Pay-Per-Use Services: Efficient, Lightweight Transactions in the Post–PC Era

Mike Chen and Andrew Geweke
Computer Science Division
Department of Electrical Engineering and Computer Science
The University of California, Berkeley
{ *mikechen*, *geweke* }@cs.berkeley.edu

Abstract. We define a subset of everyday financial transactions that involves low-cost, simple exchanges of goods or services for money; these "pay-per-use" services, we argue, are poorly served by existing electronic-payment infrastructures. We outline the desirable properties, both security-conscious and otherwise, that a new payment infrastructure should have. We then introduce a new payment model and protocol, using a hand-held computer (the 3Com Palm III) to provide user authentication. We demonstrate that the protocol achieves the goals set forth previously, and analyze its security. Finally, we conclude with experiences gleaned from our implementation and outline some future directions.

1 Introduction

Literally millions, if not billions, of small-scale financial transactions occur every day: people buy lunch, a Coke, dry-cleaning service, or gasoline. These transactions are almost uniformly small, paid for with cash, and involve a simple exchange of goods for money.

While there is a well-built, well-understood infrastructure for all-electronic payment of larger items — credit and ATM cards being the typical methods of payment — this infrastructure proves to be too heavyweight for small transactions. Gas stations typically charge an additional $0.25 (US) for the use of an ATM card, and up to 4% for the use of a credit card (except where prohibited by law); these fees, while perhaps acceptable for a $10–$20 gasoline purchase, become unacceptable for a $0.75 Coke.

At the same time, handheld, wireless computing is starting to become a reality. As computers like 3Com's Pilot and Microsoft's Palm PC become increasingly affordable and popular, an entire new generation of applications is enabled, and a new stage in ubiquitous computing unfolds. The key advance made by these "devices" is that they are sufficiently portable to be carried around on a personal level, yet powerful enough to "tie in" to the computing infrastructure: they act as the user's portal onto the world of services provided by more powerful computers and the Internet.

We, then, demonstrate how these new handheld devices can be used to provide the necessary lightweight transactions required for small, day-to-day "pay-per-use" purchases. We have constructed a working prototype of a system in which the device is used to securely issue payment for goods. The system uses cryptography to provide security to both the consumer and the merchant of the transaction, and is efficient enough to use in transactions involving quite small amounts of money, down to even a few U.S. cents.

2 Problem

We consider now the exact definition of our problem more carefully. We will use a simple model in which there are three parties involved:

- The **Customer**, who wishes to purchase a good or service;
- The **Merchant**, who provides the good or service and accepts payment;
- The **Bank**, which holds accounts for the customer and merchant.

A fourth party is introduced later — for our specific protocol — to aid in verification of the merchant's trustworthiness, but this fourth party is not part of the fundamental model of the problem.

This model is typical of a great number of pay-per-use services, and is what is used in the real world on a day-to-day basis: the customer and merchant, obviously, must be present — that is, there must be a buyer and a seller — and the bank, although not always so obvious, is indeed present whenever a transaction is conducted. In credit-card or ATM-card transactions, the bank — the customer's bank, in this case — is used to verify the customer's identity and means of payment. In cash transactions, the bank — the U.S. central bank, for example — is used to guarantee payment by issuing unforgeable cash.

Given these three parties, there are several important details of their interaction:

Trust. Trust is a critical issue in any security-conscious system. In our system, we have three potential trust relationships, of which two are assumed to be in place. We assume that the customer and the merchant both trust the bank, while we assume no trust between the customer and the merchant. These relationships of trust seem reasonable; the merchant and customer must trust the bank because, after all, the bank has their money. However, the merchant and customer need not trust each other — and perhaps are actually quite competitive with each other, as the merchant wants the goods to be sold for a high price, while the customer wants them for as low a price as possible.

Connectivity. In the model, we assume that the merchant has "good connectivity" to the bank — basically, the ability to communicate small amounts of information in real-time to the bank. We make this assumption because it is a basic requirement of all such systems: assuming that the customer's bank account must be kept consistent (that is, it must not get overdrawn), if connectivity to the bank is lost, the system cannot continue operation. We do not assume connectivity of the customer; that is, except for the customer's ability to communicate with the merchant at the point and time of sale, the customer need not have reliable connections to anyone.

Computation. The amount of computing power available to the parties involved determines the cryptographic operations available to them. For our system, we have a somewhat *ad hoc* definition of computational power that simply corresponds to the actual power available to us experimentally: the device has about 10 MIPS and well less than 1 MFLOP of power available, while the merchant and bank have over 100 MIPS and 100 MFLOPS of computation available to them. This is in line with the assumption that the merchant and bank are fixed locations, with computational power equivalent to that of today's desktop PC, while the customer only has access to a portable device like a Palm III or a Palm PC.

Notably, the computational power we expect from the customer's device prohibits asymmetric (public-key) cryptography, as it is simply too slow [#]. However, symmetric (secret-key) cryptography is fast enough to run on the device, and we use this to our advantage below.

3 Security and Privacy Requirements

Extreme care must be taken when constructing a system that is used for financial transactions, as the very nature of such a system provides substantial reward for those who can attack it successfully. Before designing the actual protocol to be used by such a system, we must determine the types of security to be enforced:

Guarantee of Payment. The most obvious sort of security to be provided by the system is a guarantee of payment to the merchant. That is, when the customer indicates to the merchant his or her desire to make a purchase, the merchant must be guaranteed that the he or she will receive the funds promised by the customer. In a traditional system, this is done by using unforgeable banknotes from a central bank, or by verifying payment in real time with the customer's bank (as is done when using a credit- or ATM-card). It is interesting to note that although we desire a complete guarantee of payment, traditional credit or ATM-card systems cannot make this guarantee: because of introduced latencies and lack of complete connectivity, one can overdraw an ATM-card account or exceed the credit limit on a credit card.

Authentication of Merchant. Equally important, but considerably less obvious, is the verification of the merchant's identity to the customer: the customer must know that he or she is indeed dealing with the merchant that is expected. Without this guarantee, the customer can be tricked into delivering payment to a third party or an attacker.

This guarantee of security may seem foreign when compared with real-world financial transactions. This is due to its implicit, rather than explicit, presence in such transactions: the authentication of the merchant's identity and trustworthiness for transactions is accomplished with a combination of brand identity and difficult-to-forge physical objects — like large signs, buildings, and uniforms demonstrating the merchant's identity. These objects, which are both difficult and, typically, illegal to forge, demonstrate the merchant's identity, and brand identity demonstrates that merchant's trustworthiness in financial transactions.

In the world of secure electronic transactions, the merchant's identity and trustworthiness must be proved by external means. The premise that the merchant's identity is proved by difficult-to-forge objects is irrelevant when purchasing goods or services remotely, and some objects — like vending machines — simply are not terribly difficult to forge. Further, brand identity is only useful for merchants that are ubiquitous enough to have a pre-established presence in the consumer's mind; when dealing with new merchants, brand identity is of no use. Thus, electronic transactions must provide some other route to authentication of the merchant.

Privacy. A subsidiary, but equally important, consideration is that of privacy: no party to the transaction should know more than is necessary to complete the transaction. For example, despite existing practice, it is unnecessary for the customer to reveal his or her identity to the merchant — the merchant only needs to know that payment is being made, not from whom it comes. Similarly, the bank, although trusted, need not know what the customer purchased.

While privacy is an important goal, we may subjugate this goal to that of security when necessary — it is not critical for the system to function. Privacy rights may, after all, are more easily upheld than security at the political layers of a system.

4 Additional Goals

There are several additional goals we had when constructing our system:

Management. A key requirement for financial systems is *manageability*: there can always be attacks on the system from a higher layer (such as the merchant claiming that the goods have been delivered when, in fact, they have not); this demands the ability to reconstruct all records associated with a transaction, at some time in the future. This provides audit and tracing information for out-of-band management and higher-order management functions. Notably, manageability considerations in a system are often at odds with privacy; the easiest management occurs when all parties have complete knowledge of the all the details of the transaction.

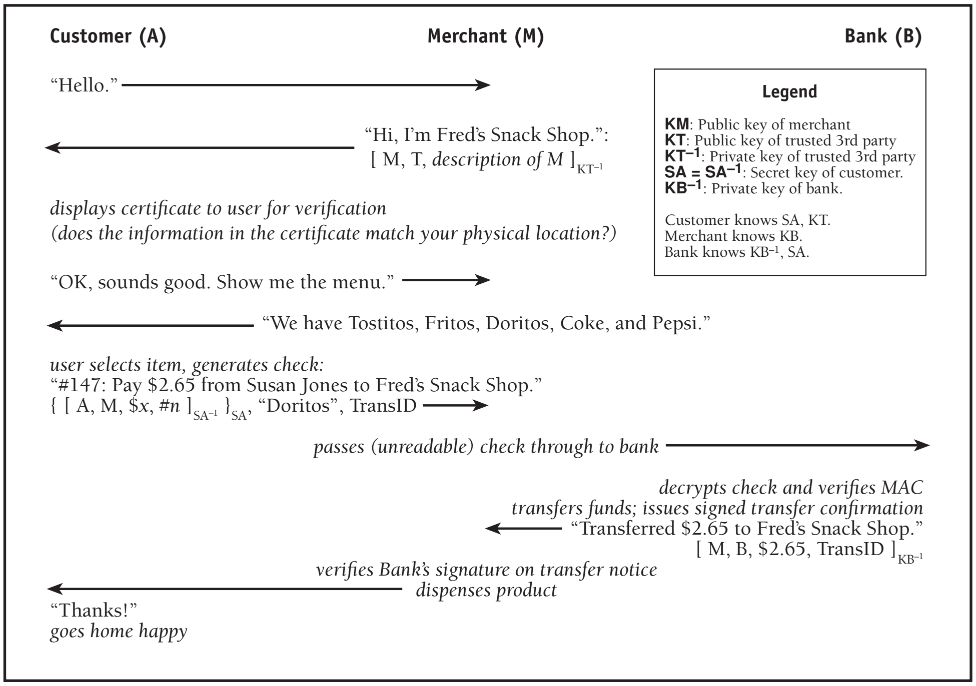
Efficiency. We desire a system that is efficient in cost. When making small purchases — from a few U.S. cents to a few tens of dollars — an overhead of even ten cents can have serious effects on the usability of the system. ATM cards appear to have a net cost to the merchant of about a quarter; this is the amount often charged back to the consumer in price-sensitive situations like gasoline sales.

Speed. We would like our system to be quick; indeed, it must be fast enough that it is usable for the quickest of transactions — a purchase from a vending machine or from a convenience store. This demands response times in the sub-second range, and optimally around a few tenths of a second. As response times increase, the delay becomes aggravating to the consumer and will make consumers tend to eschew use of the system.

5 Protocol and System Design

We present our protocol design below. The three parties to the transaction are the customer, at left; the merchant, in the middle; and the bank, at right. The customer communicates only with the merchant; the merchant acts as a proxy, passing the critical financial information on to the bank and receiving confirmation of its success.

We have introduced a fourth party, albeit one who is not directly involved, into the design of our system. This trusted fourth party is responsible for certifying the trustworthiness of merchants: by signing the public key of a merchant, this fourth party asserts that the merchant can be trusted to, upon receipt of



payment, deliver goods or services, and to settle disputes in an agreed-upon manner.

Below, we detail the important features of the protocol:

Key Management. Because the customer's device lacks sufficient computational power for asymmetric cryptography, we must use symmetric cryptography for any secure data generated by the customer. In our protocol, the customer shares a secret key, SA, with the bank. We believe this to be reasonable because the bank must already be entrusted with the customer's money, and thus is trustable with the secret key used solely to manipulate that money.

The merchant and bank, by contrast, have plenty of computational power available and thus can use asymmetric cryptography with ease. Thus, the merchant is issued a public–private key pair (KM, KM^{-1}), and the bank is issued a private–public key pair, (KB, KB^{-1}).

Finally, the trusted fourth party — the certifier of merchants — also has a public–private key pair, (KT, KT^{-1}).

Key Distribution. Key distribution is an issue in any cryptosystem; the use of symmetric-key cryptography only heightens its importance further.

The distribution of a secret key from the bank to the customer's device must be done every few months in a secure manner. We do not directly solve this problem in our protocol, but rather note that there are existing solutions — from a physically secure, trusted area where key transfer can take place to simply encrypting the key with a passphrase known only to the bank and customer**.**

The public keys used by the bank, merchant, and trusted fourth party may be distributed among those parties by any number of different PKI systems. Again, this is an issue with several well-known solutions, and we do not attempt to address it further here.

Finally, the customer must somehow acquire a copy of the trusted fourth party's public key. We assume that this is done by, again, known methods; for example, the bank can send to the customer the public key along with a MAC generated using the secret key that it shares with the customer.

Link Encryption. Link encryption provides authentication and secrecy of messages, and thus is, in general, a very desirable thing to have. However, link encryption is difficult to accomplish without using asymmetric cryptography. In particular, encryption of the customer–merchant link in our protocol is not possible, because the customer's device is not powerful enough to use asymmetric cryptography, and the customer and merchant do not, in general, share any secrets.

However, we *do* assume that all communications between the merchant and bank are made using full link encryption. Because both parties are powerful enough to use asymmetric cryptography, it is relatively trivial to set up an encrypted link (such as an SSL connection) between them.

Protocol Steps. Below, we summarize the steps of the protocol. Analysis of the security and privacy aspects of the protocol is delayed to the next section.

1. **Greeting and Merchant Authentication**. In the initial query, the merchant sends a certificate, signed by the trusted fourth party, containing his or her identity, the identity of the trusted fourth party, and any further details pertinent to a full description of the merchant (such as location).

   Because the customer has a copy of the trusted fourth party's public key, the customer can verify the authenticity of this certificate. The customer's device displays this certificate to the customer, requiring the customer to confirm that the data in the certificate matches the actual location and name of the merchant with whom the customer believes he or she is dealing.

2. **Item Selection, Check Generation**. After an exchange in which the customer is given a chance to select the desired product, the customer generates a "check".

   The check is the single most critical piece of the entire system. It represents almost exactly the same data as a traditional paper check: the identity of the customer, the identity of the merchant (payee), the amount to be paid, and a nonce or check sequence number (used to avoid replay attacks). A message authentication code (MAC) for the check is generated using the customer's shared secret, and the entire bundle is then encrypted using the customer's shared secret.

3. **Check Transfer and Forwarding**. Because the merchant does not share the customer's secret, he or she cannot decrypt the check. Instead, the merchant merely forwards the check on to the bank over a secure link.

4. **Funds Transfer and Verification**. The bank, upon receipt of the check, decrypts it, verifies the MAC, and makes the requested transfer. It then creates a verification statement containing the amount transferred and the account to which it was transferred, signs the statement using its private key, and sends the statement back to the merchant.

5. **Transfer Verification and Transaction Completion**. The merchant can verify the funds transfer verification, because he or she has access to the bank's public key. After verifying that he or she has received payment, the merchant gives the requested goods or services to the customer, and the transaction is complete.

6 Protocol and System Analysis

Now that the protocol has been outlined, we consider how it fulfills the goals set forth in previous sections.

Guarantee of Payment. This system provides an incontrovertible guarantee of payment to the merchant. Fundamentally, the bank issues the guarantee of payment to the merchant, not the customer; because the merchant trusts the bank (and indeed has an account with the bank), the signed statement by the bank that the merchant has been paid is as true a promise as the merchant can possibly get that funds have been paid. Neither the customer nor a malevolent outsider cannot forge the statement, as he or she does not have access to the bank's private key; neither can an outsider replay a verification statement, because each contains a unique transaction ID and can only be used once.

Authentication of Merchant. It is in the authentication of the merchant that the most subtle, and thus the most dangerous, protocol specifics are involved. The merchant sends a certificate, signed by the trusted fourth party, to the customer, who can verify its authenticity. However, because there is no link encryption, there is no way to guarantee that the customer is speaking to the merchant described in the certificate (or, indeed, that the customer is speaking with the *same* party throughout the transaction). However, the check that the customer issues is based *only* upon the signed information in the certificate.

It is here that the trusted fourth-party signer comes into play: the signing of a certificate by the fourth party is an indication that the merchant listed on that certificate can be trusted to correctly handle electronic transactions. A malevolent outsider can only substitute one existing certificate for another; new certificates cannot be forged from whole cloth. Thus, an outsider can only cause the customer to send funds to a different, but still trusted, merchant; this second merchant can be trusted, then, to return the funds to their rightful owner once the error is discovered (which will be immediately, when the verification from the bank is either never received or contains the wrong payee).

It is clear that a real-time, authenticated link between the customer and merchant is highly desirable: by connecting the portions of the transaction together and ensuring that the customer is actually speaking with the merchant named in the certificate, diversion of funds is impossible and greater security is achieved. However, without a (as-of-yet-nonexistent) secure secret-key distribution system, this requires asymmetric cryptography — which is currently beyond the reasonable computational power of our small device.

Privacy. In our current system, the following knowledge is imparted to the various parties:

The **Customer** knows the merchant's identity, the product purchased, and the cost of the product, all of which are desirable;

The **Merchant** knows the product purchased and the cost, both of which are desirable;

The **Bank** knows the customer's identity, the merchant's identity, and the amount transferred. All three are necessary in an account-based system, and, because the bank is trusted with funds, one can assume that it can be trusted not to reveal this information. Still, the fact that the bank knows all this information is less than desirable; we cannot solve this problem, however, without the use of electronic cash (ecash), which was beyond the means of this project in its limited time.

A malevolent outsider only learns the product purchased (it is sent in cleartext to the merchant), and the few things that can be learned via traffic analysis (specifically, the fact that the merchant was involved in a transaction — but not, importantly, the customer's identity). Without link encryption, the outsider cannot be prevented from knowing the product purchased, and prevention of traffic analysis is well beyond the scope of this paper.

Management. Management of transactional information can be achieved rather easily by using the transaction ID that is attached to all communications. If each party can be trusted to reveal correct information — which is a basic requirement for all such management — the information can be indexed and assembled using the transaction ID, and thus the entire pattern of communication and modifications can be traced.

Efficiency. Each party in this system is able to handle a great number of these transactions using few resources, thus enabling efficient implementations. The customer need only perform a basic MAC (hashing) and encryption algorithm; using fast symmetric-key encryption algorithms such as Blowfish or Twofish, this can be done with a minimum of processing power. The merchant need only perform a single public-key signature verification, which is quite cheap in terms of processing power. The merchant also needs to receive and forward the check on to the bank; as the check is likely to be considerably less than 1 KByte in length, this can be done using a standard analog modem or other such device. The bank, finally, need only perform a symmetric-key decryption and MAC check, both of which are fast; modify its internal state; and generate a short signed message to return to the merchant. All of these should be relatively trivial, especially in comparison with the amount of hardware a bank is likely to be able to provide.

However, there is yet one more factor that contributes to the efficiency of the system: by actually *guaranteeing* payment to the merchant by maintaining consistency of the customer's bank account, there is no room for customers to overdraft or intentionally defraud the bank or merchant. This is a source of the fees incurred by traditional ATM cards, and one that can be eliminated in our system.

Speed. Our system has the potential to work very quickly, even using cheap hardware. The critical path involves an encryption/decryption and MAC creation/verification of the check — all of which will be very fast, because the check is typically less than 512 bytes long; the signing and signature verification of the funds-transfer verification notice, which is slower but nonetheless still plenty fast on modern hardware; and four communication hops, two each from customer to merchant and from merchant to bank.

Using even cheap, commodity hardware, implementations of this system should have a round-trip time in the low hundreds of milliseconds, ensuring adequate response time for the customer.

7 Implementation Experiences

We have implemented this system on a small testbed to ensure its practicality and verify the design. Our hardware consisted of 3Com's Palm III (and the identical IBM Workpad) for the devices; the merchant and bank were implemented as Java programs — specifically, iSpace services in the Ninja internet-services architecture [#]. Communications between the device and merchant service were accomplished using the Palm III's built-in serial port; in future implementations we expect to use the built-in infrared communications port for greater ease of use.

While implementation efforts cannot prove the security of the protocol, they can demonstrate the basic viability of the architecture. We believe our implementation did exactly that: we achieved a working prototype that implemented the important parts of the architecture and was sufficiently efficient and quick to use in real-world situations.

Efficiency and Speed. While many security aspects of secure systems cannot be demonstrated to be valid by implementation, some of the non-security-critical aspects can be so demonstrated. In particular, our implementation showed that the protocol involved can be implemented efficiently and with a very low latency. While measuring quantitative speed proved to be too difficult in the time allotted, the transaction appeared to the user to be accomplished nearly instantaneously. Indeed, the slowest part of the transaction was the encryption and MAC generation for the check on the device, and even this was barely perceptible. Future advances in low-power hardware will decrease this speed further.

Beyond the stated goals, there were several additional interesting results of our implementation efforts:

Validation of the Ninja model. Although it is not the primary focus of this paper, we believe that our implementation nicely validated the Ninja model of the world. In particular, the devices we used were easily connected to the iSpace world of the bank and merchant, using the merchant itself as an "active proxy". It is also interesting that the merchant acted both as a transaction participant and as an "active proxy".

Creation of Infrastructure. During our implementation efforts, it became obvious that substantial effort was expended on problems orthogonal to the main cryptosystem: generation and parsing of communications between the device and merchant, key distribution to the device, and so forth. As it became evident that these problems would accompany any such system, we generalized the solutions sufficiently to provide a quicker implementation effort to any following researchers. The specifics of these efforts are, again, beyond the scope of this paper, but we intend for our implementation to be a valuable framework for any similar cryptosystems — and, indeed, any systems with a similar combination of devices and base systems.

8 Conclusions

Low-cost, pay-per-use services are an area of electronic commerce that can benefit from the efficiency, security, and speed of a lightweight framework for such commerce. Such frameworks have security goals in common with traditional electronic commerce, yet have an overall set of goals and constraints somewhat different. By tailoring a protocol specifically to these lightweight, pay-per-use services, we can achieve a high degree of security while maintaining overhead — financial and temporal — low enough to permit the system's use for very small transactions.

The coming era of ubiquitous computing opens an entirely new potential space for users, but also poses some new challenges. Specifically, we find that while it enables the use of hand-held devices — like the Pilot — to carry out lightweight, pay-per-use services, it also poses security challenges due to its inability to carry out operations in asymmetric cryptography. Using careful software design, these challenges can be surmounted to enable these new applications. The pay-per-use services we have implemented using the Pilot and Ninja-style base services are an example of this "new wave" of applications, and one that can serve as a framework for future applications.

9 Future Work

While we are satisfied with the current state of the system, there is yet considerable work to be done:

Customer–Merchant Link Encryption. Encrypting the customer–merchant link would provide a much better guarantee of security and privacy for our system. While our device cannot carry out full public-private key operations, it is possible to carry out the *faster* of these operations on the device. (Which specifically is faster is dependent upon the specific algorithm used.) Using this ability, it should be possible to establish a session key between the customer and merchant and encrypt the link. We see this as the next obvious step in development of the system.

Elimination of the Trusted Fourth Party. The trusted fourth party, used to assert merchant trustworthiness, is a fundamentally unnecessary part of the system, and one that is necessary only because of the absence of customer–merchant link encryption. Using a standard SSL-link certificate exchange would allow for elimination of this trusted fourth party and simplification of the system. We expect that once devices have adequate power to carry out full asymmetric cryptography, we will implement this modification.

Ecash Use. Electronic cash (ecash) provides a number of privacy benefits to its users; specifically, in our system, ecash would obviate the need for the bank to know the identities of the customer and merchant altogether. Modification of our protocol to accommodate ecash would be simple and straightforward.

10 Thanks

The authors wish to thank, in no particular order: David Wagner and Ian Goldberg, for their motivation, valuable insights into protocol design, and help with PalmOS programming; David Culler, for important insights into the contribution of our work; IBM's generous donation of Workpad devices to the department; and the entire Ninja project team, for provision of a powerful and fruitful programming environment.