure hardware. For example, FIPS 140-1 [16] specifies support for tamper-proof and tamper-resistant devices that can store information and perform processing tasks. These devices are secure in the sense that any attempt to penetrate them will result in erasure of all information stored inside them. We could use this to store an electronic wallet; when a charge is made, the electronic wallet withdraws funds.

We call these tamper-proof devices *secure-coprocessors*.

Now the design of such a system is not easy [32], and there are quite a few risks associated with customer approval of transactions [8]. However, with careful design it can be made to work.

My research group has been working with the US Postal Service to develop standards for PC-generated laser printed indicia for postage meters. These are

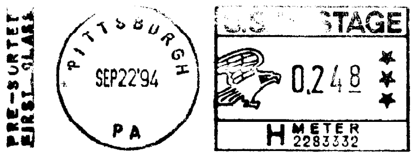

Figure 1: Traditional indicia are easy to copy.

designed to meet the needs of the Postal Service Information-Based Indicia Program [29].

As Figure 3 shows, it is trivial to copy traditional indicia using a scanner and a computer. It is equally easy to forge dates and postage values on counterfeited indicia. (Note: if you ever decide to take up the life a criminal and forge indicia, make sure to add smudges to the indicia —indicia that are reproduced too clearly can easily be recognized as forged.)

Using a secure coprocessor, it is easy to store an account balance for postal customers. This account balance is decremented whenever postage is printed. Now, the secure coprocessor prepares a cryptographically signed message that contains envelope data (sender address, receiver address, date sent, and sequence number). This information is then printed on the envelope using an efficient data representation such as PDF-417 [11]. Figure 3 shows Lincoln's Gettysburg address encoded in PDF 417. PDF 417 normally encodes 400 bytes per square inch.

When mail is received at a postal sorting facility, the data block is checked to see if they match the address used for sorting, and to verify the uniqueness of the sequence number. (Note that all mail to given address will be processed by a single sorting station.) Indicia remain valid for six months. (The US Postal Service claims to deliver more than 90% of all first class mail within three days of being sent and more than 99% in seven days. Thus, six months would appear to be a generous bound for mail delivery.) The database stored at a local sorting station can regularly be purged of entries with a date older than six months

Figure 2: PDF 417 encoding of Abraham Lincoln's Gettysburg Address.

If an adversary attempts to break money atomicity by forging indicia, he must do one of two things:

- copy existing indicia, which then will only be valid for the encrypted delivery address, and will be caught at the sorting station; or
- attempt to find the value used to digitally sign the cryptographic indicia, which will require opening the secure coprocessor, erasing all the vulnerable data within.

For a more technical exposition on secure coprocessors, see [10], [32], and [33].

6. Open Problems

The field of electronic commerce has many open problems. Here are some of my favorites:

- What is the relationship between atomicity and anonymity? Can they be mutually compatible?
- What is the relationship between atomicity and security? Can they be separated?
- What other atomicity models exist in electronic commerce (besides money atomicity, goods atomicity, and certified delivery)? Is there a general schema?
- What is the minimum number of message exchanges necessary in an atomic purchase?
- What atomic electronic commerce mechanisms can be built for multiple banks or billing servers?
- Can atomicity be used for continuously delivered information (such as continual stock market updates) or very large objects (such as video programs)?
- Can we give a formal definition for atomicity?
- How can we prove that a protocol is atomic?
- Is it possible to express atomic properties in terms of model checking?
- Can we extend electronic commerce models to auctions? Can we make them efficient and fair?
- Can we extend electronic commerce models to auction markets, such as stock markets?

- Can we protect redistributed information or reselling of information? (This is the so-called superdistribution of Mori and Kawahara [15].)
- Can we devise effective *digital watermarks* that clearly indicate the purchaser of illegally pirated or redistributed information?
- How can we represent and enforce electronic contracts governing the use, distribution, and payment conditions for information goods and software?
- Can we make a fault-tolerant version of electronic commerce protocols that remain stable even when banks fail? (The results of T. Rabin and Ben-Or [20] seem to be appropriate here.)
- Can we build systems to allow anonymous charitable contributions? Can we extend them to allow documentation so that one can take a tax credit?
- What is the minimum level microtransaction that can be supported in electronic commerce? The minimum level atomic microtransaction?
- We can express money as tokens or as entries in a server (see [5]) — is there anyway to express a formal equivalence between these two methods?

7. Acknowledgments and Further Sources of Information

Bennet Yee has been my primary collaborator for electronic commerce work. All of the work described regarding secure coprocessors and cryptographic postage indicia is joint work with Bennet. Bennet and I jointly observed that Chaum-like digital cash protocols fail to work properly if communications are interrupted, prompting this line of work. Bennet is always a pleasure to work with; I owe him a tremendous intellectual debt. Portions of our work previously appeared in [32] and [33].

Nevin Heintze contributed to the later development of cryptographic postage indicia as represented in [10].

Ali Bahreman began thinking about certified delivery mechanisms while working on his master's thesis at CMU in the context of certified electronic mail; together, we came up with several initial protocols described in [1].

Ben Cox, Tom Wagner, and I first thought of building on Ali's protocols for a practical electronic commerce protocol in the course of Ben and Tom's brilliant project in my distributed systems class in Spring 1994. Their project introduced the notion of certified delivery electronic commerce protocols. At the same time, Carnegie Mellon's Information Networking Institute, under the direction of Marvin Sirbu, had been pursuing an "internet billing server." All parties quickly recognized the superiority of the protocol developed by Ben, Tom, and I; and that protocol became the basis of the newly named NetBill project.

Marvin Sirbu and I then became the principal investigators in building the NetBill system. Our basic presentation of NetBill's properties is contained in [23]. Marvin and I, together with a joint student Jean Camp, made a preliminary distinction between money atomicity and goods atomicity in [5], although that paper is flawed and did not properly distinguish between certified delivery and goods atomicity.

Michael Rabin gave me many helpful suggestions throughout this project. All of his advice and ideas were important, but I especially appreciated his enthusiastic encouragement during the early stages of this research.

Maurice Herlihy and Mark Tuttle gave me useful feedback on a preliminary version of the presentation that inspired this paper.

Sean Smith and David Johnson have had many important conversations with me on the topic of atomic transactions and rollback.

Students and colleagues at CMU have given me a number of specific useful suggestions for this research. In additions to those named above, I would like to thank Thomas Alexandre, Brad Chen, Howard Gobioff, Mike Harkavy, Mahadev Satyanarayanan, Alfred Spector, Jiawen Su, Jeannette Wing, and Hao-Chi Wong.

More information on NetBill can be found at `http://www.ini.cmu.edu/netbill/`.

More information on cryptographic postage indicia and secure coprocessors can be found at `http://www.cs.cmu.edu/afs/cs/project/dyad/www/`.

8. Appendix A

My description above of the NetBill protocol was, as noted, only an informal description. Below I present a more formal description of the protocol, excerpted and revised from [7]. This is a revision of material originally written by me together with Benjamin Cox and Marvin Sirbu, and I gratefully thank them for permission to reprint this material here.

8.1. 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 3 shows the relationships among parties in a NetBill transaction.

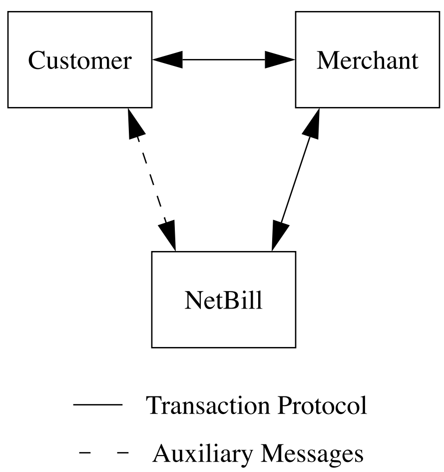

Figure 3: Parties in a NetBill Transaction.

8.1.1. 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.

8.2. 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 8.1.1 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 8.3.2 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 8.4.1 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.

Section 8.4.2 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.

8.2.1. 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 [17].
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 [21].
$E_{X\text{-PRI}}$(*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 [18].

$\{Message\}^{X}$	*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.

8.2.2. The Price Request Phase

This section assumes possession of tickets; the method for obtaining tickets is shown in Section 8.3. The *Identity* item may actually be a pseudonym, as shown in Section 8.3.2.

1. $C \Rightarrow M$ T_{CM}(Identity), E_{CM}(Credentials, PRD, Bid, RequestFlags, TID)
2. $M \Rightarrow C$ E_{CM}(ProductID, Price, Request-Flags, 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 8.3.2 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 8.4.1.)

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 8.2.5.1 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.

8.2.3. 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.

8.2.4. 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 8.4.1) 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 8.4.2)
- 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. $C \Rightarrow M$ T_{CM}(Identity), $E_{CM}([EPO]_C)$
6. $M \Rightarrow N$ $T_{MN}(M)$, $E_{MN}([[EPO]_C, MAcct, MMemo, K]_M)$

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.

8.2.5. 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 8.2.5.1):

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

8.2.5.1. 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:

8.2.5.2. 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))

8.2.5.3. 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)

8.2.5.4. 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)

8.2.5.5. 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))

8.3. 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 [12] for a discussion of public key certification.)

Section 8.2.4 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 [28]):

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.

8.3.1. 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.

8.3.1.1. Key Validation and Revocation Certificates

We use a public key certificate scheme (like that presented in [12]) 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.

8.3.2. 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 [6] 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. $C \Rightarrow P$ $[\{\text{TrueIdentity, M, Timestamp, K1, Type}\}^P]_C$
2. $P \Rightarrow C$ $E_{K1}(\text{K2}, [\{\text{Pseudonym, M, Timestamp, K2}\}^M]_P, [\text{TrueIdentity, M, Pseudonym, Timestamp}]_P)$
3. $C \Rightarrow M$ $[\{\text{Pseudonym, M, Timestamp, K2}\}^M]_P$
4. $M \Rightarrow C$ $E_{K2}(T_{CM}(\text{Pseudonym}), \text{CM})$

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.

8.4. Credentials and Authorizations

In [19], 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.

8.4.1. 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. $C \Rightarrow G$ $T_{CG}(\text{Identity}), E_{CG}(\text{Group, CAcct})$
2. $G \Rightarrow C$ $E_{CG}([\text{Group, Detail, Identity, CC(CAcct, AcctVN), Timestamp}]_G, \text{AcctVN})$

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.

8.4.2. Access Control Mechanism

As noted in [19], 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 8.2.4).

8.5. 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.

8.5.1. Customer Complaints

8.5.1.1. 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 8.2.5) 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.

8.5.1.2. 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).

8.5.2. Merchant Complaints

8.5.2.1. 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.

References

[1] A. 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." In *Proceedings of the First USENIX Workshop on Electronic Commerce,* pages 89–106, July 1995.

[3] N. Borenstein. "Perils and Pitfalls of Practical Cyber Commerce: the Lessons of First Virtual's First Year." Presented at *Frontiers in Electronic Commerce*, Austin, TX, October 1994.

[4] E. Brickell, P. Gemmell, and D. Kravitz. "Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change." In *Proceedings of the Sixth ACM-SIAM Symposium on Discrete Algorithms,* pages 457–466, 1995.

[5] L. Camp, M. Sirbu, and J. D. Tygar. "Token and Notational Money in Electronic Commerce." In *Proceedings of the First USENIX Workshop on Electronic Commerce,* pages 1–12, July 1995.

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

[7] B. Cox, J. D. Tygar, and M. Sirbu. "NetBill Security and Transaction Protocol." In *Proceedings of the First USENIX Workshop on Electronic Commerce,* pages 77–88, July 1995.

[8] H. Gobioff, S. Smith, and J. D. Tygar. *Smart Cards in Hostile Environment*. CMU-CS Technical Report CMU-CS-95-188, September 1995.

[9] J. Gray and A. Reuter. *Transactions Processing: Techniques and Concepts*. Morgan Kaufmann, San Mateo, CA, 1994.

[10] N. Heintze, J. D. Tygar, and B. Yee. "Cryptographic Postage Indicia." To appear.

[11] S. Itkin and J. Martell. *A PDF417 Primer: A Guide to Understanding Second Generation Bar Codes and Portable Data Files*. Technical Report Monograph 8, Symbol Technologies. April 1988

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

[13] N. Lynch, M. Merritt, W. Weihl, A. Fekete. *Atomic Transactions*. Morgan Kaufmann, San Mateo, CA, 1994.

[14] M. Manasse. "The Millicent Protocols for Electronic Commerce." In *Proceedings of the First USENIX Workshop on Electronic Commerce,* pages 117–123, July 1995.

[15] R. Mori and M. Kawahara. "Superdistribution: the Concept and the Architecture." In *Transactions of the Institute of Electronics, Information, and Communication Engineers (Japan),* E73(7) pages 1133–1146.

[16] National Institute of Standards and Technology. *FIPS 140-1: Security Requirements for Cryptographic Modules*. January 1994

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

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

[19] B. 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.

[20] T. Rabin and M. Ben-Or. “Verifiable Secret Sharing and Multiparty Protocols with Honest Majority.” In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 73–85, May 1989.

[21] 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.

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

[23] M. Sirbu and J. D. Tygar. “NetBill: An Internet Commerce System Optimized for Network Delivered Services.” In *IEEE Personal Communications*, 2(4) pages 34–39, August 1995.

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

[25] S. Smith. *Secure Distributed Time for Secure Distributed Protocols*. Ph.D. Thesis, CMU-CS Technical Report CMU-CS-94-177, September 1994.

[26] S. Smith, D. Johnson, and J. D. Tygar. “Completely Asynchronous Optimistic Recovery with Minimal Rollbacks.” In *Proceedings of the 25th International IEEE Symposium on Fault-Tolerant Computing*, pages 362–372, June 1995.

[27] S. Smith and J. D. Tygar. “Security and Privacy for Partial Order Time.” In *Proceedings of the ISCA International Conference on Parallel and Distributed Computing Systems*, pages 70–79, October 1994.

[28] J. Steiner, B. Neuman and J. Schiller. “Kerberos: An Authentication Service for Open Network Systems.” In *USENIX Winter Conference*, pages 191–202, February 1988.

[29] US Postal Service. *Information Based Indicia Program (IBIP) New Direction Metering Technology.* May 1995.

[30] USENIX Association. *Proceedings of the First USENIX Workshop on Electronic Commerce*, July 1995.

[31] Visa USA and Anderson Consulting. *1992 Credit Card Functional Cost Study.* September 1992.

[32] B. Yee. *Using Secure Coprocessors*. Ph.D. Thesis, CMU-CS Technical Report CMU-CS-94-149, May 1994.

[33] B. Yee and J. D. Tygar. “Secure Coprocessors in Electronic Commerce Applications.” In *Proceedings of the First USENIX Workshop on Electronic Commerce,* pages 155–170, July 1995.