Scaleable, Secure Cash Payment for WWW Resources with the PayMe Protocol Set

Michael Peirce
Donal O'Mahony

Abstract:

The use of the WWW as an electronic marketplace is increasing, and there is a need for a cash payment system that is scaleable, anonymous and secure. In this paper we examine two existing systems: Ecash and NetCash, discuss their strengths and weaknesses and propose a new system called the PayMe Transfer Protocol (PMTP). We show how it improves on existing systems, and illustrate its use with an example based on purchase of goods across the WWW.

Keywords:

Web payment, electronic cash, secure payment, scaleable payment, Internet payment mechanisms, security.

Introduction

The World Wide Web has potential to become a highly efficient electronic marketplace for goods and services. When payments are effected electronically, there is always a risk that organizations may resort to gathering information relating individuals with the amounts that they have spent, locations involved and types of good purchased. Misuse of such information can give rise to serious breaches of personal privacy[18]. If a payment system for the WWW is to receive widespread support, it must offer its users some form of protection against the gathering of such information. The most effective method of achieving this is to implement a form of electronic cash, where the coins being spent cannot be linked with their owner. This gives rise to a secondary problem in that since the coin is an electronic quality that is easily duplicated, such a payment system must guard against the coin being spent more than once. It should not be possible for an attacker to bypass the system or to falsely obtain monetary value from it.

At the time of writing, it has been estimated that there may be over 30 million users of the Internet spread across 96 different countries[12] using over 6.6 million host computers[15], and these figures are rising very rapidly. This means that an effective electronic payment system must be highly scaleable. In practice, the system must support large numbers of buyers and sellers affiliated to many different banks. The problem of detection of double spending is particularly acute, and solutions must be found that allow for large numbers of payments to take place without requiring unreasonably large databases to be maintained. In the following section, we discuss related work on two systems for electronic payment and go on to propose a new set of protocols that surmounts some of their inherent problems.

Related Work

Recently, two electronic cash systems, requiring no additional hardware such as smart cards, which can be used to make payments for WWW resources have been published.

The first, Ecash, is a fully anonymous electronic cash system, using numbered bank accounts and blind

signatures. The second, NetCash, uses identified electronic cash giving a more scaleable but less anonymous system.

Electronic cash is the electronic equivalent of real paper cash, and can be implemented using public-key cryptography, digital signatures, and blind signatures. In an electronic cash system there is usually a bank, responsible for issuing currency, customers who have accounts at the bank and can withdraw and deposit currency, and merchants who will accept currency in exchange for goods or a service. Every customer, merchant, and bank has its own public/private key pair. The keys are used to encrypt, for security, and to digitally sign, for authentication, blocks of data that represent coins. A bank digitally signs coins using its private key. Customers and merchants verify the coins using the bank's widely-available public key. Customers sign bank deposits and withdrawals with their private key, and the bank uses the customer's public key to verify the signature.

Ecash from DigiCash

Ecash[9,10] is a fully anonymous electronic cash system, from a company called Digicash, whose managing director is David Chaum, the inventor of blind signatures and many electronic cash protocols[1,3,4,5,6]. It is an on-line software solution which implements fully anonymous electronic cash using blind signature techniques.

The Ecash system consists of three main entities:

- Banks who mint coins, validate existing coins and exchange real money for Ecash.
- Buyers who have accounts with a bank, from which they can withdraw and deposit Ecash coins.
- Merchants who can accept Ecash coins in payment for information, or hard goods. It is also possible for merchants to run a pay-out service where they can pay a client Ecash coins.

Ecash is implemented using RSA public-key cryptography. Every user in the system has their own public/private key pair. Special client and merchant software is required to use the Ecash system. The client software is called a "cyberwallet" and is responsible for withdrawing and depositing coins from a bank, and paying or receiving coins from a merchant.

Withdrawing Ecash Coins

To make a withdrawal from the bank, the user's cyberwallet software calculates how many digital coins of what denominations are needed to withdraw the requested amount. The software then generates random serial numbers for these coins. The serial numbers are large enough so that there is very little chance that anyone else will ever generate the same serial numbers. Using a 100-digit serial number usually guarantees this. The serial numbers are then blinded using the blind signature technique[3]. This is done by multiplying the coins by a random factor. The blinded coins are then packaged into a message, digitally signed with the user's private key, encrypted with the bank's public key, and then sent to the bank. The message cannot be decrypted by anyone but the bank.

When the bank receives the message, it checks the signature. The withdrawal amount can then be debited from the signature owner's account. The bank signs the coins with a private key.

After signing the blind coins, the bank returns them to the user, encrypted with the user's public key.

The user can then decrypt the message, and unblind the coins by dividing out the blinding factor. Since the bank couldn't see the serial numbers on the coins it was signing there is no way to now trace these coins back to the user who withdrew them. In this way the cash is fully anonymous.

Spending Ecash

To spend Ecash coins, the user starts up their cyberwallet software and a normal Web client and then browses the Web till they find a merchant shop selling goods. The Ecash software can be used with any existing Web client and Web server software. A merchant shop is simply a HTML document with URLs representing the items for sale. To buy an item the user selects the URL representing that item. The following steps then occur as shown in Figure 1.

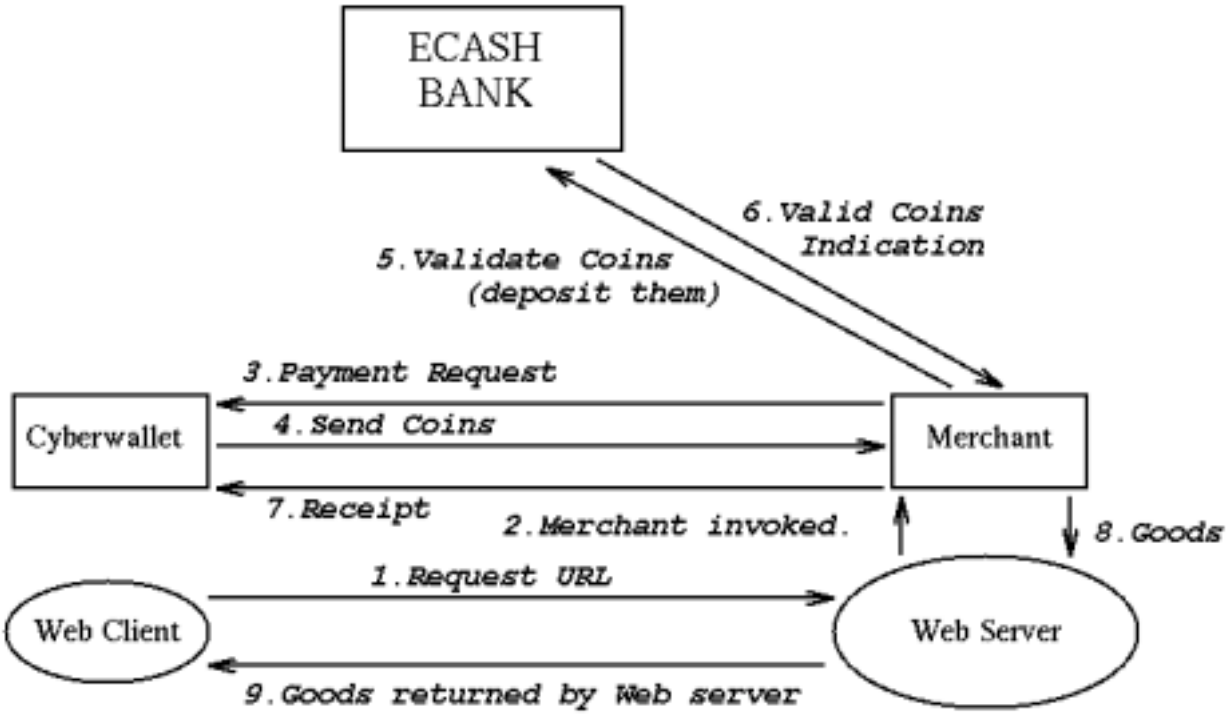

Figure 1 : Making a purchase with Ecash

1. The user's Web client sends a HTTP message requesting the URL to the Merchant's normal Web server. This URL will invoke a Common Gateway Interface (CGI) program[19].
2. The CGI program invoked will be the merchant Ecash software, and it will be passed details of the item selected encoded in the URL. The location of the buyer's host machine will also be passed in an environment variable from the server to the merchant Ecash software.
3. The merchant software, now contacts the buyer's wallet using a TCP/IP connection, asking it for payment.
4. When the cyberwallet receives this request, it will prompt the user, asking them if they wish to make the payment. If they agree, the cyberwallet will gather together the exact amount of coins and send this as payment to the merchant. The coins will be encrypted with the merchant's public key so that only the merchant can decrypt them:

{Coins}K[public,Merchant]

If they disagree or do not have the exact denominations necessary to make a correct payment, the merchant is sent a payment refusal message.

5. When the merchant receives the coins in payment, he must verify that they are valid coins, and have not been double spent. To do this he must contact the bank, as only the minting bank can tell whether coins have been spent before or not. Thus the merchant packages the coins, signs the message with his private key, encrypts the message with the bank's public key, and sends it to the bank.

   {{Coins}K[private,Merchant]}K[public,Bank]

6. The bank validates the coins by checking the serial numbers with the large on-line database of all the serial numbers ever spent and returned to the bank. If the numbers appear in the database then they are not valid, since they have been spent before. If the serial numbers don't appear in the database, and have the bank's signature on them, then they are valid. The value of the coins are credited to the merchant's account. The coins are destroyed, and the serial numbers added to the the database of spent coins. Thus coins are good for one transaction only. The bank notifies the merchant of the successful deposit.
7. Since the deposit was successful, the merchant was paid, and a signed receipt is returned to the buyer's cyberwallet.
8. The purchased item, or an indication of successful purchase of hard goods, is then sent from the merchant Ecash software to the Web Server.
9. The Web server forwards this information to the buyer's Web client.

Ecash client and merchant software is available for many platforms. Currently no real money is used in the system, but an Ecash trial[11] with 10,000 participants, each being given 100 "cyberbucks" for free has been running since late 1994. There are many sample Web shops at which to spend cyberbucks.

Advantages and Failings

The strengths of Ecash are its full anonymity and security. The electronic cash used is untraceable, due to the blind signatures used when generating coins.

By employing secure protocols using RSA public key cryptography, the Ecash system is safe from eavesdropping, and message tampering. Coins cannot be stolen while they are in transit. However, the protection of coins on the local machine could be strengthened by password protection and encryption.

The main problem with Ecash may be the size of the database of spent coins. If a large number of people start using the system, the size of this database could become very large and unmanageable. Keeping a database of the serial number of every coin ever spent in the system is not a scalable solution. Digicash plans to use multiple banks each minting and managing their own currency with interbank clearing to handle the problems of scalability. It seems likely that the bank host machine has an internal scalable structure so that it can be set up not only for a 10,000 user bank, but also for a 1,000,000 user bank. Under the circumstances, the task of maintaining and querying a database of spent coins is probably beyond today's state-of-the-art database systems.

NetCash

NetCash[13,14] is a framework for electronic cash developed at the Information Sciences Institute of the University of Southern California. Many of the ideas used in PayMe came from the NetCash proposal. It uses identified on-line electronic cash. Although the cash is identified there are mechanisms whereby coins can be exchanged to allow some anonymity. The system is based on distributed currency servers where electronic checks, such as NetCheque[16,22] can be exchanged for electronic cash. The use of multiple currency servers allows the system to scale well.

The NetCash system consists of buyers, merchants, and currency servers. An organisation wishing to set up and manage a currency server obtains insurance for the new currency from a central certification authority. The currency server generates a public/private key pair. The public key is then certified by being signed by the central authority. This certificate contains a certificate ID, name of the currency server, currency server's public key, issue date and an expiry date, all signed by the central authority:

{Certif_id,CS_name,K[public,CS],issue_date,exp_date}K[private,Auth]

The currency server mints electronic coins, which consist of:

1. **Currency Server Name:** Identifies a currency server.
2. **Currency Server Network Address:** Where the currency server can be found. If this address is no longer in use, a name server can be queried to find the current address.
3. **Expiry Date:** Limits the state that must be maintained by each currency server.
4. **Serial Number:** Uniquely identifies the coin.
5. **Coin Value:** Amount coin is worth.

The coin is signed with the currency server's private key:

{CS_name,CS_addr,exp_date,serial_num,coin_val}K[private,CS]

The currency server keeps track of the serial numbers of all outstanding coins. In this way double spending can be prevented by checking a coin's serial number with the currency server at the time of purchase (or exchange). If the coin's serial number is in the database it has not been spent already and is valid. When the coin is checked the serial number is then removed from the database. The coin is then replaced with a new coin (coin exchange).

An electronic cheque can be exchanged with a currency server for electronic coins. The currency server is trusted not to record to whom the coins are issued. To further aid anonymity a holder of coins can go to any currency server and exchange valid coins for new ones. The currency server does not know who is exchanging coins, only the network address of where they are coming from. By performing the exchange and by choosing any currency server to do this with, it becomes difficult to track the path of the coins. If a currency server receives coins that were not minted by it, it will contact the minting currency server to validate those coins.

Figure 2 shows how a buyer uses NetCash coins to purchase an item from a merchant. In this transaction the buyer remains anonymous since the merchant will only know the network address of where the buyer is coming from. NetCash assumes that the buyer has or can obtain the public key of the merchant,

and that the merchant has the public key of the currency server.

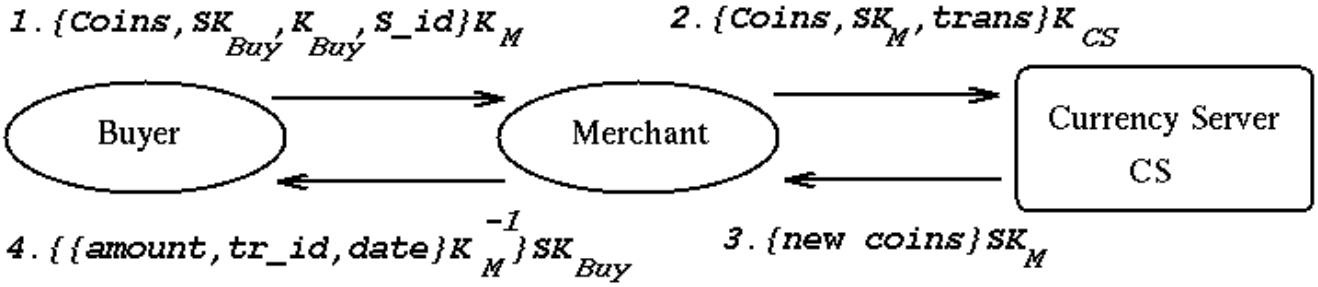

Figure 2 : Purchasing from a merchant using NetCash

Implementation details of how the NetCash protocols might be linked with applications such as the Web are not available, but it could be done in a similar fashion to Ecash using an out-of-band communications channel. The transaction consists of the following four steps, starting from when the buyer attempts to pay the merchant:

1. The buyer sends the electronic coins in payment, the identifier of the purchased service(S_id), a freshly generated secret key (SK[Buyer]), and a public session key (K[public,Buyer]), all encrypted with the Merchant's public key, to the merchant.

   {Coins,SK[Buyer],K[public,Buyer],S_id}K[public,Merchant]

   The message can't be eavesdropped or tampered with. The secret key is used by the merchant to establish a secure channel with the buyer later. The public session key is later used to verify that subsequent requests originate from the buyer who paid for the service.
2. The Merchant needs to check that the received coins are valid. To do this he sends them to the currency server to be exchanged for new coins or for a cheque. The merchant generates a new symmetric session key SK[Merchant] and sends this along with the coins and the chosen transaction type to the currency server. The whole message is encrypted with the server's public key so that only it can see the contents:

   {Coins,SK[Merchant],transaction_type}K[public,CS]

3. The Currency Server checks that the coins are valid by checking its database. A valid coin is one whose serial number appears in the database. The server will then return new coins or a cheque to the merchant, encrypted with the merchant's session key:

   {New_coins}SK[Merchant]

4. Having received new coins (or a cheque) the merchant knows that he has been properly paid by the buyer. He now returns a receipt, signed with his private key and encrypted with the buyer's secret key:

   {{Amount,transaction_id,date}K[private,Merchant]}SK[Buyer]

   The buyer can then use the transaction identifier and the public session key to obtain the service

purchased.

This is the basic purchase protocol used in NetCash. While it prevents double spending it does not protect the buyer from fraud. There is nothing to stop the merchant spending the buyer's coins without providing a receipt.

Extensions to the protocol are detailed in [14]. These are more complex and give protection against fraud for both the merchant and buyer. There are also mechanisms to allow the merchant to be fully anonymous to the buyer. Partially off-line protocols where the bank does not need to be contacted during a purchase are also described. These however rely on the buyer contacting the currency server beforehand, and knowing who the merchant is at that time. They use a time window in which the coins are only valid for certain short lengths of time. Full technical details are given in [14].

The advantages of NetCash are that it is scalable and secure. It is scalable since multiple currency servers are present and security is provided by the cryptographic protocols used. Possible disadvantages of the system are that it uses many session keys and in particular public key session keys. To generate a public key of suitable length to be secure takes a very large amount of time compared with that involved in generating a symmetric session key. This could compromise the performance of the system as a whole.

NetCash is not fully anonymous, unlike Ecash. It is difficult but not impossible for a currency server to keep records of who it issues coins to and who it receives them back from. The ability to exchange coins and use any or multiple currency servers increases the anonymity of the system.

A NetCash system is currently being implemented, but no details are given as to how it will be linked with applications such as the Web. NetCheque will be used to provide cheques which can be used to buy coins or which can be issued when coins are traded in.

Discussion

The two payment systems outlined each have their strengths and weaknesses. Ecash is a fully secure system that provides for very strong anonymity. The use of banks within the system reflects current practice in non-electronic payment systems. Successful operation of the Ecash system depends on the maintenance of a central database of all coins ever issued within the system. If it were to become accepted as a global payment system, this would quickly become a major problem.

NetCash uses identified coins with multiple currency servers, and thus, while anonymity is maintained, there is only a requirement to keep track of all currency currently in circulation. This makes for a much more scaleable solution to the payment problem. NetCash is also fully secure, and achieves this using protocols that are quite complex in nature.

The PayMe Protocol Set

In an attempt to combine the best features of the two systems described, a new payment system called the *PayMe Protocol Set* was devised. A major goal was to preserve as much of the anonymity provided by Ecash while adopting many of the features of NetCash that allow it to scale to large numbers of users with multiple banks. In the following sections, we will discuss the overall design of the protocol set and

work through an example of a network payment. Since this paper concentrates on payment for WWW resources, detailed coverage will be given of both the currency representation and the protocol primitives used during a Web transaction.

The PayMe system and protocol set are now presented. Many of the design ideas are based on a close examination of systems such as NetCash, Ecash and other related systems such as Magic Money[7] and Netbill[20,21]. In this way PayMe is a collection of the successful parts from existing systems, minus the failings of those systems.

PayMe is an on-line electronic cash system. The entities involved are banks and users. Users can be either buyers or merchants but each has the same functionality. They can make payments, accept payments, or deal with the bank. Each bank mints its own identified electronic cash with serial numbers. Double spending of coins is prevented by the bank maintaining a database of coins in circulation. This scales better than the blind signature electronic cash approach. Any user in the PayMe system can accept payments and make payments. Merchants can receive payments for selling Web goods but they can also make payments to the buyers. This can be used for making refunds or in pay-out services.

A simple model showing the basic functionality of the PayMe system can be seen in Figure 3.

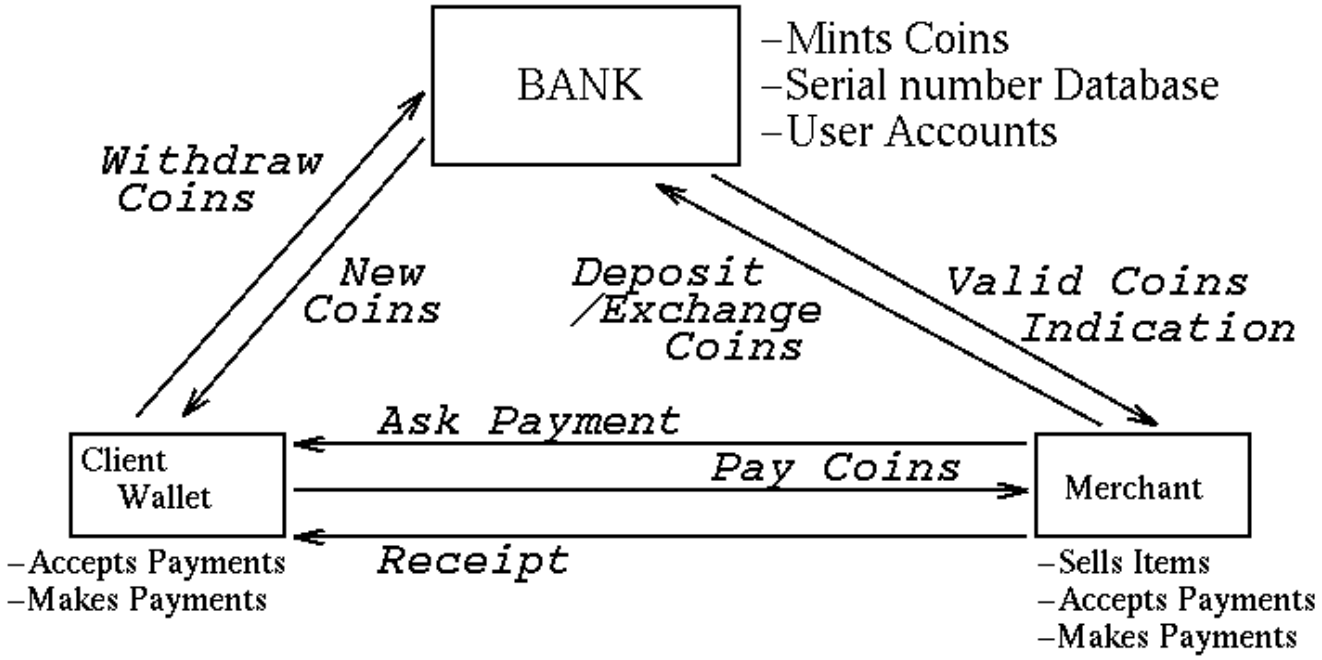

Figure 3 : Basic functionality of the PayMe system

Both symmetric and public-key cryptography are used. Each entity has its own public/private key pair. It is a stand-alone system which has been tailored for use with the Web.

The PayMe system uses its own secure communications protocol, the PayMe Transfer Protocol(PMTP), to communicate between entities. This provides security and a means of communicating out-of-band, that is, outside the Web's HTTP protocol. This approach was adopted to allow a full prototype to be developed that could eventually be used with any emerging Web security standard.

PayMe Currency

Coins are the pieces of data that represent monetary value within the system. The coins are digitally signed by the bank using public key cryptography to make them valid currency. Each coin has a serial number which is entered into the bank's database when the coin is minted. Coins have fields for the coin value, serial number, bank id, bank host name and port number, and expiry date.

When these five fields are put together and signed with the bank's private key, a valid coin is created. An example coin is of the form:

{10 MIK1234 BANK1 bank.cs.tcd.ie.8000 18-12-98}K[private,BANK1]

Here the coin is worth 10, its serial number is MIK1234, the userid of the bank's public key is BANK1, the bank is located at port 8000 on the machine bank.cs.tcd.ie, and the coin expires on 18th December 1998.

A bank within the PayMe system mints coins, maintains a database of the serial numbers of coins in current circulation to prevent double spending, and manages the accounts of merchants and buyers.

PayMe Transfer Protocol (PMTP)

PMTP is the set of secure messages designed to provide the communications necessary in the PayMe system. It uses both symmetric and public-key cryptography. PMTP consists of six request-response message types.(1) For each of the six message types there are three different possible message identifiers. There is one request message identifier and two different response message identifiers. These have been called request, response and refusal respectively. A request is where the receiver is being asked to perform an action. A response message identifier indicates that the action has been performed and the message body contains the results of that action. A refusal is where the receiver refused to perform the action, and the message body may contain a reason for this refusal.

The first three messages are used by a bank account owner to withdraw or deposit coins, or obtain a bank statement from the bank for that account.

- **Withdraw Coins**
  Requires an account identifier, matching account name, account password, and amount, digitally signed by the account owner.
- **Deposit Coins**
  Attempts to deposit coins into a bank account. The bank will check that the coins are valid before crediting the account. The account identifier, name, and digital signature are required to make a deposit. A deposit can be done with any bank with which the user has an account. If the coins are not minted by that bank then the minting bank will be contacted to validate the coins. Banks have accounts with other banks and in this way records are kept of how much each bank owes another. These accounts could then be settled using a real-world inter-bank clearing mechanism.
- **Request Bank Statement**
  Returns a bank statement for an account. A digital signature is required to authenticate the account owner.
- **Exchange Coins for new ones**

Any user who holds valid coins from a bank, can exchange the coins for new ones. The process for doing this is anonymous, but it is still secure. During the exchange the bank only knows the network address of where the coins are being sent from. If the coins it receives are valid it will return new ones in exchange. It is not necessary to have an account at a bank to exchange coins. For efficiency an exchange must be done with the bank that minted the coins.

Either a buyer or merchant can use this mechanism to help hide their identity. When a user withdraws coins from a bank the bank could record the numbers on the coins and who it gave them to. Then when a merchant later deposits the coins the bank could check to whom it issued the coins. In this way the spending habits of a user could be recorded.

However, if during a purchase a merchant exchanges the coins rather than depositing them, then the bank does not know who has performed the exchange. Either the merchant or buyer, or even another trusted third party could perform this exchange to "launder" the money, making it more difficult to trace spending habits.

- **Ask for payment**

The last two messages are used between a user and another user such as a merchant. The ask_payment message is used to ask a buyer for a payment amount. During a purchase a buyer remains anonymous to the merchant. Ideally the buyer should have obtained the merchant's public key before the purchase. However the merchant's public key is also sent within the payment request. There is some risk involved with this, since an attacker could replace the merchant's key with his own. The user is given the choice to accept a new merchant key in this way or not. If the user already holds the merchant's public key, then this is compared with the one received in the payment request as part of the procedure to authenticate the merchant. The ask_payment request and refusal messages are shown in Table 1.

Table 1 : Ask_payment Messages

Parms: amount(integer).
ask_payment_request: $\{\textbf{PAYMENT\_REQ} < amount > : K_{Merchant} : \{< nonce >\} K^{-1}_{Merchant}\}$
ask_payment_response: Same as pay_coins_request. A successful response to a merchant's request for payment, is an attempt to pay that merchant.
ask_payment_refusal: $\{\textbf{PAYMENT\_REFUSAL} < amount > : < nonce > \} K_{Merchant}$

- **Pay coins**

Attempt to pay coins to a merchant. The buyer remains anonymous to the merchant in this transaction. The merchant only knows the network address of the buyer. The specification of the pay_coins messages are shown in Table 2.

Table 2 : Pay_coins Messages.(2)

Parms: Coins(Money_bag).
pay_coins_request: $\{\textbf{PAY\_COINS\_REQ} < Money\_bag > : < symmetric\_session\_key > : < nonce > \} K_{Merchant}$
pay_coins_response: $\{\textbf{PAY\_COINS\_RESPONSE} < Receipt > : < nonce > \}\mathrm{SK}$
pay_coins_refusal: $\{\textbf{PAY\_COINS\_REFUSAL} < reason > : < nonce > \}\mathrm{SK}$

The full specifications of all PMTP messages can be found in [17]. The parameters will often be generated automatically by the PayMe software. The address of where to send the message to, also needs to be given.

PMTP Security

PMTP messages are secure from attacks using eavesdropping, message tampering, replay, and masquerading techniques.

Eavesdropping Prevention

An attacker cannot see the contents of a PMTP message because the message is either:

- Encrypted with the public key of the receiver. Only the private key can decrypt the message.
- OR encrypted with a symmetric session key which has been distributed securely. The session key was distributed by sending it in a public-key encrypted message.

The only exception to this is the ask_payment_request message. Since the buyer is to remain anonymous this message is transmitted in cleartext.

Message Tampering Prevention

Any encrypted message cannot be tampered with, since it will not be possible to decrypt it after it has been changed. By using message digests, a digitally signed message cannot be tampered with.

Replay Prevention

A nonce is used within each PMTP message to ensure that the message can be used for one occasion only, and to prevent a replay of that message. It ensures that the message must come from a specific network address and within a small time window. If an attacker can forge the IP network address to be the same as that of the message sender, then he could possibly replay the message within the short time frame that it is valid. To help prevent this the software keeps track of all recently received nonces and will not accept two messages with the same nonce such as a replayed message would have.

Masquerading Prevention

Where possible all messages are authenticated with a digital signature. Bank withdrawals also require the password of the bank account. In the anonymous messages where a digital signature is not possible,

knowledge of a symmetric session key is used. The network address within the nonce prevents an attacker at another site from masquerading as the message sender at the original network address.

Private Key Protection

The private key of a user is stored on file at the user's local site. It is encrypted with a secret passphrase. If the user's account is broken into, this prevents the attacker being able to access the private key. Without this private key any cash stored locally cannot be decrypted, and PMTP messages cannot be sent.

PayMe with the Web

PayMe was tailored for use with any Web client or server. To purchase an item a user starts up both their PayMe Wallet and any Web client. They browse the Web until they find a merchant shop, which will be presented by a HTML document. A combination of PMTP messages are used in a purchase transaction, as shown in Figure 4.

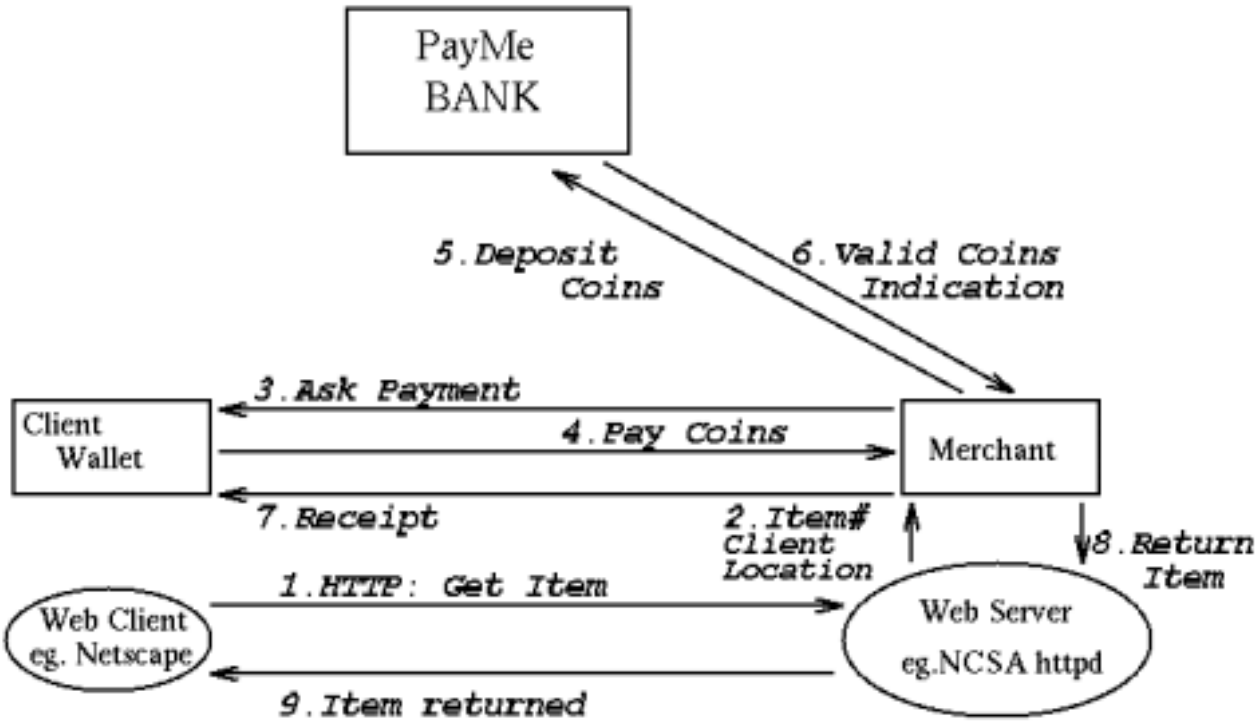

Figure 4 : Purchasing a Web service with PayMe

1. To purchase an item (information, hard goods, or pay-out service) a URL is selected representing that item. When selected the URL causes the Web server to automatically start up a merchant's Wallet software. This is done using the Common Gateway Interface(CGI)[19].
2. The Wallet is passed the item details and the network address of the requesting Web client. Additional information, such as a shipping address for hard goods, can be passed through a Web form if required.
3. The Wallet then looks up the cost of the item and contacts the buyer's Wallet software asking for

payment. This is a PMTP ask_payment_request.

4. The buyer will be notified of the request. He will then either refuse (ask_payment_refusal) or accept (pay_coins_request) the payment request. If he accepts the Wallet selects the coins needed to make the exact payment and sends them to the Merchant.
5. The Merchant validates the coins by either anonymously exchanging them for new coins or depositing them into a bank account. For efficiency, if an exchange is performed it must be done with the bank that minted the coins. A deposit can be done with any bank with which the merchant has an account. The minting bank checks the serial numbers of the coins with those in its database. If a serial number is not present in the database the coin is *not valid* and is rejected. If the serial numbers are present then the coins are valid.

   Having performed the check the bank then removes the serial numbers from the database, thereby invalidating the coins. This must be done because otherwise the same coins could be presented many times and they would always be valid. The merchant is given new coins in replacement, or the amount can be credited to his bank account.

6. The merchant will receive an indication from the bank as to whether the coins were valid. A valid coin indication will be new coins in an exchange (exchange_coins_response), and a deposit acknowledgement (deposit_coins_response) with a deposit.
7. For a good payment the merchant then issues a signed receipt to the buyer (pay_coins_response).
8. The purchased item is sent from the merchant to the Web server.
9. The Web server then forwards this to the buyer's Web client.

Payments must be made with the exact amount. No change can be given since this could compromise anonymity if a merchant colluded with the minting bank.

Implementation

A prototype was implemented in a C++/Unix environment on a Sun workstation cluster. An extended version of PgpTools[8], a set of C functions which provide low-level PGP[23,24] packet functionality in memory, was used to implement the cryptographic functions. It uses RSA to provide the public key cryptography, and IDEA for the symmetric key cryptography. Full technical details of the implementation can be found in [17]. Pgptools is subject to similar patent restrictions as PGP.

Coin backups and log files are maintained to increase the fault tolerance of the system. In this way the chance of losing coins, and hence monetary value, is kept to a minimum if any of the entities crash.

PayMe could be used for schemes other than just monetary payment. A coin within the system could be used to represent a unit of CPU time, or connection time to a limited resource, in order to provide resource sharing in an institution. Jobs which require units of CPU time could be submitted or initiated through the Web where the merchant would be the CPU host requesting the PayMe coins representing time on that CPU.

For applications where anonymity is important the exchange coins mechanism can be used to anonymously exchange the coins with a bank preventing the bank knowing who now holds the new coins. In an environment where anonymity is not necessary or desirable the banks involved can be configured to refuse any requests to exchange certain coins, such as those representing CPU time. In this way the bank can record to who it issues the coins and who then deposits them, knowing for certain that

no anonymous exchange has taken place. Thus the configuration of the bank can control the anonymity available to its users.

Discussion

Taking the best features of existing systems, a new payment mechanism using electronic cash for use with the Web has been designed and implemented. It offers the following desirable properties:

1. **Security**
   The system was designed to be secure from fraud. The possibility of an attacker being able to bypass the system or falsely obtain value in it was minimized. PMTP was designed to provide secure communication. Security steps were also taken to protect coins, the private cryptographic keys used, and the accounts at the bank.
2. **Scalability and Reliability**
   Multiple banks can be used in the PayMe system, giving no central point of failure. The simple PMTP protocols can be used for inter-bank communication as well as with regular users. Electronic cash where only a database of the serial numbers in current circulation is used, much like in the NetCash system. In this way it is much more scaleable than Ecash. The serial numbers of every coin ever spent need not be maintained. Secondly the serial numbers can be short, unlike the long serial numbers of about 100 digits, necessary to prevent serial number collisions when using blind signatures.
3. **Usable by all**
   It is important that the system can be used by anyone provided they have the money to pay for the items they wish to buy. No credit card numbers are used, since not all Internet users, for whatever reasons, hold valid credit cards. In theory anyone who wants to can buy PayMe electronic coins and have an account at a PayMe on-line bank.
4. **Usable with any Web client or server software**
   PayMe can be used with any Web client or server software and it is not limited to any specific product or HTTP version. As many new innovations and advances in Web technology are designed and released, it is important that a Web payment mechanism can be used with all of these. By using its own secure out-of-band protocol, PayMe can be used with both current and emerging Web technology and protocols.
5. **Payment for information, hard goods, and pay-out services**
   Web information of any type such as text, images, audio streams or video can be purchased using PayMe. Hard goods can be paid for through the Web, using forms. The PayMe client software used by a buyer can also receive payments. In this way pay-out services can be used.
6. **Hardware independent**
   No special hardware, such as smart cards, are required to use PayMe. The system can be used right now using only software, and this is more suited to the global Internet where it would take time for users to obtain and begin to use new hardware.
7. **Limited Anonymity and Privacy**
   It is desirable to prevent a database being built with full details of every purchase made by an individual. Some anonymity can be provided by the system by anonymously exchanging coins with a bank, similar to NetCash's exchange mechanism[14]. A buyer will also remain anonymous to a merchant during a purchase transaction, as only the buyer's network address will be known.

The system does not offer offline operation. It was not possible to fulfill all the above requirements and

at the same time remove the need for a bank to be contacted during a purchase transaction. However it is felt that with the trend towards faster, and more reliable global networks, offline operation is not required. Secondly, on the Internet where it is easy to hide one's identity, it is not acceptable to use an off-line electronic cash system where fraud will only be detected after it has occurred, as in [2].

The final implemented system provides a secure and scalable means of paying for all types of Web services. It would seem to be more scaleable than the fully anonymous Ecash system, and more efficient than the complicated protocols and use of both symmetric and asymmetric session keys of NetCash.

Conclusions

We have examined two existing means of effecting anonymous electronic payment across networks and looked at their strengths and weaknesses. We then presented the design of PMTP, a hybrid of these two approaches that offers a fully secure, scaleable anonymous payment system. We have shown how this can be combined with WWW client and server software allowing payment to occur on an out-of-band link as users browse the Web. Only a payment system with these properties will allow the Web to be used as an electronic marketplace without compromising the privacy of its users.

References

[1] J.Bos and D.Chaum. Smart Cash: A Practical Electroinc Payment System. Technical Report, CWI-Report: CS-R9035, August 1990.

[2] D.Chaum, A.Fiat, and M.Naor. Untraceable Electronic Cash. In Advances in Cryptology - Crypto '88 Proceedings, Springer-Verlag, Berlin, 1990, pages 319-327.

[3] D.Chaum. Blind Signatures for Untraceable Payments. In Advances in Cryptology - Crypto '82 Proceedings, Plenum Press, 1983, pages 199-203.

[4] D.Chaum. Security without Identification: Transaction Systems to Make Big Brother Obsolete. Communications of the ACM v.28, n.10, October 1985, pages 1030-1044.

[5] D.Chaum. Online Cash Checks. In Advances in Cryptology, EuroCrypt '89 Proceedings, Springer-Verlag, Berlin, 1989, pages 288-293.

[6] D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. In Advances in Cryptology - Crypto '88 Proceedings, Springer-Verlag, Berlin, 1990, pages 319-327.

[7] Product Cypher. Magic Money Digital Cash System, ftp://ftp.csn.org, 1994.

[8] Product Cypher. The PGPTools Security Toolkit, ftp://ftp.csn.org, 1994.

[9] DigiCash. Ecash, http://www.digicash.com/, 1994.

[10] DigiCash Press Release. World's First Electronic Cash Payment over Computer Networks, May 27th 1994.

[11] DigiCash Press Release, Ecash Trial is Now Worldwide, January 6 1995.

[12] Larry Landweber. International Connectivity. Version 14 - June 15, 1995. ftp://ftp.cs.wisc.edu/connectivity_table/Connectivity_table.ps Computer Sciences Dept. University of Wisconsin - Madison, 1210 W. Dayton St., Madison, WI 53706, U.S.A.

[13] Gennady Medvinsky and B.Clifford Neuman. Electronic Currency for the Internet. Electronic Markets Vol 3. No. 9/10, October 1993, pages 23-24.

[14] Gennady Medvinsky and B. Clifford Neuman. NetCash: A design for practical electronic currency on the Internet. In Proceedings of the First ACM Conference on Computer and Communications Security, November 1993.

[15] Network Wizards, Menlo Park, California, http://www.nw.com/. Internet Domain Survey, July 1995.

[16] B. Clifford Neuman and Gennady Medvinsky. Requirements of Network Payment: The NetCheque Perspective. In Proceedings of IEEE Compcon'95, San Francisco, U.S.A., March 1995.

[17] M. Peirce, PayMe: Secure Payment for World Wide Web Services, B.A. (Mod) Project Report, Computer Science Department, Trinity College Dublin, Dublin 2, Ireland. May 1995.

[18] Jeffrey Rothfeder, Privacy for Sale, Simon & Schuster, 1992.

[19] Tony Sanders, Ari Luotonen, George Philips, John Franks, and Rob McCool. The CGI Specification. http://hoohoo.ncsa.uiuc.edu/cgi/interface.html

[20] Marvin Sirbu and J. Douglas Tygar. Netbill: An Electronic Commerce System Optimized for Network Delivered Information and Services. In Proceedings of IEEE Compcon '95, March 1995.

[21] J. D. Tygar. NetBill: An Internet Commerce System Optimized for Network Delivered Services. Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 1995.

[22] University of Southern California Chronicle. The Check is in the E-mail, usc-chronicle-941107, November 1994.

[23] Phil Zimmermann. PGP User's Guide, Volume I: Essential Topics. Phil's Pretty Good Software, ftp://ftp.pegasus.esprit.ec.org/pub/arne/pgpdoc1.ps.gz. October 1994.

[24] Phil Zimmermann. PGP User's Guide, Volume II: Special Topics. Phil's Pretty Good Software, ftp://ftp.pegasus.esprit.ec.org/pub/arne/pgpdoc2.ps.gz. October 1994.

Footnotes

(1) A request/response message is where a client sends a request to a server and the server sends a reply message to that request.

(2) A Money_bag is an implementation structure which holds coins.

About the Authors

Michael Peirce [http://www.cs.tcd.ie/www/mepeirce/mepeirce.html]
Computer Science Department
Trinity College
Dublin 2
Ireland
Email: Michael.Peirce@cs.tcd.ie

Michael Peirce graduated with a B.A.(Mod) in Computer Science from Trinity College Dublin, Ireland in June 1995. His final year dissertation concerned the design of a new scalable anonymous electronic payment mechanism for the purchase of goods and services on the WWW. He has acted as the maintainer of a highly popular WWW page on Electronic Payment since December ’94. He was senior technical reviewer for the popular "Internet: The Complete Reference", and chief researcher for "The Internet Yellow Pages" (1st and 2nd editions), both published by Osborne/McGraw-Hill. He has completed a summer internship with Hitachi Research Laboratory in Dublin. Currently he is pursuing a Masters degree, working in the area of mobility, with the Networks and Telecommunications Research Group (NTRG) at Trinity College, Dublin.

Donal O’Mahony [http://www.cs.tcd.ie/www/omahony/omahony.html]
Computer Science Department
Trinity College
Dublin 2
Ireland
Email: Donal.OMahony@cs.tcd.ie

Donal O’Mahony received B.A., B.A.I. and Ph.D. degrees from Trinity College Dublin, Ireland. After a brief career in industry at SORD Computer Systems in Tokyo and IBM in Dublin, he joined Trinity College as a lecturer in Computer Science in 1984. He is author of many papers and articles on networking and security and coauthor of "Local Area Networks and their Applications" published by Prentice-Hall. At Trinity, he coordinates a research group working in the areas of Networks and Telecommunications. Within this group, projects are ongoing in X.500, Electronic Data Interchange (EDI), Networked multi-media data streams and Network Security. Dr. O’Mahony has acted as consultant to government and private industry organizations across Europe on a wide variety of projects involving strategic networking issues.