An efficient commerce system

Andrew C. Myers

November 1996
CMU-CS-96-191

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This report is a revision of an undergraduate thesis submitted to Princeton University in May, 1996.

Keywords: electronic commerce, DigiCash, First Virtual, Millicent, Secure Electronic Transactions

Abstract

We give an introduction to some of the issues surrounding the construction of electronic commerce systems and discuss some proposed solutions:

- Visa and MasterCard's SET
- First Virtual's Green commerce model
- DigiCash
- Millicent

We also present a new payment protocol created to enable the electronic purchase of small bits of information such as database queries, newspaper articles, web searches, etc. Our system's goals:

- The minimum acceptable transaction size will be five cents.
- Committing fraud will be very difficult.
- Merchants will not know customers' identities.
- The credit company will not know what customers buy.

Security is implemented with one–way hash functions. We chose not to use public key cryptography because it would make transactions too computationally expensive.

This report is a revision of an undergraduate thesis submitted to Princeton University in May, 1996.

Acknowledgments

I would like to thank Andrew Appel, my adviser, for his guidance and support; Mark Manasse, Charles Jones, and Saul Hansell for their assistance.

Contents

Chapter 1

Introduction

Creating a commerce system for the Internet is very much like trying to carry a gallon of water around in a sieve: there are many tiny difficulties that must be avoided in order to succeed. The task becomes a matter of constant compromise and balance. Complete success is practically impossible to achieve; partial success is more likely.

The main problem in a commerce system is security. Commerce is about moving money; security flaws allow money to slip away. Patching one flaw can create several others. At the same time, there are other issues to be considered, such as the system's overhead, speed, and how much privacy it affords users. The problem is a mix of theoretical and practical obstacles whose solution must in the end appeal to potential users in order to be effective.

The motivation for an electronic commerce system is to allow people to buy information over the Internet. Currently, we are forced to buy a whole newspaper just to read one article. To find one piece of information, we might be forced to buy a large book. We would like to be able to buy these small items individually rather than paying and receiving more than we want. Unfortunately, none of the many proposed systems meets our goals.

Some systems are designed for much larger transactions, others for much smaller transactions. Some are too insecure, others do not offer enough privacy for our tastes. Thus, we have decided to create a system suitable for handling small transactions (about five cents) securely, with moderate privacy, and with protection for both the buyer and seller in a transaction.

Still, other systems are worth examining closely to learn how they have addressed the issues. These solutions aid us in designing new systems, both commerce–related and otherwise. In addition, it is likely that at some time in the future, if not already, we will buy something from a vendor on the Internet. Examining these systems will help us to make an informed decision about which system we should actually employ to make a purchase.

Chapter 2

Building a commerce system

The most important issue in electronic commerce is security. Security supplies trust: merchants trust that customers will pay the proper amount, and customers trust that they will receive the goods for which they asked. In addition, customers trust that they will be billed only for the item they bought at the price to which they agreed. Without security, trust is eradicated as the system is exploited by evil parties bent on acquiring goods and money. To avoid losing money, honest merchants and customers will prefer a system with effective security.

Security is not an all–or–nothing feature. A system that is more secure is more appealing until the security begins to make the system inconvenient to use. If completing the security procedure for a one cent purchase requires five minutes, customers will be disinclined to make the purchase. The minimum amount of security necessary is the amount that will make the benefit of breaking the security smaller than the cost. We believe that a thief will not spend two dollars in computer time to decode information that he could have purchased for one dollar. On the other hand, people interested only in making mischief, not in financial gain, will not be deterred by a low return on their invested effort.

The second most important issue in electronic commerce is speed. If a transaction requires an enormous amount of computation, the cost of the transaction will be dominated by the cost of the computing time. Assuming a typical Internet commerce company needs to make roughly $300,000 per year per computer to pay for hardware maintenance and other expenses (this estimate was inspired by Manasse [7]), it would need to take in about one cent per second (a year has about 31.5 million seconds).

Computing the speed of a particular commerce system is a question not only of how quickly each transaction is completed, but also of the peak demand on the system. Let's assume that at peak times, a system receives ten times the average number of requests per second (this estimate also from Manasse). We judge how much load a computer can handle by the peak number of requests, but we judge how much revenue a computer will receive by the average number. For instance, we can assume that a computer which can handle a maximum of 100 transactions per second will receive the revenue from at most 10 transactions per second. If the computer handles more than 10 transactions per second on average, it will not be able to handle peak load.

An issue related to the speed of transactions is the number of messages sent per transaction. In general, as the number of messages sent increases, so does the time required to complete the transaction. The network delay, i.e. the time the messages take to travel on the network, does not contribute to a transaction's cost of computing time. While a computer is waiting for a message, it can be engaged in other processes. However, the amount of time that a customer has to wait for a transaction to be completed will be increased by network delay.

A protocol with a larger number of messages per transaction usually implies that a larger number of parties are involved. Increasing the complexity of a transaction in this way increases the number of possible points at which the transaction can fail. Not only will a transaction with more opportunities for failure fail more often, but it will also take longer to find the problem and report an error. The result is that a complicated protocol increases both the time a customer has to wait and the cost of each transaction through increased use of network resources.

The amount of privacy in a commerce system is also an important consideration. Merchants will probably want to be well known. Customers may feel the opposite way. They may not want their names to be put on a mailing list, or they may not want where they shop or what they buy to be publicly known. In the best case, the merchant would know what a customer buys, but would not know the customer's identity. The rest of the world might know who the customer is, but would not know who he bought from or what he had bought.

On the other hand, it would be nice for the merchant and customer to have a written contract witnessed by a third party specifying the item to be purchased and its price. A contract would be useful in case a party was not satisfied with the outcome of a transaction and wanted to take additional action to rectify the situation. Though it seems that maintaining privacy and producing a contract are opposed to each other, they can both be achieved simultaneously (but partially), as our system demonstrates.

There are other issues to be considered. We should not only look at how difficult it is to become a customer in a given system, but how difficult it is to become a merchant. If it is easy, more people will become merchants, increasing the content available to customers. There are many other criteria by which to judge commerce systems; we have presented the ones we feel are most significant.

2.1 Selected solutions

2.1.1 Credit Cards

There are several reasons to include an explanation of standard credit cards in this paper. The primary reason is that some of the commerce systems to be discussed are merely wrappers for credit cards. Also, studying the cost of credit card transactions may help us to estimate the costs associated with other commerce systems.

A typical credit card transaction involves five parties, the cardholder, the merchant, the merchant's bank, the cardholder's bank, and the credit card company. When a cardholder buys something from a merchant, the merchant asks its bank and the merchant's bank asks the credit card company if the charge can go through. The credit card company asks the cardholder's bank. The reply from the cardholder's bank filters back through the network to the merchant, who proceeds with the transaction. Billing information propagates from party to party in the same way. Finally, the cardholder's bank sends the cardholder a bill. [5]

Payments take the opposite route through the system, beginning with the cardholder paying his bank. The cardholder's bank keeps 1.3% of the payment in addition to any interest that may have accrued if the cardholder did not pay right away. The rest of the payment is sent to the credit card company, which charges the merchant's and cardholder's banks about five cents each. Finally, the payment is sent to the merchant's bank, which keeps 1.9% and sends the rest to the merchant. [5]

The plan of action that credit card companies, specifically Visa [4], are pursuing is to build a range of services around issuing credit cards. With a Visa card comes insurance, buyer protection, and many other bonuses. This approach is contrary to electronic payment systems, where the focus is only on producing a workable method to transfer money from the customer to the merchant.

2.1.2 First Virtual

First Virtual's system [11], known as the Green Commerce Model, is to be admired if for no other reason than that it is already in operation. The system's purpose is to allow consumers to use their credit cards over the Internet without fear that their card information will be stolen. While this would usually mean that the system inherits the overhead associated with a credit card, First Virtual avoids that problem through aggregation. Instead of sending each charge to the credit card company, charges are cached at First Virtual and sent off to the credit company as a single charge. This scheme effectively reduces the overhead of a credit card, but it also increases the risk. A customer may be over his credit limit when the charges are finally sent to the credit company.

The Green system is particularly remarkable because it uses no cryptographic tools. First Virtual's position is clear:

> Although use of cryptographic services could provide protection against masquerade, replay, and manipulation of the transfer–response message, the Internet community presently lacks the supporting infrastructure to deploy these services on a widespread basis. [11]

Because of its lack of security as well as its heavy reliance on email, First Virtual's system is a veritable Swiss cheese of security holes. It seems that any user with knowledge of how to fake an email and how to eavesdrop on network traffic can disrupt others' transactions and make purchases using others' credit. Perhaps First Virtual's system only remains in operation due to a dearth of malicious hackers or First Virtual subscribers. While this system will protect credit card information, its method of protecting users' accounts from unauthorized use in the electronic realm is feeble and ineffective.

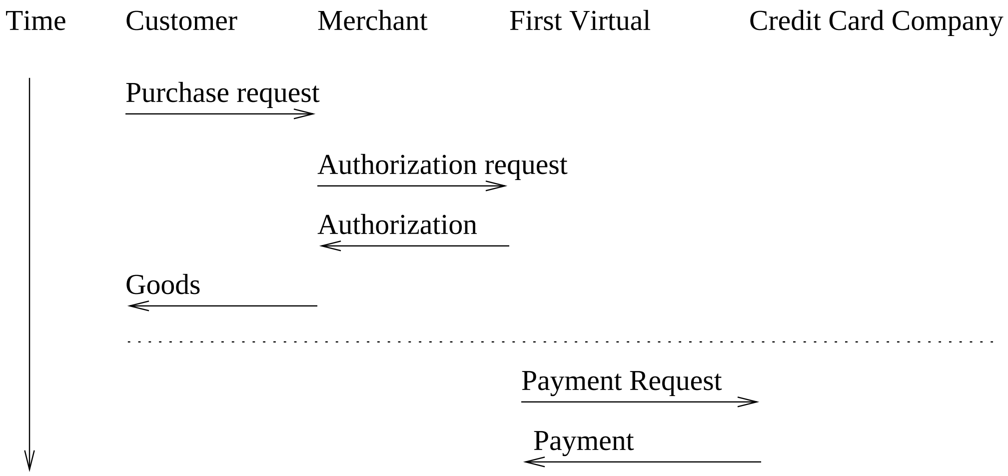

Figure 2.1: First Virtual's protocol

Both the customer and merchant have to register with First Virtual. The customer supplies his credit card information, email address and some alphanumeric characters. First Virtual returns an identification number made from the customer's characters plus a few more of its choice. The merchant tells First Virtual how it would like to be paid.

To start a transaction, a customer sends his identification number to the merchant. The merchant sends this on to First Virtual with an order to transfer funds from the customer's account to the merchant's. If the customer's identification number is invalid, First Virtual tells the merchant. If not, First Virtual sends an email to the customer asking whether it should go ahead with the funds transfer. In the meantime, the merchant delivers the ordered goods to the customer. Note that in this system there is no guarantee that the merchant will be paid.

The customer can answer First Virtual's email in one of three ways. He can reply that the transfer should go ahead, that it should not, or that it should not because fraud was involved. Answering that there was fraud will instantly cancel the customer's account with First Virtual. Also, if the customer decides not to pay often enough, his account will be cancelled. If the customer does not answer at all within a set number of days, his account will be suspended. If the customer decides to pay, all parties in the transaction are satisfied. At some future time, First Virtual will charge the transaction to the customer's credit card and will distribute payment to the merchant.

One possible attack

Attacking First Virtual's security is mostly a matter of acquiring a user's identification number. With an identification number, anyone could make a purchase at a merchant who supports First Virtual's system. The items bought would be transmitted immediately. Later, the owner of the account would be asked to pay for the transaction. The number of transactions that can be made depends on how quickly the account owner responds to First Virtual's queries. Using an account number is as simple as filling it in where requested (usually an order form on the World Wide Web) when making purchases.

Another possibility is to acquire transaction numbers. These identifiers will enable the bearer to authorize or refuse transactions as well as to report fraud, instantly closing the account associated with the transaction number. Obviously, the potential for mischief is very high. If Alice intercepts Bob's identification number, uses it to make a purchase, and intercepts the associated transaction number, then she can authorize First Virtual to bill Bob. Even better, when Bob tries to stop payment, First Virtual will report that the transaction has been closed and that he probably made a mistake.

Conversely, if Alice is able to intercept a transaction number from one of Bob's purchases, she can use it to tell First Virtual that fraud took place. The result is that Bob's account would be instantly shut down, and the merchant Bob used would not be paid.

Because First Virtual employs no cryptography in their protocol, all that is required of Alice (besides moral turpitude), is the ability to read Bob's email. If Alice's computer is on the same network as Bob's, she could sniff packets going in and out of Bob's computer. Otherwise, she could exploit any of the multitude of security holes in most operating systems to examine Bob's electronic mailbox.

2.1.3 Visa and MasterCard's SET

On February 1, 1996, Visa, MasterCard, Microsoft, Netscape, and others announced that they would support the Secure Electronic Transactions (SET) protocol [13]. SET's purpose is to enable credit cards to be used safely on the Internet. Care is taken to ensure that no party knows more than they should: All messages are encrypted via DES (with the DES key exchanged through RSA public key cryptography) to prevent third–party oberservers

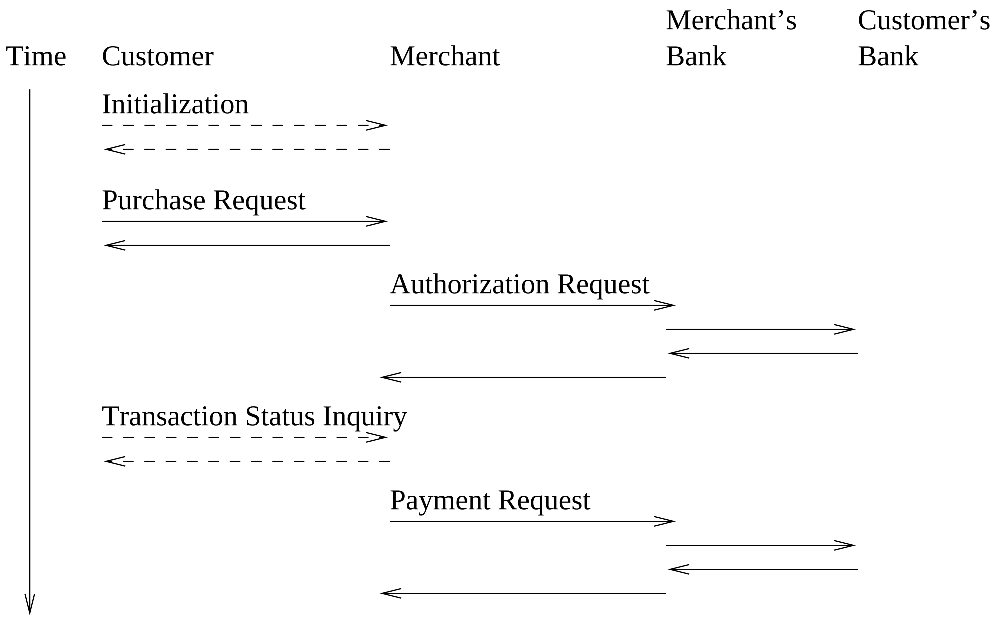

Figure 2.2: SET's message protocol (dotted messages are optional).

from learning anything. The customer's payment information is encrypted so that the merchant is unable to read it. The merchant and the cardholder are the only two parties who know what the cardholder is ordering. The system is heavily reliant on authentication through public key signatures, as distributed by a series of trust brokers. [10]

A SET transaction bears a close resemblance to a standard credit card transaction. To initiate a purchase, a cardholder sends an order for some item to a merchant via the Internet. The merchant, in order to get approval for the transaction, sends a message to the Acquirer payment gateway, another name for a computer which acts as the intermediary between the Internet and the Acquirer, a.k.a. the merchant's bank. The Acquirer sends a message to the Issuer, a.k.a. the cardholder's bank, via the banks' proprietary communications network. The Issuer checks to see whether the cardholder is in good standing and returns the result, which propagates back to the merchant. The merchant has the choice of requesting payment in the same message as the request for approval, or in a separate message which takes the same path. [10]

A whole transaction can be accomplished in six messages, but the number of messages can swell to 14 if a few optional but recommended pieces of the protocol are used. A simple transaction requires two encryptions: one for the message to the merchant and the other for the message to the Acquirer. Before that, the cardholder needs to acquire a series of certificates to verify that the Acquirer is legitimate. Verification involves checking that the root certificate, which the cardholder holds, matches the signature on the next certificate, which matches the signature on the next certificate, etc., for three or four levels. At each level, another RSA decryption is necessary. [10]

Whether or not SET is secure, it is certainly slow. The extensive use of public key cryptography guarantees that a transaction will take a significant amount of time to complete. To estimate how long it would take for the cardholder to process a transaction, we found the amount of time necessary for PGP (version 2.6.2) to decrypt and verify the signature on an eight byte message encrypted and signed with 1024 bit keys: around 1.2 seconds on an Intel '486. Completing one SET transaction would probably consume a significant number of seconds on the same computer. The Acquirer would have a smaller but still significant amount to compute. The sum of the standard credit card costs along with the cost of the computing time necessary for one transaction will probably turn out to be quite a large number. Unfortunately, Visa and MasterCard have not release estimated or actual performance figures for SET.

2.1.4 DigiCash

DigiCash [1] uses RSA public key cryptography to provide strong, anonymous digital cash. One drawback to this system is the amount of processor time needed to complete transactions. DigiCash requires that the serial number on each piece of money spent needs to be compared with the serial numbers on every previously spent piece of money to prevent cash from being spent more than once. A modified version of the protocol eliminates the verification step, but does not detect double spending until after the transaction is completed. In the modified scheme, when double spending does occur, the anonymity of the cash is compromised, eventually revealing the culprit.

In addition to the time required to check for double spending, an enormous amount of processor time is required to verify digital cash. The merchant must verify the bank's signature on each digital coin that it is given. On an Intel '486, verifying a 1024 bit public key signature on a short message using PGP takes about 0.2 seconds. Additional processing would be necessary to check for double spending. It is probably safe to assume that the average transaction would take the merchant about half a second. At this rate, each transaction would cost the merchant five cents in computing time. For transactions below 25 cents, DigiCash's system would be very expensive.

2.1.5 Millicent

Millicent [6] is a scheme for making tiny (less than one cent) purchases that seeks to evade the traditional difficulties with digital cash through decentralization. Instead of issuing money through banks, cash, now called *scrip*, is issued by each merchant. Customers buy various merchants' scrip from a broker, an entity that has a contract to produce and sell scrip for merchants. Millicent does not address the process of establishing a relationship between a broker and a customer or between a broker and a merchant: these are left to other protocols which are meant to handle larger transactions.

In an effort to make transactions fast and inexpensive, Millicent has been stripped of features. There are no receipts in a standard transaction. There is no method of guaranteeing that a customer gets the item that was ordered, or even any item at all. Customers are encouraged to complain to the merchant's broker, but the broker is unlikely to act until a significant number of customers have complained. Finally, if a customer loses unspent scrip, there is no way to recover it. The sacrifices made in customer service pay off in efficiency: Millicent can be used to pay for transactions as small as 0.1 cents. The spartan nature of the system may not be important for small transactions, but they limit the system's usefulness for larger transactions.

Scrip consists of a vendor identification, an amount, an expiration date, and a signature. The signature, which can only be produced or verified by a vendor (and any brokers to whom the vendor licenses scrip production), is produced by a one–way hash function (see Appendix A). Each piece of scrip has a serial number which is used to detect double spending.

The protocol for buying goods from a merchant is usually very quick and easy. A customer sends a request and some scrip to a merchant and the merchant replies with the goods and perhaps more scrip as change. Unfortunately, the worst case version of this transaction is much slower. If a customer does not have scrip for a vendor, he will need to contact his broker. If his broker also lacks the required scrip, it will ask the merchant where to buy scrip and transfer this information to the customer. The customer buys scrip from the merchant's broker. Finally, the customer will actually make

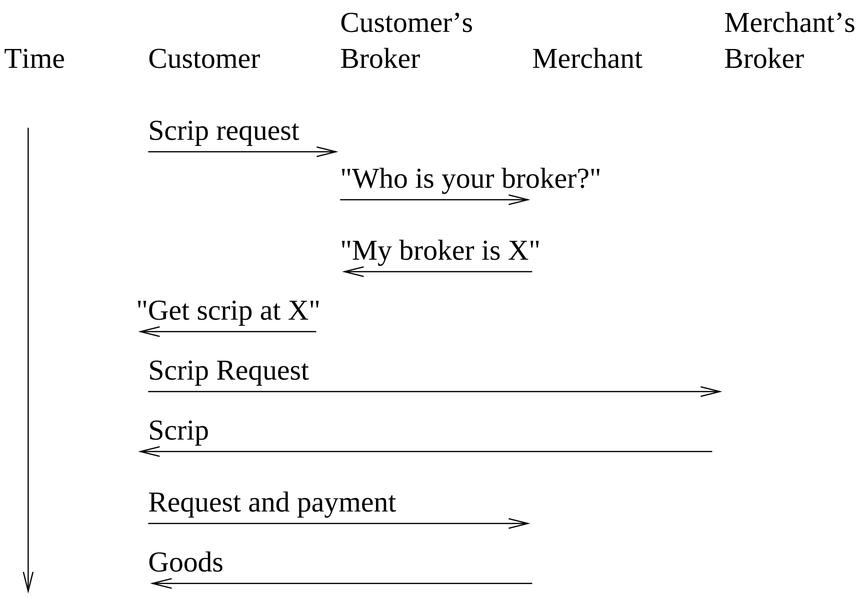

Figure 2.3: Worst case scenario for a Millicent transaction.

the purchase. At this point, eight messages have been passed between four hosts (see Figure 2.3). As the number of places at which the customer shops increases, this sequence of messages will be more frequent and the additional delay will become significant.

Another difficulty that can arise when a customer shops in many different places is accumulation of spare scrip. It is unlikely that a customer will spend all of his scrip. Rather, the customer will usually have a small amount left over after every purchase. Because Millicent deals with tiny purchases, the amount of leftover scrip will be on the order of a penny. Since pennies can eventually add up to a large amount, customers can sell their scrip back to the associated vendors at a discount.

Chapter 3

Our system

3.1 Design Goals

There are three parties in every transaction in our system: the customer, the merchant, and the credit company. The credit company pays the merchant on behalf of the customer. Eventually, the customer repays the credit company. Both the customer and merchant need to have accounts with the credit company before they can interact. A merchant or customer gets an account by agreeing on an identification number and secret bit string with the credit company.

Our system allows customers some privacy, but not total anonymity. Merchants never see the real identity of a customer, but they can link the customer's identification number to all of that customer's purchases. This allows merchants to reward their frequent customers without knowing who those people are. The credit company is able to link identification numbers to actual people, but it will not know what merchants send to customers or what customers order from merchants. If a dispute between a customer and merchant arises, the credit company will be able to verify both the customer's order to the merchant and what the merchant intended to send to the customer. The customer and merchant are given privacy and protection from fraud.

If a group of merchants were to share their customer information, they would be able to develop a more complete record of customers' buying habits. However, only the credit company can link customer identification numbers with actual people. We hope that most credit companies will abstain from releasing the identities of their customers. Given that current credit companies release information about their cardholders (credit record information, for instance), our assumptions may be somewhat unrealistic. Perhaps people who are concerned about their privacy will boycott overly loquacious credit companies.

Since our system uses no munitions [2], i.e. strong cryptography, it is exportable. While it may be desirable to encrypt traffic between the merchant and the customer, it is not necessary. For the security–conscious, our system can be easily converted to use a protocol such as secure sockets for communication between the merchant and customer. All messages between the merchant and the credit company should not be encrypted, since encryption would make our system much slower and therefore, more expensive.

3.2 Why our system works

Our system relies mainly on hashing to enforce security and allow anonymity. The customer and merchant sign[1] their messages so that the credit company can verify the signatures. The credit company produces two signatures, one that the merchant can verify and the other that the customer can verify. If a message is altered in any way, it is extremely likely that the signature will reveal that a change has been made. Signatures prevent an evil party from adding or corrupting information during a transaction.

Hashing is also used to produce a digital "contract" that the customer and merchant sign for each transaction. The customer produces a signed order by signing the hash of the description of the item to be purchased. The merchant can check that the hash is correct. The customer's signature guarantees that he cannot later lie about what he ordered. Similarly, the merchant hashes the data to be delivered to the customer and signs the hash. The credit company, after verifying the merchant's signature, will also sign the hash with a signature that the customer can verify.

Anonymity is supplied in two respects. Customers are known to merchants only by their identification numbers. A merchant can tell from where a customer is connecting, but cannot tell the customer's identity. If a customer does not want to reveal from where he is connecting, he can use a gateway computer to forward his packets rather than sending them directly. The credit company is able to link a customer's identification number with

[1] A discussion of signatures can be found in Appendix A.

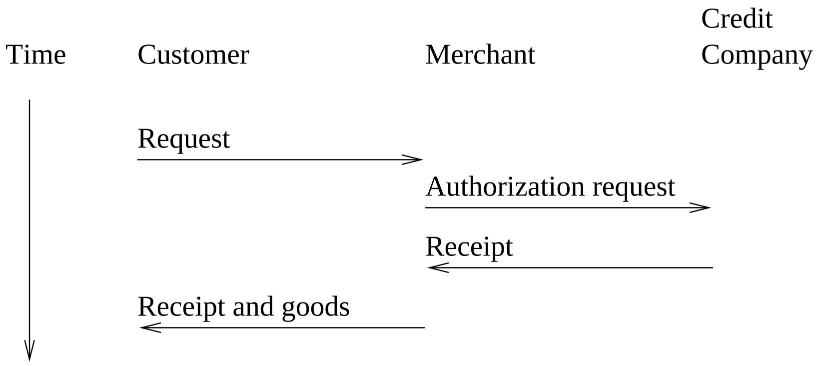

Figure 3.1: Our system's message passing scheme.

his true identity, precluding total anonymity. But unless the credit company decides to release the name to identification number matching, customers' anonymity will be preserved.

The other aspect of anonymity is that the credit company will not know either what the customer orders or what the merchant delivers. While the credit company will receive signed hashes of both, it will never see the actual data. Appending random bits to both the customer's order and the merchant's reply makes it very difficult for the credit company to find out what is being hashed since the random bits increase the number of possible documents from which the hash could be produced.

3.3 The protocol

The customer initiates a transaction by sending a purchase request to the merchant. The purchase request consists of two parts. One part is a description of what the customer wants to purchase. The other part consists of all other necessary information: identification numbers for the customer, the customer's credit company, and the merchant; the amount of the transaction, the number of transactions that the customer has initiated before this one (a.k.a. the transaction counter), a hash of the item description, and the customer's signature on the whole message except the item description.

The merchant, upon receiving the customer's packet, checks that the item description matches its hash. If it does not or if there are any other problems with the item description or amount to be paid, the merchant will send an error message to the customer explaining what the problem is. If everything seems to be in order, the merchant will send the following to the credit company: the customer's message minus the item description, a revised amount on the purchase (which can only be less than or equal to the customer's amount), the merchant's transaction counter, a hash of the data that the customer has ordered, and a signature on the whole message.

When it receives the merchant's message, the credit company will check the merchant's and customer's signatures. The credit company will also make sure that both the merchant and the customer's transaction counters have not been used more than once. The credit company expects to receive transactions counters from a host in ascending order. The credit company will tolerate small skips in transaction counters that could have been produced by a customer or merchant requesting several transactions in a very short time. If the gap in transaction counters remains for a significant period of time or is very large, the credit company will contact the customer or merchant to ask what the problem is.

If both signatures as well as the rest of the packet are valid, the client's account will be charged and the merchant's account will be credited. The credit company's reply to the merchant will include each party's identification number, the client's and merchant's transaction counters, the amount, the merchant's hash of the requested data, and two signatures of the whole packet, one for the merchant and one for the customer.

If the credit company reports that the transaction is successful, the merchant will forward the credit company's message to the customer. The merchant will also send the data that the customer ordered. Here the merchant has a choice: it can either send the customer's data before it receives the credit company's reply, or it can send the data after. The merchant would be exposing itself to greater risk by not waiting for the credit company, but it would potentially increase customer satisfaction by decreasing the time it takes for the customer to receive his purchase. If the network link between the merchant and credit company is slow, the merchant will have a large incentive to send the data first.

3.4 Potential weaknesses

Our system is not immune to attacks in which packets are destroyed or corrupted. It is quite possible to prevent all transactions just by changing a byte or two in each packet going to or from the credit company. It should be noted that this attack would be effective against any of the other systems we have considered. While our system can be shut down, there are few if any vulnerabilities that will allow theft.

One possible attack is for an evil hacker to impersonate a customer that is known to some merchant. The merchant, being the trusting sort, would send the data that the fake customer requested before receiving the bank's confirmation. To impersonate a customer, the attacker needs to use the customer's identification number and IP address. Packet sniffing is all that is required to find the identification number. However, if the attacker also tries to make his packets seem to come from the customer's IP address, he will have problems. If the customer's computer is turned on when the attacker initiates a connection with the merchant, the merchant's packets to the customer's IP address will cause the customer's computer to reset the connection between the merchant and fake customer (assuming a TCP connection is used) [8]. Even if the customer's computer is off, the attacker will only get one order filled before the merchant stops trusting the spoofed customer.

A more direct approach might be to steal the customer's identification number, secret bit string, and transaction counter. We require that each customer's data be protected by a password. Thus, if the data is stolen, it will most likely be the customer's fault, not our system's. Assuming that the data is stolen, the customer's transaction counter will reveal the theft. When the thief uses the customer's data, the transaction counter will be incremented at the thief's computer and at the credit company but not at the customer's computer. The next time the customer makes a purchase, a transaction counter value will be used twice, indicating that there is a problem. If the thief decides to add some value to his copy of the transaction counter to avoid collisions, the credit company, noticing the gap in transaction counters, will contact the customer to make sure that everything is in order.

Finally, there is the possibility that a brute force attack will succeed [9]. Having intercepted customer A's message, an evil hacker could try hashing every possible bit string concatenated with the message until he generated a signature identical to the one on the message. Assuming that he knew the correct value of the transaction counter (which is possible), he could generate messages that appear to come from customer A. The chance that this attack will succeed is very low given that there are at 2^{160} possible signatures to search through. Even if the hacker could try 10^6 bit strings per second, it

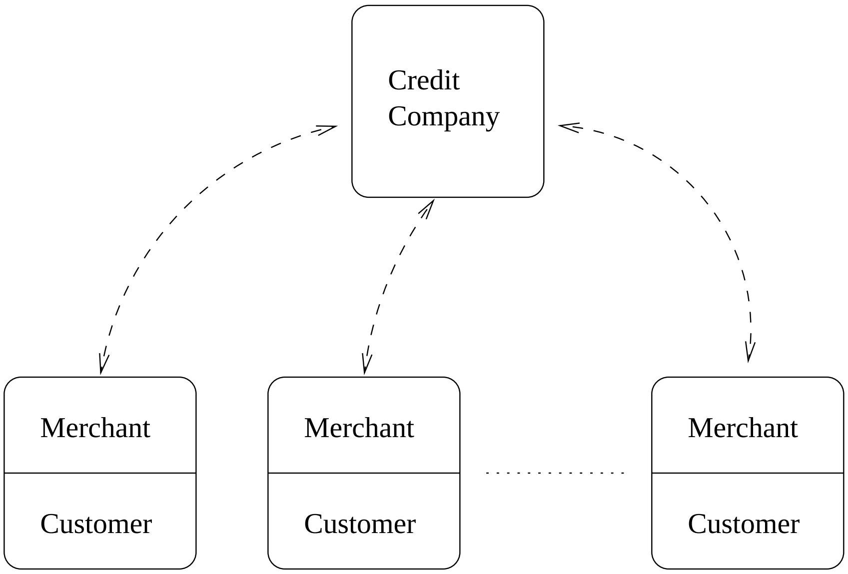

Figure 3.2: The setup for our performance analysis.

would take about 10^{34} years to try every possibility.

3.5 Performance analysis

In order to find out exactly how fast our system is, we implemented and benchmarked it. The experiment consisted of eleven customers generating purchase requests continuously, one merchant per customer, and a central credit company server which processed all of the merchants' requests. We found that the credit company could process a sustained rate of 57 transactions per second.

We implemented the software in C to avoid any unnecessary overhead. The credit company kept records on each customer and merchant as well as receipts for every transaction. Customers kept the location of a merchant and personal information (their secret bit string, etc.). Merchants only kept records about themselves. We stored all of this information in regular files without a database. All programs operated serially, so there was no need to lock records.

A transaction consisted of the following: The customer would open a TCP connection to the merchant (the customer and merchant ran on the

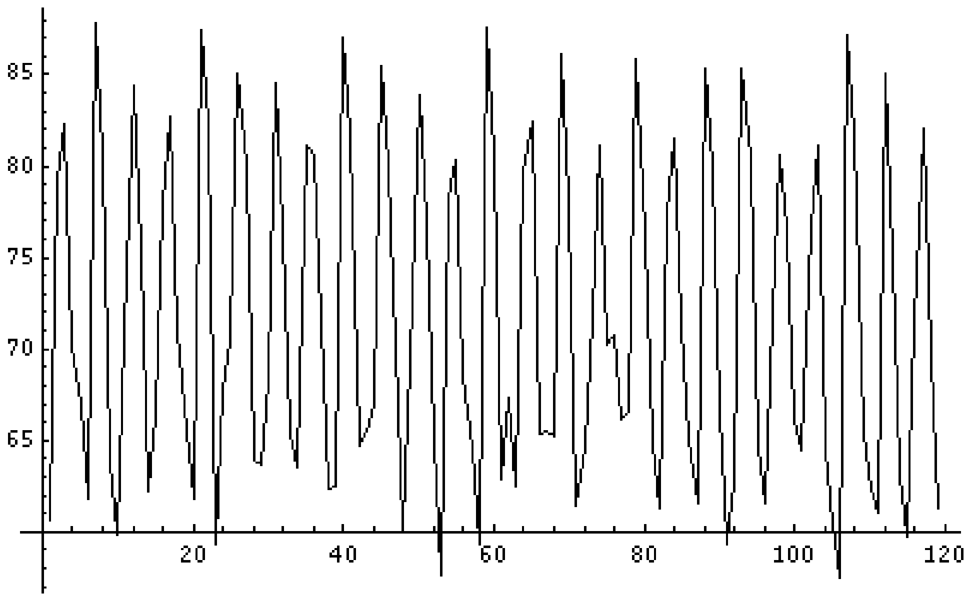

Figure 3.3: Transactions per second versus time (in minutes).

same machine) and transmit a purchase request. If the merchant found that the request was invalid, it would send an error message back to the customer. Otherwise, it would open a TCP connection to the credit company and transmit payment information. The credit company would process the information, return a receipt that either confirmed that the transaction occurred or explained why it failed, and close the connection. The merchant would forward the receipt to the customer along with the purchased electronic data, and then it would close the connection. A counter in the credit company server was incremented each time it closed a connection to a merchant. Once a minute, an interrupt was generated, the value of the counter was printed, and the counter was reset to zero.

The computers used were all connected by a 10 Mb/s local area ethernet network. The credit company ran on an otherwise idle Hewlett Packard 712/80 with HP-UX. (The current list price of an HP 712/80 is about $10,000.) We collected data once per minute for 119 minutes. Over that period, the credit company processed an average of 71.65 transactions per second. The peak average rate sustained for a minute was 87.78 transactions per second and the lowest average rate sustained for a minute was 57.51 transactions per second. The standard deviation was 8.93.

Figure 3.3 shows the rate at which the credit company processed transactions versus time. We believe the fluctuations in the transaction rate are due to various unrelated system daemons periodically demanding resources (X was running, among other things). By removing all unrelated processes from the computer, we believe that the credit company server could achieve a sustained transaction rate between 70 and 80 transactions per second.

Assuming that our system cannot operate faster than 55 transactions per second, the credit company's cost of computer time per transaction is 0.18 cents. The merchant's cost of computer time per transaction should be roughly the same unless the channel between the merchant and customer is encrypted. The credit company would have to take a 3.6% commission on a five cent transaction, well within bounds of acceptability. Transactions as small as two cents each are possible, since the credit company's overhead would be only 9%. Of course, these calculations assume that the credit company will charge 0.18 cents per transaction, but in reality, its pricing policies are dependent on many other factors, such as demand for the system.

Having found the minimum transaction size, it would be useful to find the maximum. Finding the largest possible transaction is a matter of analyzing the amount of work required to steal. The maximum transaction size should be small enough so that a thief will invest more in stealing than he will receive in return. We believe our system is not vulnerable to any currently known cryptographic attack. It is most vulnerable to physical attacks, such as a thief using a gun to coerce people to buy him things. How safe our system is from physical attack is outside the scope of this investigation. Therefore, we cannot make a judgment about how large a transaction can be.

Chapter 4

Conclusion

The systems we have examined all have various flaws which make them unsuitable for our purposes. First Virtual's system offers no protection against fraud. Visa's SET has adequate security but seems to be too slow both because of the amount of computing time required to deal with all the necessary cryptography and because of the number of messages that need to be sent to complete one transaction. DigiCash seems to be faster than SET, and promises anonymity. Unfortunately, DigiCash is probably not fast enough. Millicent, while appealing for its normally low overhead, can become inefficient in some circumstances, sending eight messages to complete one transaction.

DigiCash and Millicent have one thing in common: neither offers any protection to the participants in a transaction. Our system creates a signed contract which formalizes the agreement between customer and merchant. DigiCash and Millicent have no such mechanism, relying instead on methods outside their systems to prevent or limit fraud. Altering DigiCash to provide protection would make as much sense as altering physical cash in the same way. DigiCash was not built to offer robust protection to its users. Millicent could theoretically install some amount of protection, but it might make the protocol too complicated to be efficient.

We believe that our system compares very favorably with other systems for handling transactions as small as five cents. Our system offers security, some privacy, the ability to verify that merchants and customers have fulfilled their contracts, and low overhead. If privacy is not an issue, no encryption at all is required, allowing transactions to be extremely fast. Our system successfully gives customers and merchants both convenience and security.

Appendix A

Become a cryptographer in five minutes

We will need to use a one–way hash function [9], an algorithm that transforms an arbitrary amount of data into a set length (usually around 100 to 200 bits) of data. A good one–way hash function makes it very difficult either to find the input from the output or to find two inputs that share the same output. Our system uses the Secure Hash Algorithm (SHA) because we believe it to be more secure than other well–known algorithms, e.g. MD5.

With a hash function, it is easy to construct an almost unforgeable signature for a document (as shown in [6]). Call the document to be signed D. The signature of D is the one–way hash of the concatenation of D and a long sequence of random bits (call this B) known only by the person who signed the document. Observe that if the hash function is good, then the easiest way to duplicate the signature (without knowing B) is to try computing the signature with every possible value of B. To prove that a person signed a document, the person need only reveal B, at which point anyone can compute the signature and verify that B is the right sequence. If two parties both know B but nobody else does, they can use B to sign and verify messages to each other.

Bibliography

[1] David Chaum. Achieving Electronic Privacy. Originally appeared in Scientific American, August 1992. Also found at `http://www.digicash.com/publish/sciam.html`, August 1992.

[2] John Gilmore. Cryptography Export Control Archives. Found at `ftp://ftp.cygnus.com/pub/export/export.html`.

[3] Douglas A. Hayes. *Bank Lending Policies*, pages 175–98. Division of Research, Graduate School of Business Administration, The University of Michigan, Ann Arbor, Michigan, 1977.

[4] Visa International. Visa U.S.A: The Evolution of a Full–Service Consumer–Payment System. Background information received from Visa., August 1995.

[5] Patrick J. Lyons. What Happens When a Customer Says 'Charge It'. *The New York Times*, March 7 1993. Section 3, page 8.

[6] Mark S. Manasse. The Millicent protocols for electronic commerce. Found at `http://www.research.digital.com/SRC/millicent/`.

[7] Mark S. Manasse. A talk on Millicent. The talk was presented at the Princeton University Department of Computer Science, February 1996.

[8] Information Sciences Institute of the University of Southern California. RFC 793: Transmission Control Protocol DARPA Internet Program Specification. Found at `http://www.cis.ohio-state.edu/htbin/rfc/rfc793.html`, September 1981.

[9] Bruce Schneier. *Applied Cryptography*, chapter 7 and 18. John Wiley & Sons, Inc., second edition, 1996.

[10] Secure Electronic Transaction (SET) Specification Book 2: Technical Specifications. Found at `http://www.visa.com/cgi-bin/vee/sf/set/settech.html?2+0`, February 1996.

[11] Lee H. Stein, Einar A. Stefferud, Nathaniel S. Borenstein, and Marshall T. Rose. The Green Commerce Model. Found at `http://www.fv.com/pubdocs/green-model.txt`, May 1995.

[12] Salvatore J. Stolfo. A conversation with Mr. Stolfo at Columbia University on February 29, 1996.

[13] Visa and MasterCard combine security specifications for card transactions on the Internet. Found at `http://www.visa.com/cgi-bin/vee/vw/news/PRelco020196.html?2+0`.