Chapter 4: Credit card–based systems

Overview

Credit card schemes have been in use as a payment method since the early 1960s and the two major international brands, VISA and MasterCard, are household names all over the world. The VISA brand grew from a scheme launched by the Bank of America, which was subsequently licensed by Barclaycard in the United Kingdom in 1966. By the middle of 1995, this organization, owned by its 18,000 member financial institutions, had issued more than 420 million cards and is now accepted by more than 12 million merchants in 247 countries.

Its principal competitor, MasterCard, is of comparable size with 13 million merchants in 220 countries and 22,000 member organizations. As Table 4.1 shows, they collectively account for a total of amost 800 million cards issued and nearly (U.S.) $1,300 billion of sales each year.

Since their inception, the introduction of more payment options has led to the development of a number of different *payment card* schemes. These include the following:

- *Credit cards*, where payments are set against a special-purpose account associated with some form of installment-based repayment scheme or a revolving line of credit. Cards typically have a spending limit set by the card issuer and the interest rate levied on unpaid balances is typically many times the base lending rate.
- *Debit cards* are linked to a checking/savings account. Normally, a payment cannot be made unless there are funds available to meet it. In effect, this type of payment can be considered a paperless check.
- *Charge cards* work in a similar way to credit cards in that payments are set against a special-purpose account. The principal difference is that the entire bill for a charge card must be paid at the end of the billing period. Often, there is no associated spending limit.
- *Travel and entertainment cards* are charge cards whose usage is linked to airlines, hotels, restaurants, car rental companies, or particular retail outlets.

Table 4.1 Usage of Two Main Branded Credit Cards in 1995

	VISA		MASTERCARD	
Region	**Sales Volume billions of dollars (U.S.)**	**Number of Cards (millions)**	**Sales Volume billions of dollars (U.S.)**	**Number of Cards (millions)**
U.S.	358.4	228.1	202.4	174
Europe	262.4	81.2	not available	53.5
Asia-Pacific	91.6	73	116.2	72.5
Canada	36.8	18.6	not available	not available
Middle East, Africa	5.6	2.3	5.5	2
Latin America	23.6	21.4	19.1	21.2
Totals	778.4	424.7	470	338.7

Combined	US $1248.4B Sales	763.4 million cards

With the exception of debit cards, where funds transfer takes place at the moment of payment, the above schemes all operate in a similar fashion. Figure 4.1 shows the actors involved. Banks that belong to the card association may act as *card issuers* to their personal or business clients. This will involve the provision of a card and maintenance of a credit card account for that individual, to which transactions can be posted as they occur.

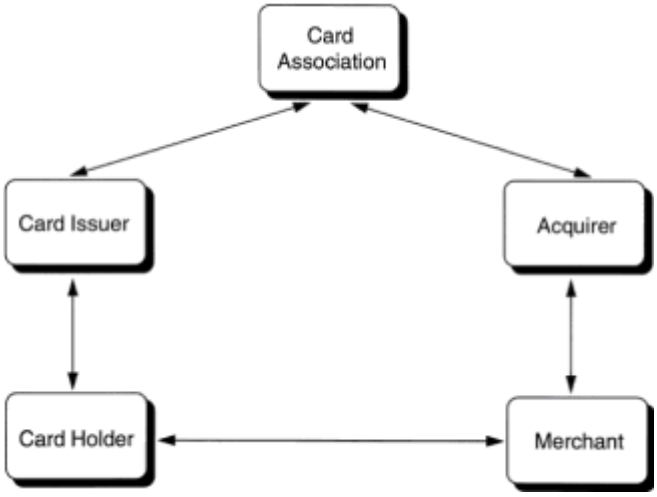


Figure 4.1 Entities involved in a conventional credit card transaction.

Another or the same bank will act as an *acquirer* for clients of theirs who wish to accept credit card payments. This will usually involve providing equipment and/or software to process payments at the merchant's premises. Arrangements to perform online verification of transactions, as well as the policy for requiring online verification, is set by the acquirer. This may involve the setting of a *floor limit*, where any transaction exceeding this limit requires an online check as to the card status.

In a typical purchase, a merchant will capture the cardholders details at the point of sale. Depending on the policy in force, the transaction may be completed straight away or an online check may be made. Batched transactions are later sent to the acquirer for processing.

.1 Mail order/telephone order (MOTO) transactions

For many years now it has been possible to make payments with credit cards without requiring the buyer and merchant to be colocated. Credit card companies have for some time allowed orders to be taken either by post or by telephone. These orders are referred to as mail order/telephone order (MOTO) transactions, and special rules have been imposed by the card companies on how these transactions are processed.

Normally, cardholders are asked to supply additional information, such as their name and address, that can be used to verify their identity. If goods that require physical delivery are being ordered, they must be dispatched to the address associated with the card. This gives limited protection against bogus orders. Since there is no cardholder signature involved, the processing rules allow the buyer to opt out of any transaction if they claim that they did not agree to the purchase. Clearly, this increases the risk borne by merchants.

Although there are more possibilities for fraud associated with this type of ordering, it is still a very popular form of payment, and it is clear that the benefits outweigh the fraud risks involved.

4.2 Unsecured network payments

Using credit cards to make payments across computer networks has similar associated risks as are experienced with MOTO transactions. Attackers eavesdropping on network

traffic may intercept messages and capture credit card details as well as any associated verification information (e.g., name and address). Because of the distinctive structure of credit card numbers, with their inbuilt check digits, programs can be written to scan a data stream for occurrences of such patterns. The data stream could be either an intercepted transmission, a file on disk, reclaimed disk space on a shared system, or even the stream of keystrokes produced by someone typing at their workstation.

What makes the risks considerably higher than MOTO transactions is the speed with which transactions can be conducted. If merchants are processing orders electronically, then fraudsters can generate vast numbers of orders before the fraud is detected. The global scope of the network means that transactions can be carried out at locations that are far removed from where the card was issued. Thus the gap between the card details being intercepted and blacklist notifications reaching all relevant merchants can be very significant.

Despite the increased risk, the absence of commonly accepted network payment schemes in a period when the Internet was rapidly expanding meant that many people resorted to this method of effecting payment. Some buyers adopted simple security measures such as spelling out the numbers of the credit card, or splitting the number across multiple messages. At the merchant end, the orders were processed in the same way as a MOTO transaction, using existing point of sale clearing systems.

4.3 First Virtual

One of the earliest credit card–based payment systems launched for the Internet was the product of a company called *First Virtual Holdings, Inc*. In October 1994, the company commenced operation of a payment system called the VirtualPIN that did not involve the use of encryption. The goal was to allow the selling of low-value information items across the network without the need for special-purpose client software or hardware to be in place. The system is not entirely fraudproof, but in the context of its target market this is not of such great importance.

Both merchants and buyers are required to register with First Virtual (FV) before any transactions can take place. The First Virtual server also has an involvement in every transaction and at intervals will lodge the proceeds to the merchant's bank account.

A buyer registering with FV forwards his credit card details and electronic mail address to FV and in exchange will receive a pass phrase, called a VirtualPIN. The initial part of this exchange can take place across the network, with the user filling in a WWW form and inventing a pass phrase. FV acknowledges this and adds a suffix to the pass phrase to form the VirtualPIN. The next step involves the buyer making a telephone call to FV to tender his credit card number. This allows FV to establish a link between the VirtualPIN and the credit card without ever using the credit card number on the network.

Merchants must go through a similar registration process where they give their bank details to FV and are given a merchant VirtualPIN. The normal method of transferring the bank details is to send a conventional check drawn on the bank account associated with the merchant. FV takes all account identifier information from the check itself. Once this is done, the merchant can request FV to process transactions from registered FV customers and, after deducting a per-transaction charge, deposit the funds in the merchant's bank account using the conventional bank automated clearing house (ACH) service.

Figure 4.2 shows a buyer using FV to make a purchase. Initially, the buyer browses the FV InfoHaus(FV Web server) or another Web server, where a FV merchant is selling goods. The buyer selects the item she wishes to purchase. The buyer is asked to enter her FV account identifier (Virtual PIN), which is forwarded to the merchant. The merchant checks that this account identifier is valid by querying the FV server. This can be done in a number of ways, ranging from manual queries to automated dialogues with an FV server. If the VirtualPIN has not been blacklisted, then the merchant will deliver the information to the buyer, either by e-mail, WWW reply, or other means.

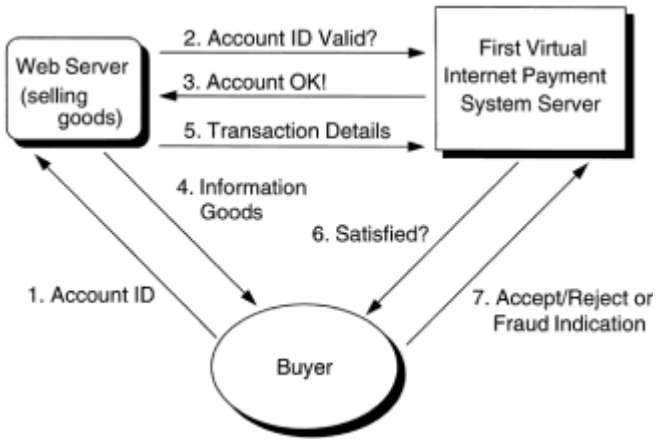


Figure 4.2 Buying with First Virtual.

The merchant forwards information about the transaction, including the buyer's VirtualPIN, to the First Virtual Internet payment system server. No payment has yet been made, since the system is based on a "try before you buy" philosophy. Accordingly, the next step is for the FV server to send electronic mail to the buyer asking if the information was satisfactory.

There are three possible replies to this request:

- *Accept*, in which case the payment proceeds;
- *Reject*, indicating that the goods either were not received or that the buyer is not happy to pay for them;
- *Fraud*, which means that these goods were not ordered by the buyer. Upon receipt of this message, the FV server will immediately blacklist the VirtualPIN.

At the end of every 90 days, the buyer's credit card account is billed for the charges that have accumulated during the time period. The merchant's checking accounts are then credited with payments for items sold. FV performs the accounting for both the buyer and merchant, taking a percentage of the transaction as commission.

It is quite clear that if a VirtualPIN becomes compromised by attackers eavesdropping on network traffic, bogus purchases can be made from then until such time as the VirtualPIN is blacklisted. Since payment authorization requests are sent to the buyer by e-mail, this time period could range from a few minutes to perhaps a day or so. Also, denial of service or masquerade attacks on the e-mail system could prolong this period quite substantially.

In addition, a stolen credit card number could be used to set up VirtualPINs associated with e-mail addresses controlled by the attacker, which may also allow long periods where bogus transactions can be carried out.

The exposure to fraud outlined above is of lesser importance if the payment system is used for information items. In this context, although a fraud has been committed, the seller will have lost a sale rather than incurring a large financial loss. Experience with the first year of operation of the system has shown a very low fraud rate. The overall model of how credit card payments work has been modified somewhat by the FV system. From the card company's point of view, FV takes on the role of merchant in that they are the ones that establish the relationship with the acquirer, and the value of all transactions is credited to their account in the first instance before being distributed using ACH bank transfers.

Perhaps the major advantage of the FV system is its simplicity. Since it makes no use of encryption, there are no export problems, and the simple exchanges that make up the protocol mean that no special software is needed at the front end, and the backend software is not complex.

Its major disadvantage is that before either merchants or buyers can use the system, they must preregister and have either a bank account (in the case of a merchant) or a credit card (in the case of a buyer). There are, however, no other qualifications demanded of merchants, in contrast with the stipulations typically made by credit card acquirers, and this makes the system more attractive for merchants who are likely to have limited turnover.

4.4 Collect all relevant information (CARI)

CARI [1] is a unique and simple system that allows hard goods to be ordered by credit card through the World Wide Web. It was designed by a team of technologists at Information Technology Partners (ITP), Milford, CT, USA. Like the FV system described above, credit card numbers are not present on the Internet. The CARI designers believe that consumers are reluctant to transmit their credit card numbers over the Net, regardless of the security or encryption provided. For this reason, the real credit card information is obtained by telephone using a "voice robot."

4.4.1 Virtual credit cards

To use the system, a consumer must first obtain and activate a virtual credit card (VCC). This is simply a random number assigned by CARI that will map to a consumer's real credit card number and details, as shown in Figure 4.3. It serves much the same purpose as the VirtualPIN used by FV. The virtual credit card is protected by a short personal identification number (PIN) that prevents entering another user's VCC number by accident during a purchase.

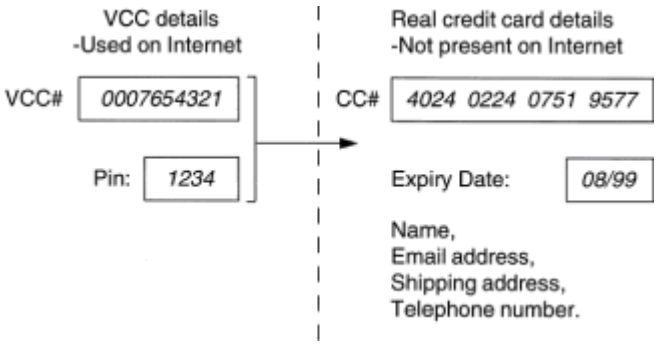


Figure 4.3 The virtual credit card mapping.

Once the virtual credit card has been set up, it can be used to order goods at a merchant's shop on a "CARI-connected" Web server. A CARI-connected Web server is one that can pass details from the Web server to the CARI system, as shown in Figure 4.4. The VCC and PIN are passed to the merchant in a normal Web form at the time of purchase. This allows CARI to work with all Web browsers and Web server software.

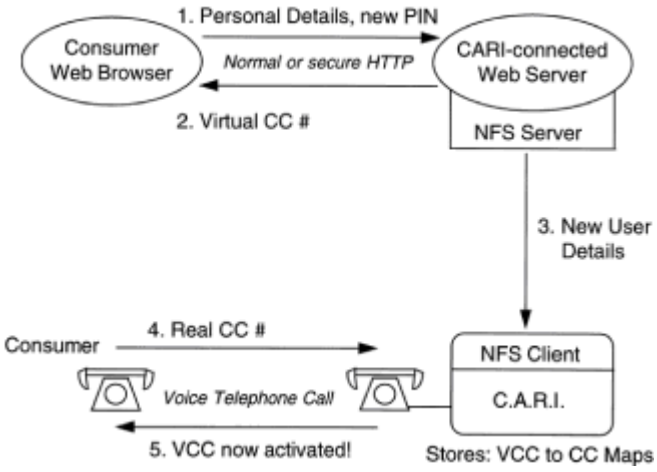


Figure 4.4 Setting up a virtual credit card.

4.4.2 Setting up a virtual credit card

To obtain a virtual credit card number, the user enters some personal details using a Web form at the CARI-connected Web server. This usually includes their name (to match the one on the real credit card), e-mail address (for confirmation of purchases), shipping address (to send goods to), and telephone number. Having chosen a PIN, they are then assigned a virtual credit card number as shown in Figure 4.4. Normally, none of this

information is encrypted since normal HTTP is used. However, provided both parties had secure HTTP capabilities (such as SSL, described later), where the Web's HTTP connection is encrypted, the information could be protected.

The new user details are now automatically transferred to the CARI machine, a PC-compatible system not accessible from the Internet. CARI obtains the information from the Web server using NFS [2], a network filesystem protocol. The new user details are placed in a file on the NFS exported drive, from which CARI collects them. Since NFS is not a secure protocol [3], some of the user details are encrypted so that only the CARI machine will be able to decrypt them. The Web server, or any other Internet host, cannot access any information on the CARI machine.

NFS is used to provide a simple one-way only link between the CARI machine and the Web server. Any other file exchange protocol such as ftp could have been used to obtain the user details from the Web server. The CARI designers found NFS to be the most suitable for the task.

To activate the VCC, users must give their real credit card details to CARI by telephone. A voice robot (basically a sophisticated telephone answering system) obtains the information by telephoning the user's number or by awaiting a call from that user. Once the real card is verified, the matching VCC is activated and the user can start shopping.

4.4.3 Independent CARI systems

In a typical CARI system configuration there will be a single Web server from which the CARI PC can obtain orders and new user registration details. Many merchant storefronts may appear as individual pages on this Web server. Each merchant's Web page will be capable of accepting orders using a virtual credit card. New-user registration for a virtual credit card will be performed through the same Web server. A single VCC is usable at all merchants in the same CARI system; that is, on the same CARI-connected Web server. In this setup, virtual credit cards are usually free to consumers, and merchants only pay for the Web space on the server. NetResource [4] offer this service.

Alternatively, a large organization with a high order volume may wish to purchase the CARI system for use with their own Web server. This solution does not scale very well since VCCs are valid only within one independent CARI system. There is no mechanism for CARI PCs to communicate with each other to exchange account details.

4.4.4 A virtual credit card purchase

A user places an order by sending the VCC, PIN, and order details to the Web server where the merchant's shop resides, using a Web form, as shown in Figure 4.5. CARI then collects the order from the Web server and verifies it. The order information, including the customer's real credit card number, is then forwarded to the merchant via fax, secure ftp, encrypted e-mail, or a dial-up line. The merchant system must have the ability to process the credit card purchase. Typically, this will require a merchant agreement with a credit card company.

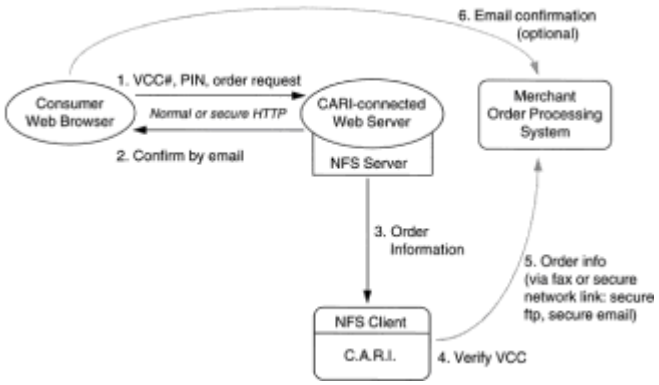


Figure 4.5: Purchasing using a virtual credit card.

If the HTTP connection between the buyer's Web browser and the Web server is not encrypted, an attacker can intercept the virtual credit card number and PIN. However, since goods must be sent to the shipping address associated with the VCC, the damage an attacker can do is limited to causing merchandise that has not been ordered to arrive at the real user's home. To help prevent this, e-mail confirmation can be required from

the user before shipping, much like in the FV system.

CARI is best suited for selling hard goods (physical goods) that are actually sent to a user's address. Due to the one-way NFS link between the Web server and CARI, information cannot be returned easily via the Web to the user. However, once a merchant has received and processed a valid order she may be able to provide purchased information or services to the user via e-mail.

In the future, it is planned for the CARI system to work with bank account details rather than credit card information to provide a payment instrument equivalent to an electronic check.

4.5 The secure socket layer (SSL)

Both FV and CARI effect payment using the transfer of a semisecret quantity (VirtualPIN or VCC) without the use of cryptography. An alternative approach is to securely transfer the credit card details across the network and then treat the transaction in the same way as a mail order/telephone order (MOTO) transaction.

For this to occur, two basic security services are required. First, some means must be found to encrypt the buyer-to-merchant communications link so that attackers cannot intercept the credit card details. Second, the merchant must be authenticated in some way to prevent attackers posing as merchants in order to capture card details. Although, it would be desirable for the merchant to have some assurance that the person tendering the card details is in fact the cardholder, this is not strictly necessary in a MOTO type of transaction.

The security services alluded to above can be provided using the secure socket layer (SSL) [5,6] which is a general-purpose protocol designed to be used to secure any dialogue taking place between applications communicating across a "socket" interprocess communications mechanism, though its primary use to date has been to enable secure credit card transactions on the WWW. The protocol was developed by staff at Netscape Corporation in late 1994 and progressed as an Internet draft standard late the following year. At the time of writing, it has reached version 3.0 and is being progressed as an Internet standard. It is in widespread use by merchants selling goods across the Internet to buyers using credit cards with no preregistration required. A protocol called Private Communications Technology (PCT) [7], offering the same facilities and having many similarities, is proposed by Microsoft Corporation. It is not yet clear whether either one of these proposals or some form of hybrid will succeed in becoming an Internet standard.

Parties to an SSL exchange identify themselves by producing certificates that link their *name* to a public key. No trust hierarchy is specified, so software agents taking part in SSL dialogues must be initialized with the certificates of certification authorities that are trusted. Much network commerce taking place today uses a product called Netscape Commerce Server, which can establish HTTP dialogues secured by the SSL with Netscape World Wide Web client software. Merchants using the commerce server will apply for a certificate from one of a small number of certification authorities whose public keys are preconfigured into all Netscape WWW clients. This enables them to authenticate themselves to clients in advance of payments being made.

The SSL protocol is transparent to the application using it. Once both sides are equipped with an SSL implementation, application data should pass through the secure socket in the same way as it would a normal (insecure) socket. Secured applications can exist side by side with normal applications. For example, calls to port 80 of a machine running a WWW server can expect to have a dialogue with a WWW server in an unsecured fashion, whereas calls to port 443 of the same machine will talk to the server over a secure socket connection.

Figure 4.6 shows the major components of the SSL protocol. When a socket connection is being set up, the *handshake* protocol does some initial work to establish the identities of the parties and negotiate cryptographic parameters for the session. Thereafter, application data is manipulated in accordance with these parameters and sent in *Application_Data* packets across the underlying socket connection. If the need arises to change the parameters midsession, then the *Change_Cipher* primitives come into play. Similarly, any problems that may occur are dealt with by the *alert* protocol.

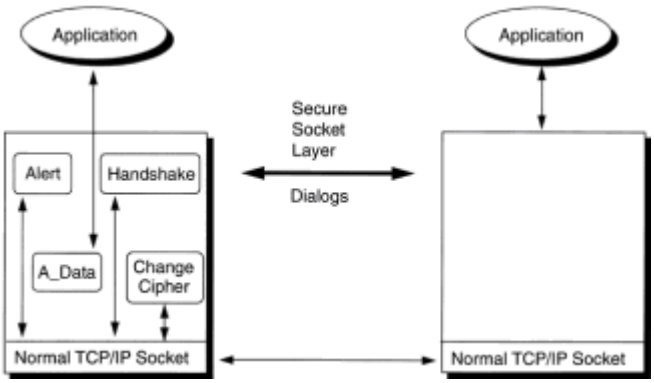


Figure 4.6 The main protocol components of the secure socket layer (SSL).

Within the *Handshake* protocol, there are a wide variety of options depending of whether the client, the server, or both are to be authenticated, and which encryption and key-exchange algorithms should be used. In the following description, we will concentrate on the one most commonly used to effect electronic payment. In this configuration, only the server side is authenticated and subsequent payment dialogues must be encrypted.

Figure 4.7 shows a typical handshake process. When the client application initiates a connection to the server, the SSL layer emits a Client_Hello message. This contains 28 bytes of data generated by a secure random number generator in addition to a list of client-supported cryptographic and compression methods listed in order of preference. A unique session ID is also established that can be used to allow this session to be resumed later in a subsequent connection.

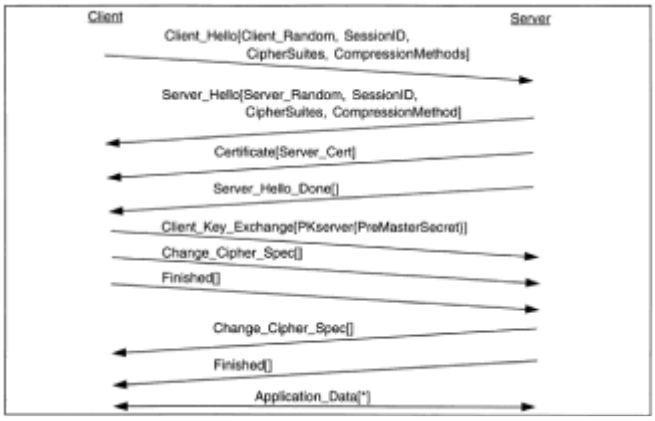


Figure 4.7: The main exchanges in the handshake protocol.

The server picks a cipher suite and a compression method from those offered and sends this back to the client in the Server_Hello message. The server also generates a random value that must be different from, and independent of, that included in the Client_Hello message. This is sent as part of the Server_Hello message. Following this, the server sends a Certificate message that contains a list of X.509 version 3 certificates beginning with the it's own certificate and including all certs as far as the root of the certification hierarchy. This part of the exchange is terminated with a Server_Hello_Done message.

Using the certificate path, the client can now extract the public key of the server and verify it's authenticity by following the trust path as far as the root. It computes a quantity called the *PreMasterSecret.* This is 48 bytes long and consists of 2 bytes giving the protocol version and 46 bytes of randomly generated data. The client encrypts this with the server's public key and sends it in a Client_Key_Exchange message. This PreMasterSecret contains all of the information necessary to secure the SSL dialogue, and after this message is sent, both the client and server perform an identical set of computations (given below) on it to generate the *MasterSecret:*

```
MasterSecret=MD5(PreMasterSectre + SHA('A'+PreMasterSecret+
ClientHello.Random+ServerHello.Random))+
MD5(PreMasterSecret+SHA('BB'+PreMasterSecret +
ClientHello.Random+ServerHello.Random))+
MD5(PreMasterSecret+SHA('CCC' +PreMasterSecret+
ClientHello.Random+ServerHello.Random))
```

This computation combines the PreMasterSecret with the random quantities sent in the Client_Hello and Server_Hello messages using concatenation (<F"MathematicalPi 1""P11.5>1), the message digest algorithm MD5, and the secure hash algorithm (SHA). Once the MasterSecret has been generated, a similar computation, shown below, is performed repeatedly to generate a stream, called a KeyBlock, until sufficient material has been generated for the partitioning process that follows:

```
KeyBlock=MD5(MasterSecret+SHA('A'+MasterSecret+ServerHello.Random+ClientHello.Random))
ClientHello.Random))
+MD5(MasterSecret+SHA("BB"+MasterSecret+
ServerHello.Random+ClientHEllo.Random))
+MD5(MasterSecret+
SHA('CCC'+MasterSecret+ServerHEllo.Random+
ClientHello.Random))+[…]
```

Depending on the cipher suite agreed to in the negotiating process, key lengths for signing and encryption will vary. Figure 4.8 shows the sequence in which these are extracted from the KeyBlock. The first two quantities (client and server Write_MAC_Secret) will be used to generate message authentication codes (MACs) for messages. The remaining material is used to provide the encryption key (Write_Key) and initialization vector (IV) for whatever symmetric algorithm is used for bulk encryption of the dialogue. If there is material remaining after all keys have been generated, it is simply discarded.

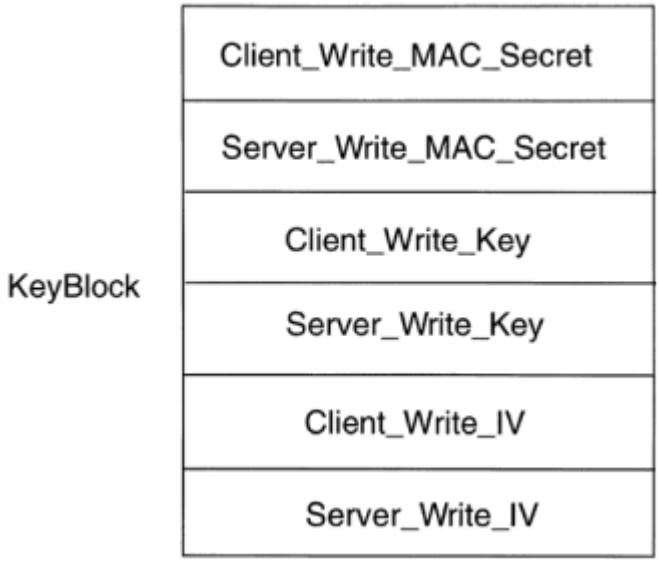


Figure 4.8 Partitioning the KeyBlock to obtain the individual keys.

At this point, both sides are equipped with all the necessary quantities to implement the agreed cryptographic regime. The client signals this by first sending a Change_Cipher_Spec message to indicate a changeover to the specification that has just been negotiated. This is followed by a Finished message, which is sent under this new specification. To signal its agreement, the server sends a similar Change_Cipher_Spec and Finished message pair.

At this point, the handshake protocol has terminated, the client and server application processes will be informed that a socket connection has been initiated, and they will start to exchange user data. This data is encapsulated in Application_Data units that vary in form depending on the cipher specification in force.

Figure 4.9 shows how user data to be sent on the secure socket is fragmented first into fragments of up to 2^{14} bytes (16 KB) in size. If both ends have agreed to use compression, then this fragment is passed through the agreed upon algorithm before proceeding. Depending on the cipher specification in force, the sender of data will want to do one of two things. Either she will want to produce a GenericStreamCipher unit sending the data in cleartext with a MAC appended, or a GenericBlockCipher will be produced where the data is fully encrypted as well as having a MAC appended. The length of the MAC field will depend on which algorithm is used to generate it. Similarly, the PAD fields included in the GenericBlockCipher serve to bring the size of the overall data unit up to some multiple of the block size associated with the ciphering algorithm.

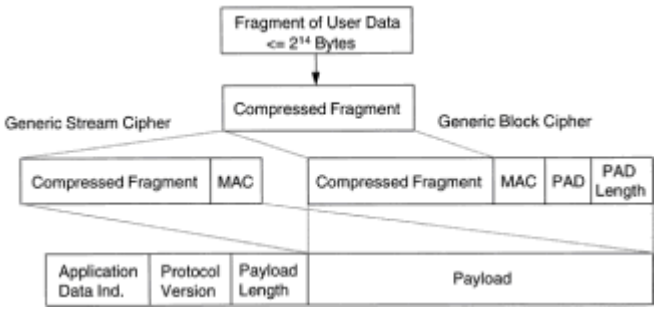


Figure 4.9 Encapsulating user data into application data protocol data units.

The cipher suites that can be used in an SSL connection are given names that are of the form:

SSL_KeyExchangeAlg_WITH_BulkCipherAlg_MACAlgorithm

where the algorithms used for key exchange, bulk enciphering, and generation of message signatures are individually specified. Some examples include

SSL_RSA_WITH_NULL_MD5
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_RSA_WITH_IDE A_CBC_SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DH_ANON_EXPORT_WITH_DES40_CBC_SHA
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA

It can be seen from the above that a large number of different algorithms are catered for in the SSL version 3.0 specification. At the time of writing, the bulk of electronic payment is being carried out using the predecessor (version 2.0) of this protocol, and in this situation, the RSA algorithm is used for the KeyExchange phase (Client_Key_Exchange message). WWW clients used within the United States use 128-bit RC4 to encrypt the messages, while export versions of the same software use a 40-bit variant of the same algorithm.

When SSL is used to effect payment, the merchant operates an SSL-enabled WWW server loaded with a certificate signed by an authority trusted by the WWW client software. The client software will request a WWW page whose address begins with *https* rather than the usual *http*. This indicates that SSL is to be used to secure the http dialogue. The server first authenticates itself, bulk encryption parameters are established, and the buyer's credit card details are sent in a WWW form to the server, where they can be decrypted and processed. Eavesdroppers will be unable to access the encrypted network traffic, and the merchant processes the credit card details in the same way as for a MOTO order.

4.6 CyberCash

CyberCash, Inc., [8] of Reston, VA, USA, was founded in August 1994 to provide software and service solutions for secure financial transactions over the Internet. The CyberCash Secure Internet Payment System, which uses special wallet software, enables consumers to make secure purchases using major credit cards from CyberCash-affiliated merchants. The CyberCash payment system was launched in April 1995, and by mid 1996 over half a million copies of the wallet software were in circulation. The system is used mainly by merchants to sell hard goods.

CyberCash intends to provide a complete solution for all types of Internet payments. Its CyberCoin electronic cash system is covered in Chapter 5, and as this book goes to press, a *PayNow* electronic check scheme has been launched. The description below is limited to the credit card payment system initially deployed.

Figure 4.10 gives an overview of the CyberCash system. A gateway server links to the existing financial infrastructure. It is connected to the Internet on one side and to many banks and bank card transaction processors on the other side. Purchase messages containing a consumer's credit card details are forwarded through this gateway from a merchant at the time of purchase. The actual credit card purchase is authorized and captured in the existing banking network. The results of the transaction are forwarded back through the CyberCash gateway to the merchant. If the transaction was successful, the merchant can then ship the goods to the consumer. CyberCash is not an acquirer,

issuer, or bank, but provides the gateway as a means of securely passing messages between the Internet and the banking networks and vice versa.

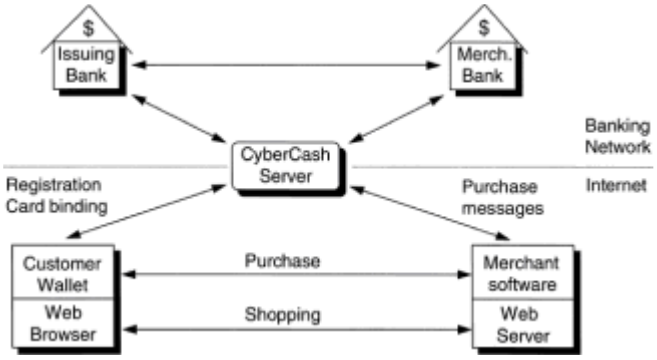


Figure 4.10 CyberCash model.

Much like other credit card payment systems, the CyberCash payment protocol [9] is concerned only with payment messages and not other electronic commerce protocols such as shopping.

4.6.1 CyberCash wallet

The CyberCash wallet is the application software used by a consumer to make purchases with their credit card. It aims to make purchases as transparent as possible to the user by hiding the details of the payment steps and messages during a purchase. The wallet software runs alongside any Web browser software.

The software uses 56-bit DES and 768-bit RSA for protecting the card details stored on the user's hard drive and in the CyberCash payment protocol. Since the encryption is only used to protect the financial data, the software has been approved for international export from the United States. This allows CyberCash to be global, and no U.S. banking presence is needed to use the system. Both CheckFree, CompuServe, and America Online have adopted the CyberCash software to provide Internet payment solutions to their customers. While their versions of the wallet may differ slightly, the payment protocols and security used are identical to the original CyberCash wallet.

4.6.2 CyberCash persona

Every CyberCash user chooses a unique CyberCash ID and pass phrase. This ID is registered with the CyberCash payment server and maps to the user's public/private key pair.

The ID and pass phrase are used to unlock the wallet, where the card data and secret keys are stored in encrypted form. The ID also acts as a mapping between an identified user and that user's public key within the system, much like in a certificate except that there is no third-party authorization signature. Finally, the ID and pass phrase can be used to perform an "emergency close-out" in the case of fraud. Any further attempted purchases using the consumer's credit cards are blocked once the close-out has been initiated by contacting CyberCash with the correct pass phrase.

4.6.3 A CyberCash purchase

Having completed shopping at a merchant's Web site, the consumer clicks on the "Pay" button at that site. The merchant returns a summary of the item, price, and a transaction ID through the consumer's Web browser. This message causes the CyberCash wallet to be launched on the consumer's machine, and it is passed the purchase details.

The buyer chooses the credit card that he wishes to use to pay with from those cards which he has already registered with the software (described in a later section). Having agreed with the order details presented, he then clicks on the wallet's Pay button. This initiates the CyberCash payment protocol. The card details are securely sent to the merchant. The merchant authorizes and clears the payment with the financial network via the CyberCash payment server. The merchant returns an unsigned receipt to the buyer.

Figure 4.11 shows the payment steps in more detail. The financial data is always encrypted and the merchant never sees the card data, thereby eliminating some risk of merchant fraud. Digital signatures are used by all three parties (the cardholder,

merchant, and payment gateway) for authentication and nonrepudiation during a purchase. Each message in the payment process is now briefly examined.

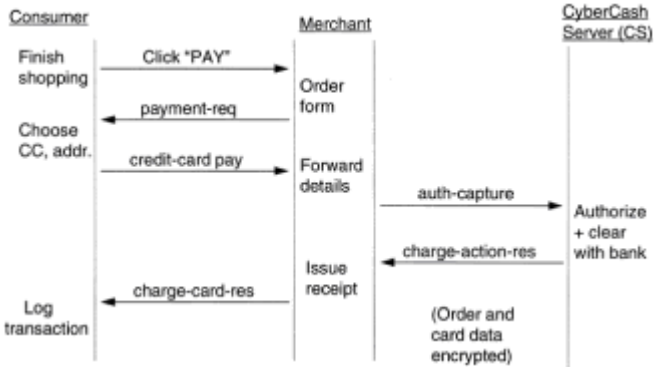


Figure 4.11 Payment steps in a CyberCash purchase.

4.6.3.1 Payment-Req (PR)

This is the message initially sent from the merchant to the cardholder to launch the CyberCash wallet software. It contains a summary of the order signed by the merchant. The signature is verified later by the CyberCash server and not by the actual cardholder.

4.6.3.2 Credit card payment (CH1)

This is the payment message from the cardholder to the merchant. It includes the card data encrypted with the public key of the CyberCash server (PK_{CS}) and signed by the buyer. It also contains a hash of the order to show agreement with the merchant without revealing what the order is and the forwarded merchant's initial signature.

4.6.3.3 Auth-Capture

The merchant forwards the encrypted card data and order to the gateway. The gateway verifies the cardholder signature on the card details, and the forwarded merchant's signature on the order details. The gateway verifies that both the customer and merchant agree on the order.

4.6.3.4 Charge-Action-Response

Having authorized and captured the purchase in the banking network, the gateway returns unsigned receipts for the merchant and buyer to the merchant.

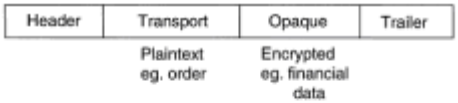
4.6.3.5 Charge-Card-Response (CH2)

The merchant forwards the unsigned receipt from the gateway to the cardholder.

4.6.4 CyberCash messages

The CyberCash messages are transport-independent so that they can be sent over the Web's HTTP, the SMTP protocol of electronic mail, or any other transport protocol. An example CyberCash message is shown in Figure 4.12. It consists of four parts:

- *Header:* This identifies the start of a CyberCash message.
- *Transport:* This is the plaintext part of the message. It can contain the order information in a purchase, transaction ID, date, and the identifier of the key used to encrypt the opaque part. The key identifier allows the receiver to know which key to select to decrypt the opaque part.
- *Opaque:* The encrypted part of a message. Typically, this will contain the encrypted financial data if it is present.
- *Trailer:* Used to indicate the end of the CyberCash message. It also contains a transmission checksum to allow the receiver to check that the entire message was received intact. This checksum is implemented as an MD5 hash of the first three message parts (header, transport, and opaque parts).




```
$$-CyberCash-1.0-$$
id: MIKE-50
transaction: 918273645
date: 199712250955
cyberkey:CC1001
opaque:
 GpOJvDpLH62z+eZlbVkhZJXtTneZH32Qj4T4IwJqv6kjAeMRZw6nR4f0OhvbTFfPm+GG
 aXmoxyUlwVnFkYcOyTbSOidqrwOjnAwLEVGJ/wa4ciKKl2PsNPA4sThpV2leFp2Vmkm4
 elmZdS0Qe350g6OPrkC7TKpqQKHjzczRRytWbFvE+zSi44wMF/ngzm/VsUCW01FXc8T
 9EB8KjHEzVSRfZDn+lP/c1nTLTwPrQ0DYiN1lGy9nwM1ImXtljHR19LZIHIRXy8=
$$-End-CyberCash-End-jkn38fD3+/DFDF3434mn10==-$$
```

Figure 4.12 CyberCash message structure.

4.6.5 Binding credit cards

A cardholder must register or bind the cards she wishes to use to her CyberCash persona. The card details are validated through the CyberCash gateway server before being stored securely in the wallet software. The messages used to do this are shown in Figure 4.13. By binding the credit cards to a persona, those card details can only be used in a purchase where the payment request has been signed by the private key corresponding to that persona. Also, in the binding process, the issuing bank will be asked whether this card may be used on the Internet. These precautions help reduce the risk of fraud.

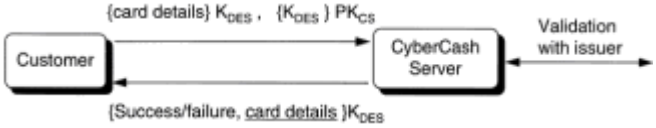


Figure 4.13 Binding a credit card.

4.6.6 Software updates

In the CyberCash system, software is informed if it is out of date automatically. Figure 4.14 shows the method of securely getting the location of new software. Such software migration is useful since CyberCash plans to update their payment protocol to be SET-compatible.

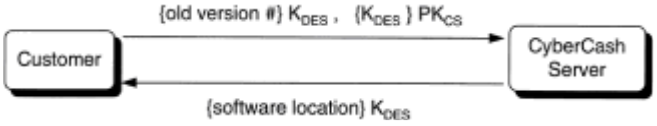


Figure 4.14 Automatic software updating.

4.7 *i*-Key protocol (*i*KP)

i KP [10] (where *i* = 1, 2, 3) is a family of secure payment protocols developed at IBM Research Labs in Zurich and Watson Research Center in the United States. The *i*KP protocols are based on public-key cryptography and differ from each other based on the number of parties that possess their own public-key pairs. This number is indicated by the name of the individual protocols: 1KP, 2KP, 3KP. The greater the number of parties that hold public-key pairs, the greater the level of security provided. The 1KP protocol is based on what security infrastructure already exists today. The 2KP and 3KP protocols can be phased in gradually to achieve full multiparty payment security as a more sophisticated certification infrastructure is put into place. This allows for gradual deployment of the system.

The current emphasis of the protocols is on credit card payments as this is envisaged to be the most popular form of payment in the near future due to a large user base in place already. The entities involved in the system are the customer, the merchant, the customer's bank, and the merchant's bank. In the context of credit card systems, the merchant's bank is known as the *acquirer* because it acquires paper charge slips from the merchants. The customer's bank is known as an *issuer* because it issues credit cards to users. In the *i*KP protocols, the acquirer acts as a gateway between the Internet and the existing financial networks that support transactions between banks. The *i*KP

protocols deal with *payment* transactions only (i.e., the solid lines in Figure 4.15). Therefore, the main parties involved in the transaction are the Customer (C), the Merchant (M), and the Acquirer gateway (A).

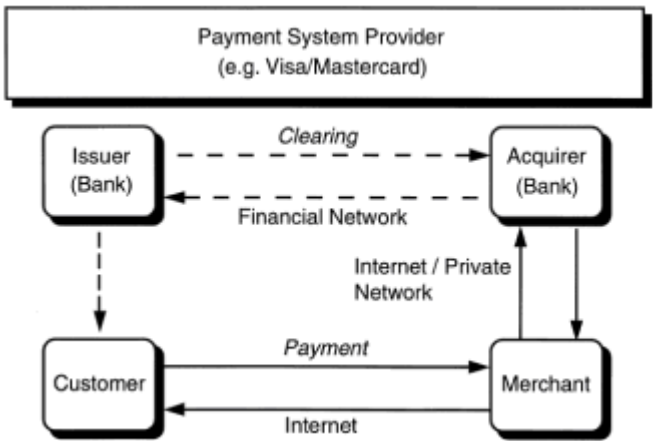


Figure 4.15 Payment protocol.

An important point to note is that the *i*KP suite of protocols are not a *shopping protocol* (i.e., the protocols do not provide encryption of the order information and assume that the order and price details have already been agreed upon by the customer and the merchant). Their sole function is to *enable* payment transactions between the various parties involved. This allows the protocols to be compatible with different browsing mechanisms.

4.7.1 Framework of *i*KP protocols

Each of the protocols consists of the seven basic steps shown in Figure 4.16. The contents of these steps may differ in each case:

1. *Initiate:* The customer initiates the protocol flow.
2. *Invoice:* The merchant responds by providing an invoice.
3. *Payment:* The customer generates a payment instruction and sends it to the merchant.
4. *Cancel:* The merchant can refuse to further process the transaction.
5. *Auth-Request:* The merchant sends an authorization request to the acquirer.
6. *A*uth-Response: The acquirer uses existing networked clearing systems to obtain the authorization and returns an authorization response.
7. *Confirm:* The merchant forwards the acquirers signed response and any other additional parameters to the customer.

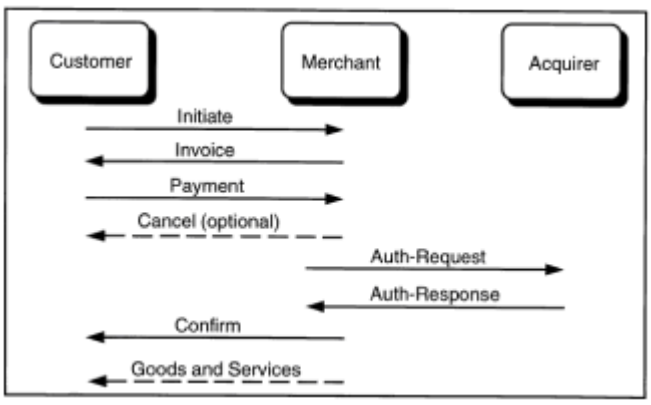


Figure 4.16 Overview of *i*KP protocol exchanges.

Table 4.2 lists a number of fundamental data elements that are exchanged in the course of an *i*KP transaction. Table 4.3 lists a number of fields that are formed by combining one

or more of the atomic fields listed in Table 4.2. The quantities surrounded by square brackets are either optional or may occur in individual *i*KP protocols.

Table 4.2 Quantities Occuring in the iKP Protocols

Item	Description
CAN	Customer's account number (e.g. credit card number)
IDM	Merchant ID. Identifies merchant to acquirer
TIDM	Transaction ID. Uniquely identifies the transaction
DESC	Description of the goods. Includes payment information such as credit card holder's name and bank identification number
SALTC	Random number generated by C. Used to randomize DESC and thus ensure privacy of DESC on the M to A link
NONCEM	Random number generated by a merchant to protect against replay
DATE	Merchant's current date/time
PIN	Customer's PIN which, if present can be optionally used in 1KP to enhance security
Y/N	Response from card Issuer. Yes/No or authorization code
RC	Random Number chosen by C to form CID
CID	A customer pseudo-ID which uniquely identifies C. Computed as CID 5 H(R_C, CAN).
V	Random number generated in 2KP and 3KP by merchant. Used to bind the Confirm and Invoice message flows.

Table 4.3 Composite Field

Item	Description
Common	Information held in common by all parties:PRICE, ID_M, TID_M, DATE, $NONCE_M$, CID, H(DESC, $SALT_C$), [H(V)]
Clear	Information transmitted in the clear:ID_M, TID_M, DATE, $NONCE_M$, H(Common), [H(V)]
SLIP	Payment Instructions:PRICE, H(Common), CAN, R_C, [PIN]
EncSlip	Payment instruction encrypted with the

	public key of the acquirer:PK_A (SLIP)
CERTX	Public key certificate of X, issued by a CA
SigA	Acquirer's signature:SK_A[H(Y/N, H(Common))]
SigM	Merchant's signature in Auth-Request:SK_M[H(H(Common), [H(V)])]
SigC	Cardholder's Signature:SK_C[H(EncSlip, H(Common))]

4.7.2 1KP

The 1KP protocol is the most basic of the protocols. Only the acquirer needs to possess and distribute its public-key certificate $CERT_A$. Public-key encryption is required from the customer only, while decryption is required from the acquirer only. Both customer and merchant are required to verify the signature generated by the acquirer. Each entity participating in the protocol is assumed to possess some starting information.

Table 4.4 lists the starting information for each of the entities in the system. It is assumed that the customer and merchant have agreed on the description of the goods (DESC) prior to initiating the protocol (i.e., DESC is not carried within any *i*KP flow). It is also assumed that both the cardholder and merchant are in possession of the public key of the CA and the acquirer's certificate from which they can extract its public key.

Table 4.4 Starting Information

Actor	Information Items
Customer	DESC, CAN, PKCA, [PIN], CERTA
Merchant	DESC, PKCA, CERTA
Acquirer	SKA, CERTA

All parties in 1KP must perform certain public-key computations. Each cardholder has a customer account number (CAN). This is normally the customer's credit card number and is kept secret from the merchant. The following sections describe each of the steps shown in Figure 4.17.

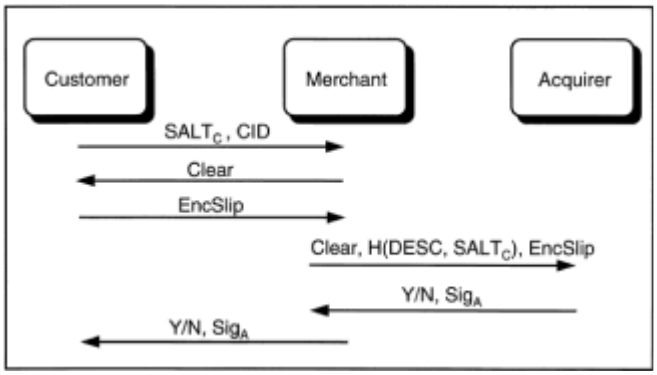


Figure 4.17 1KP protocol flows.

4.7.2.1 Initiate composition

1. The customer forms a one-time cardholder ID (CID) that uniquely identifies him. It is computed as the hash of the customer's account number (CAN), and a random value R_C. CID = $H(R_C, \text{CAN})$. (This avoids revealing the CAN to the merchant).

2. Generates another random number $SALT_C$ that will be used by the merchant to randomize the goods' description information to avoid revealing it to the acquirer.

3. The customer transfers the two quantities to the merchant as part of the Initiate message.

   Initiate: ($SALT_C$, CID)

4.7.2.2 Initiate processing and invoice composition

1. The merchant generates $NONCE_M$ and DATE. This allows the acquirer to uniquely identify an order. She also chooses a transaction ID (TID_M) to identify the context.

2. Computes H(DESC, $SALT_C$) and is now in a position to form Common. The merchant then computes H(Common) (see Table 4.3).

3. Sends Clear in the Invoice flow to the customer. (The fields in Clear are sent as cleartext.)

   Invoice: (ID_M, TID_M, DATE, $NONCE_M$, H (Common))

4.7.2.3 Invoice processing

1. The customer already has DESC and $SALT_C$ as part of his starting information and computes H(DESC, $SALT_C$). This quantity is included in Common by the merchant and is used by the customer to form Common in the next step.

2. Computes H(Common) and verifies that this matches the value of H(Common) in Clear generated by the merchant. This confirms that the customer and merchant agree upon the contents of Clear.

3. Generates a payment instruction (SLIP). He also includes the random number R_C, which was used to create the CID. This allows the acquirer to check that the CID given to the merchant corresponds to the CAN in the Payment message, but not the merchant to recover the customer's account number. He then encrypts the SLIP using the acquirer's public key.

4. Transfers the encrypted SLIP (EncSlip) to the merchant as part of the payment flow.

   Payment: (PK_A (SLIP))

4.7.2.4 Payment processing

1. If for some reason the merchant decides not to further process the message, she sends a Cancel message to the customer.

2. The merchant now performs an authorization. She forms an Auth-Request message. The merchant includes Clear and H(DESC, $SALT_C$) along with EncSlip, which she received as part of the payment instruction. This allows the acquirer to form Common and verify H(Common) generated by both the merchant and customer.

   Auth–Request: (EncSlip, Clear, H (DESC, $SALT_C$))

4.7.2.5 Auth-Request processing

1. The acquirer extracts from Clear the value H(Common) as computed by the merchant. This is referred to as h_1. It also checks for replays using the values ID_M, TID_M, DATE, and $NONCE_M$.

2. Decrypts EncSlip and extracts H(Common) from Slip as computed by the customer. This is referred to as h_2.

3. Checks that $h_1 = h_2$. This ensures that both customer and merchant agree on the order information.

4. Reforms Common from the various fields it receives in the Auth-Request and ensures that H(Common) = $h_1 = h_2$. In short, Common consists of a number of fields and H(Common) allows each of the participants in the protocol to verify that they all agree on the details (e.g., the price and description of the goods) of the transaction.

However, quantities such as the description information (DESC) and the cardholder's account number (CAN) are disguised/salted in such a manner that only the parties that need to know that information are given access to it.

5. The acquirer then contacts the card issuer and obtains clearance for the transaction.

Upon receipt of a response from the issuer, the acquirer computes a digital signature on the response (Y/N) and *H*(Common) and sends the Auth-Response to the merchant. (Note that both the merchant and customer are already in possession of *H*(Common) and thus it is not sent as part of the Auth-Response message flow.)

Auth–Response: (Y / N, Sig_A)

4.7.2.6 Auth-Response processing

1. The merchant verifies the signature (Sig_A).

2. The merchant forwards the response and the acquirer's signature to the customer as part of the Confirm message flow. The customer in turn verifies the acquirer's signature and the transaction is complete.

The 1KP protocol is simple and efficient for effecting electronic payments over open networks such as the Internet with minimal requirements for additional certification infrastructure. Its main weaknesses are the following:

- A customer authenticates himself to a merchant only using a credit card number and optional PIN as opposed to using digital signatures.

- The merchant does not authenticate herself to the customer or acquirer.

- Neither the merchant nor customer provide undeniable receipts for the transaction.

4.7.3 2KP

In 2KP, in addition to the acquirer, each merchant needs to possess a public-key pair and is required to distribute the public key contained in its certificate $CERT_M$ to both customer and acquirer. This enables the customer and acquirer to verify the authenticity of the merchant. We now describe the additions to the basic flows that are required in 2KP.

The starting information for each of the parties is as shown in Table 4.5.

There are now three new elements in the Invoice:

1. The merchant generates a random number V and creates a message digest $H(V)$. He adds $H(V)$ to Clear. (Note that Common now also contains $H(V)$). The inclusion of V in Confirm later acts as a receipt of proof to the customer that the merchant has accepted the authorization response.

2. The merchant uses his secret key (SK_M) to sign the pair *H*(Common) and $H(V)$ to produce Sig_M.

3. The merchant also includes his public key certificate $CERT_M$ so that the customer can verify the signature Sig_M.

Table 4.5 Starting Information

Actor	Information Items
Customer	DESC, CAN, PKCA, CERTA
Merchant	DESC, PKCA, CERTA, SKM, CERTM

The transaction message flow is shown in Figure 4.18. On receipt of the Invoice, the customer checks the merchant's signature Sig_M and generates a Payment message flow

as before. The merchant appends the same signature Sig_M that he sent to the customer as well as his public-key certificate $CERT_M$ to Auth-Request. The acquirer checks the merchant's signature before authorizing the transaction. Finally, the value of *V* is sent in Confirm to the customer, who in turn computes *H*(*V*) and verifies that this matches the value in Invoice.

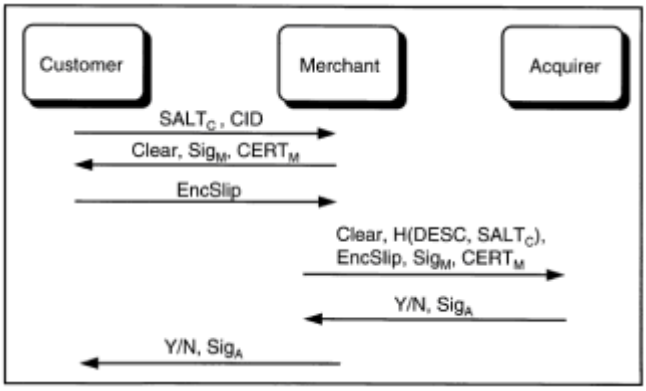


Figure 4.18 2KP protocol flows.

The 2KP protocol satisfies all the requirements of 1KP as well as:

- The customer and the acquirer can verify the authenticity of the merchant due to the inclusion of the merchant's signature Sig_M and $CERT_M$.
- The merchant generates a random number *V* and includes a message digest of the number *H*(*V*) in Common. He also signs the pair *H*(*V*) and *H*(Common) to form Sig_M . This signature and *H*(Common) are verified by both the customer and the acquirer. The acquirer also signs *H*(Common) in Sig_A. When the customer receives *V* as part of the Confirm message flow, he is able to verify that *H*(*V*) contained in Sig_M and Sig_A matches *H*(*V*) that he has computed. Also, no other party is capable of finding *V* as this would involve inverting a strong one-way function. In effect, the customer receives a receipt of the transaction that assures him that the merchant has received and accepted the payment.

4.7.4 3KP

In 3KP, all the parties possess public-key pairs and corresponding certificates. This allows for nonrepudiation of all protocol exchanges. The protocol is modified to have the customer send a certificate to a merchant, who forwards it onto the acquirer. We describe the additions that are required to the flows of 2KP.

As Figure 4.19 shows, the customer now sends his public-key certificate to the merchant as part of the Initiate message. As a part of the Payment, the customer sends his signature Sig_C to the merchant, where Sig_C is computed by encrypting EncSlip and *H*(Common). The merchant is able to verify the customer's signature as he already possesses the customer's certificate $CERT_C$. The merchant forwards Sig_C as part of the Auth-Request to the acquirer, who verifies the signature in turn.

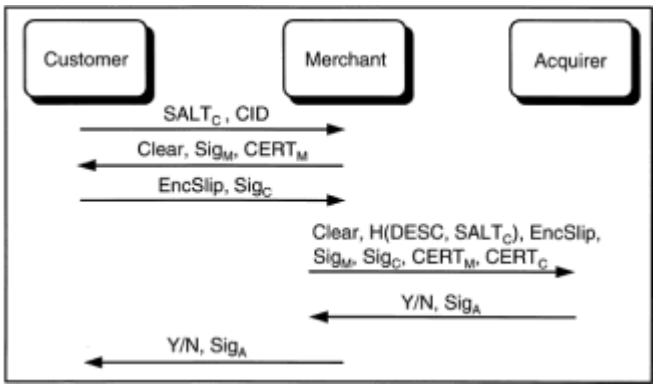


Figure 4.19 3KP protocol flows.

The starting information for each of the parties is as shown in Table 4.6. In 3KP, the customer's signature provides undeniable proof of the transaction authorization by the

customer. This can be verified by both the merchant and the acquirer.

Table 4.6 Starting Information

Actor	Information Items
Customer	DESC, CAN, PK_{CA}, SK_C, $CERT_C$
Merchant	DESC, PK_{CA}, $CERT_A$, SK_M, $CERT_M$
Acquirer	PK_{CA}, SK_A, $CERT_A$

The *i*KP protocols provide varying degrees of protection depending upon the number of entities that possess public-key pairs. It is envisaged by the developers that 1KP will be the protocol used most in the short term, with 2KP and 3KP being phased in over a period of time.

4.8 Secure electronic payment protocol (SEPP)

SEPP [11] is a protocol developed by MasterCard in October 1995 for secure payment processing using bank card transactions over public networks and is based on 3KP (see Section 4.7). As Figure 4.20 shows public-key cryptographic techniques are used to ensure that message content is not altered during the transmission between originator and recipient. (Note that as in the *i*KP protocols, privacy of nonfinancial data is not addressed by SEPP.) Merchants are able to verify that a cardholder is using a valid account number. SEPP also provides a mechanism to prevent fraudsters from posing as legitimate merchants and "collecting" card data. In addition to defining the electronic payment protocol, SEPP also defines a certificate management system to control certificates supporting security services.

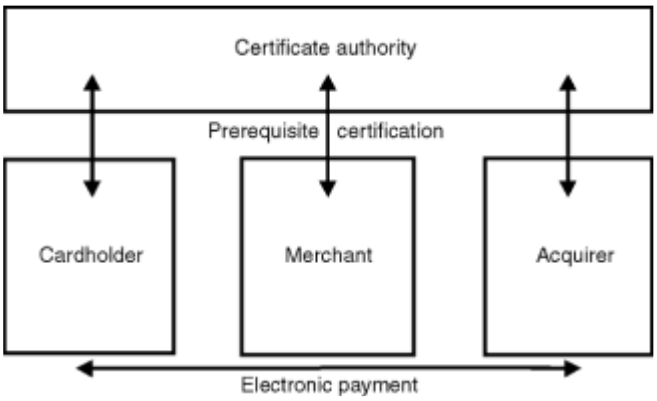


Figure 4.20 Overview of SEPP.

SEPP is the electronic equivalent of the paper charge slip, signature, and submission process. It assumes that the customer and merchant have already negotiated the price of the goods to be purchased out-of-band (e.g., through the WWW). SEPP takes input from the negotiation process (payment amount, order description, payment method, etc.) and causes payment to happen via a three-way communication among the cardholder, the merchant, and the acquirer.

4.8.1 System architecture

The SEPP system is composed of a number of entities depicted in Figure 4.21, namely:

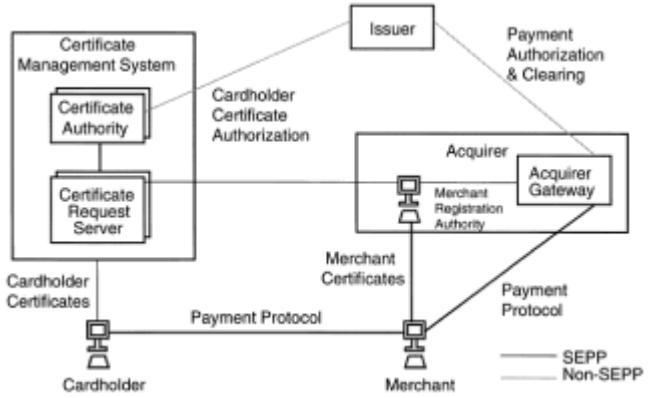


Figure 4.21 System overview.

- *Cardholder:* An authorized holder of a bank card issued by an issuer who is registered to perform electronic commerce.
- *Merchant:* A merchant of goods or services and who accepts payment for them electronically.
- *Acquirer:* A registered financial institution that supports merchants by providing service for processing credit card transactions. The acquirer consists of an acquirer gateway and merchant registration authority. The acquirer gateway interfaces to the merchant system to support authorization and capture services to merchants. The merchant registration authority (MRA) enables the acquirer to securely receive, validate, and forward merchant certificate requests to the certificate management system and receive back certificates.
- *Certificate management system (CMS):* An agent of one or more bank card associations that provides for the creation and distribution of electronic certificates for merchants, acquirers, and cardholders. The CMS consists of one or more certificate authorities to provide a trusted, reliable certificate granting service to cardholders, merchants, and acquirers. It also contains certificate request servers that issue certificates to cardholders through the WWW and interfaces to the acquirer's MRA to provide merchant certificates. X.509 version 3 public-key certificates are specified.

4.8.2 Certificate management system

The CMS is implemented as a hierarchy of servers, each with its own cryptographic support. A certificate is used to bind the public key of a user to some uniquely identifying information (e.g., a customer's name or account number).

- A cardholder certificate binds the public key of a certificate holder to a specific account number and assures the merchant that a legitimate account number is provided in the transaction. In order to protect the cardholder's privacy a hashed (effectively encrypted) version of the account number is specified in the cardholder's public-key certificate. The cardholder passes her account number and a secret variable called a certificate code to an acquirer so that the acquirer can verify the hashed value contained within the cardholder's certificate.
- A merchant certificate assures the cardholder that the merchant is a legitimate MasterCard merchant (i.e., has a relationship with a MasterCard acquirer).
- The public key within an acquirer's certificate is used by the cardholder to encrypt the payment instruction, which includes the account number.

4.8.3 SEPP keys used

Several keys are used in SEPP, including the following:

- *Cardholder keys:* A cardholder requires a key pair for creating digital signatures during a payment transaction. The secret key is stored locally on disk in encrypted form using a password known only to the cardholder. The cardholder's public and secret key pairs are generated locally by a cryptographic device or software. In order to provide the maximum protection against compromise, the secret key is stored securely in tamper-proof hardware or protected with a password if it is stored on disk.

- *Merchant keys:* A merchant may either have one or two pairs of keys. One pair is required for creating digital signatures. The other pair is optional and is used for encryption of payment data sent from the acquirer back to the merchant, if the acquirer provides this option.

- *Acquirer keys:* Three key pairs are required by the acquirer gateway. The first is used to digitally sign receipts provided to the cardholder and merchant. The second is used for encrypting and decrypting payment data received from the cardholder. The third is used for signing renewal requests for certificates to be sent to the CMS.

The public components of each of the above key pairs are certified by the CMS.

4.8.4 Overview of payment process

SEPP assumes that the cardholder and merchant have been communicating in order to negotiate the terms of a purchase and generate an order. SEPP can be used in both interactive (e.g., WWW) and noninteractive modes (e.g., CD-ROM catalogue). In a noninteractive environment, the transaction details can be sent to the merchant via electronic mail.

SEPP defines six basic flows:

1. Purchase order with inline authorization;
2. Purchase order with delayed authorization;
3. Offline purchase order;
4. Capture;
5. Purchase order inquiry;
6. Error.

The following sections summarize the most important of these message flows.

4.8.5 Purchase order with inline authorization

This is the basic purchase order flow where the merchant performs an “online” authorization check. It is shown in Figure 4.22.

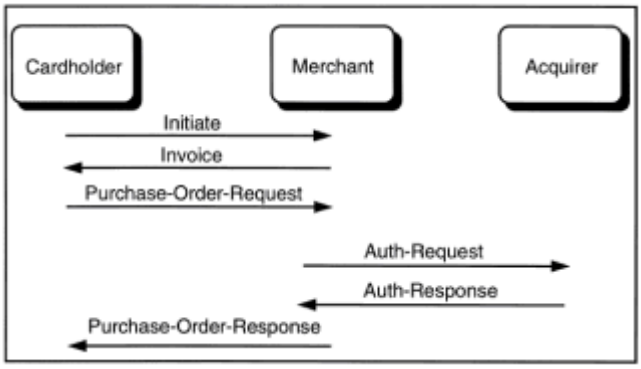


Figure 4.22 Purchase order with inline authorization.

1. In the interactive message environment, a SEPP transaction starts when the cardholder sends an Initiate message to the merchant. This message is used to request the merchant to prepare an invoice as the first step in the payment process.
2. The merchant responds with an Invoice message, which contains the amount of the transaction, merchant identification information, and data used to validate subsequent transactions in the sequence.
3. The cardholder responds by sending a PO-Request. It contains the payment instruction (PI) of the cardholder. The PI is encrypted such that it can be only read by the acquirer. This is necessary since the PI contains the cardholder’s primary account number (PAN), which should not be revealed to the merchant.

4. The merchant then sends an Auth-Request to the acquirer.
5. The acquirer performs the following:
   - Authenticates the merchant;
   - Decrypts the PI from the cardholder;
   - Validates that the cardholder's certificate matches the account number used in the purchase;
   - Validates the consistency between the merchant's authorization request and the cardholder's payment data;
   - Formats a standard authorization request to the issuer and receives a response;
   - Responds to the merchant with a validated Auth-Response.
6. The merchant responds with a PO-Response indicating that either the merchant has received the purchase order request and the authorization request will be processed later, or the authorization response has been processed by the acquirer.

The cardholder can then (or at a later date) request a status of the purchase order by using a PO-Inquiry message. The merchant responds with a PO-Inquiry-Response message.

4.8.6 Purchase order with delayed authorization

This variation on the basic purchase order flow is shown in Figure 4.23. Here, the merchant chooses to delay authorization until after sending a response to the cardholder. The final message to the customer is a PO-Inquiry-Response rather than a PO-Response because the merchant has not yet completed the authorization. The Auth-Request and Auth-Response portion of flow is the same as in the previous case.

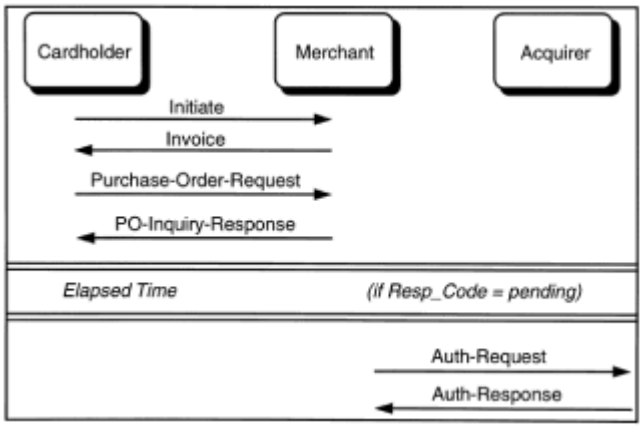


Figure 4.23 Purchase order with delayed authorization flow.

4.8.7 Offline purchase order

In the case of electronic mail transactions, the first message from the cardholder is a PO-Request. As Figure 4.24 shows, the PO-Response from the merchant is sent back to the cardholder via e-mail. The Initiate and Invoice messages are omitted.

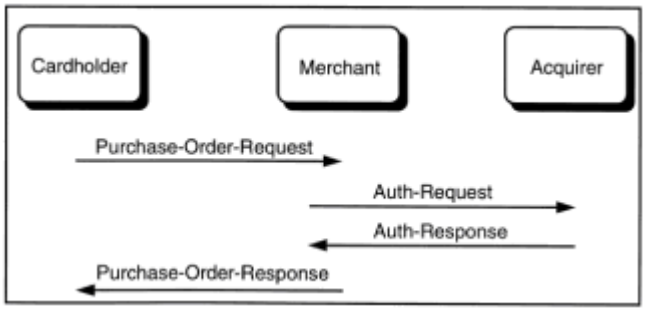


Figure 4.24 Offline purchase order flow.

4.8.8 Capture

A merchant uses the capture flow to request the credit card clearing system to transfer funds from the cardholder to the merchant. A capture flow may only occur after the authorization flow has been completed and if the purchase order was not captured as part of the authorization flow. The merchant must send the Capture-Request to the same acquirer to whom the Auth-Request message was sent. Figure 4.25 shows this dialogue.

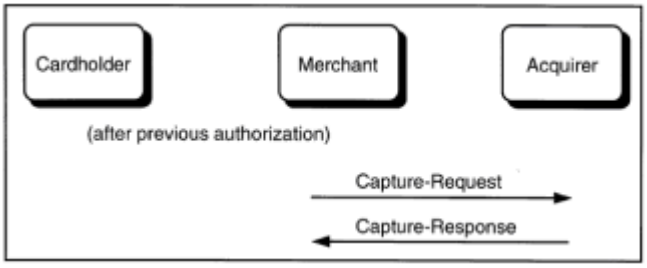


Figure 4.25 Capture flow.

SEPP supports three distinct modes of capture:

1. *Online capture combined with authorization*. The merchant may request that payment be captured by the acquirer at the time authorization is obtained.

2. *Deferred online capture*. The merchant may request payment capture separately from and subsequent to authorization.

3. *Offline capture*. The merchant may obtain payment through offline mechanisms that do not involve SEPP.

4.8.9 Security requirements

SEPP addresses a range of requirements for each party directly involved in the SEPP payment process.

4.8.9.1 Cardholder

- Unauthorized payments are impossible and cannot be charged to a cardholder's account without possession of the cardholder's account number and secret key.
- A cardholder can verify the authenticity/trustworthiness of the merchant.
- A cardholder receives undeniable proof of transaction authorization from the acquirer.
- A cardholder receives a receipt from the merchant that gives undeniable proof that the merchant received the payment authorization.

4.8.9.2 Merchant

- A merchant receives undeniable proof that the acquirer has authorized the payment transaction.
- A merchant also obtains undeniable proof that the cardholder has authorized the transaction.

4.8.9.3 Acquirer

- The acquirer obtains undeniable proof of transaction authorization by the cardholder prior to debiting the account.
- The acquirer obtains undeniable proof of transaction authorization by the merchant. It is impossible to authorize a payment transaction without the merchant's authorization.

Further work on SEPP has been abandoned in favor of the SET specifications as discussed in the next section.

.9 Secure Electronic Transactions (SET)

The launch by the alliance comprising of MasterCard, Netscape Corporation, IBM, and others of the Secure Electronic Payment Protocol (SEPP) in October 1995 came within a few days of the launch by a VISA and Microsoft consortium of a different network payment specification called Secure Transaction Technology (STT). This led to an unfortunate situation where the two major credit card associations were each backing an independent solution for network payments.

For a number of months the two efforts proceeded in parallel, each developing separate reference implementations. Efforts were made to develop payment-method selection protocols that could accommodate both systems. Ultimately, good sense prevailed, and in January 1996, the companies announced that they would come together to develop a unified system that would be called Secure Electronic Transactions (SET).

Late in February of that year, two documents were issued, the first of which [12] gave a business overview of the protocols, and the second [13] of which gave more technical details. This was followed by a public comment period during which interested parties discussed the specifications, and identified flaws. Following this, a revised book 3 protocol description was released [14] that defines the production SET protocol.

Most significant organizations in the Internet payment industry have stated that they will support SET, and indeed, very shortly after the specifications were released, American Express—another payment card company with a global presence—stated that they would also adopt the protocol. Note, however, that the SET specifications at the time of writing are an interim standard, parts of which may change as the standardization work progresses.

The scope of the SET protocols was quite restricted from the outset. Firstly, it was intended only as a payment protocol. The specification documents make clear that protocols would be developed by other parties to address online shopping, price negotiation, payment-method selection, and other electronic commerce functions. SET would only come into play after the customer had decided what to buy, for how much, and that she wanted to pay with a payment card. Although not part of the standard, an appendix to the SET technical specification [13] does outline a method for “Interim Merchant/Cardholder Communication” that is likely to be influential in the absence of well-developed alternatives.

In a conventional MOTO credit card transaction, a cardholder forwards her details to the merchant who will then contact his (the merchant’s) acquirer to obtain clearance for the payment. The acquirer can obtain this authorization from the institution that issued the card via a financial network operated by the card association (e.g., MasterCard or VISA). These private networks have existed for some time and have their own set of proprietary protocols operating on dedicated links with appropriate security mechanisms in operation. Thus an infrastructure of links and transaction-processing computer hardware exists to electronically authorize credit card payments. SET assumes the existence of such a facility and only specifies the subset of dialogues between the customer and merchant and between the merchant and an entity known as the payment gateway.

An overview of the payment process is shown in Figure 4.26. The cardholder initiates a payment with the merchant using SET. The merchant then uses SET to have the payment authorized. The entity involved is called the payment gateway and it may be operated by an acquirer or could be some shared facility operated by a group of acquirers (or, indeed, the card association). The payment gateway acts as the front end to the existing financial network, and through this the card issuer can be contacted to explicitly authorize each and every transaction that takes place.

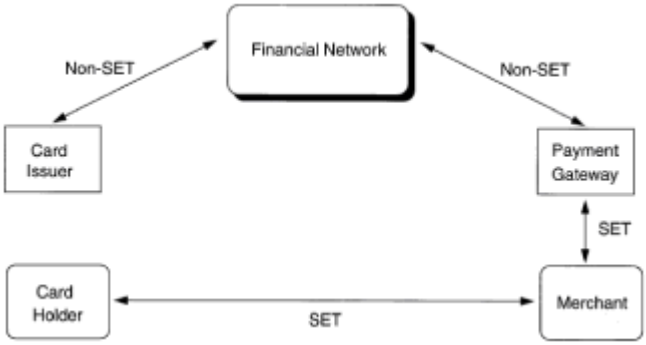


Figure 4.26 Phases of a credit card payment addressed by SET standards.

It is clear from the diagrams that SET is not intended to be a general-purpose payment protocol and is restricted to payment card or similar applications where parties will take on the role of buyer, merchant, or acquirer. It does not address transfer of funds from one individual to another and relies on the existing credit card infrastructure to effect the payment. The cardholder will see SET transactions on her credit card statement side by side with more conventional credit card payments, and the acquirer will see this as an extension of the current relationship he has with his merchant customers.

4.9.1 The SET trust model

Each of the parties that participates in a SET payment, with the possible exception of the cardholder, will be required to authenticate themselves at some point in the payment process. They will do this by using the private part of a public-key pair. For this to operate satisfactorily, the corresponding public keys must be certified by some trusted third party.

In SET, bindings between identities and their corresponding public keys are stored using certificates in X.509 version 3 format.

Figure 4.27 shows the format of such a certificate expressed in abstract syntax notation 1 (ASN.1). The principle purpose of this certificate is to bind the identity given by the *subject* field to the public key held in *subjectPublicKeyInfo* where the binding is certified by *issuer*: the authority that applies the signature to the certificate. In SET, a hierarchy of certification authorities shown in Figure 4.28 will be used.

```
Certificate ::= SIGNED { SEQUENCE {
                    version [0]Version DEFAULT v1,
                    serialNumber  CertificateSerialNumber,
                    signature  AlgorithmIdentifier,
                    issuer  Name,
                    validity  Validity,
                    subject  Name,
                    subjectPublicKeyInfo SubjectPublicKeyInfo,
                    issuerUniqueID  [1] IMPLICIT UniqueIdentifier OPTIONAL,
                    subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
                    extensions  [3] Extensions OPTIONAL  }
```

Figure 4.27 The certificate definition in X.509 version 3.

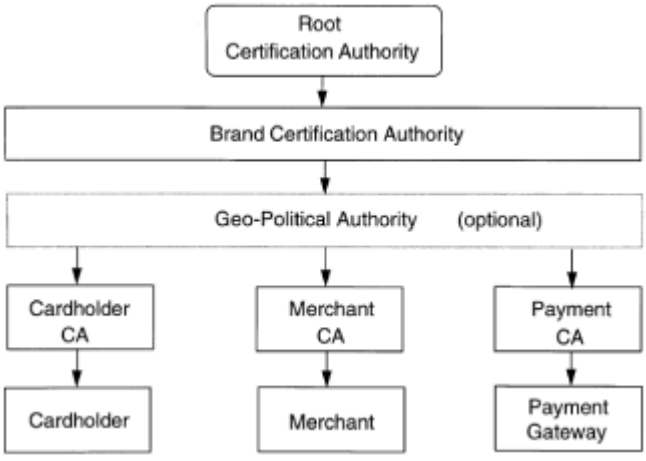


Figure 4.28 Certification authority hierarchy used in SET.

There will be a single root key for all entities using SET. This will sign the certificates of each of the brands (e.g., VISA, MasterCard, American Express, etc.) using a 2,048-bit key. Procedures are defined to change from one root key to another before the root certificate reaches the end of its validity period. Each of the brands will then either deal directly with member financial institutions in individual countries or they will delegate this to some kind of geopolitical organization (e.g., VISA Europe). The certificates issued by the brand CAs and those lower down in the hierarchy are all signed using a 1,024-bit key length, reflecting the less onerous security regime required at the lower levels.

If cardholder certificates are being issued, this function will be carried out by the card issuer. In the period leading up to the launch of SET, MasterCard had resolved not to do this, while VISA's policy was to require all cardholders to produce certificates when they engaged in a SET transaction. Payment gateways may be operated by an acquirer, who would already act as a merchant CA, but nevertheless, these gateways must be able to produce a certificate that has been signed by the payment CA before they can participate in a transaction.

SET also makes use of the *Extensions* field that became available in version 3 of the X.509 certificate definition. In particular, each certificate has a *Key Usage Restriction* field that specifies what tasks (e.g., signature verification, data encipherment, etc.) the key is approved for by the issuing CA. Cardholders are normally issued with a signature generation key certificate and they should not use the associated key pair for message confidentiality purposes.

The entities (cardholder, merchant, etc.) involved in SET transactions are identified in the certificates using an X.500 distinguished name. This is a collection of attributes that, taken together, uniquely identify the entity. Some examples of these attributes include country (c) and organization (o). Subdivisions of an organization can be identified by including one or more organizational units (ou) in the distinguished name. Normally, the last attribute used is the common name (cn), which would ordinarily give the individual name of an entity. In the case of SET cardholder certificates, neither the person's name nor the credit card number (often referred to as the primary account number or PAN) is shown in the certificate. Instead, a numeric quantity, computed by concatenating the PAN with a nonce as well as some fixed-character strings and computing the resulting hash, is used. This value is never regenerated, but is stored by the card issuer, and when a payment is attempted, the stored value is compared with the blinded account number contained in the request for payment authorization.

Figure 4.29 shows some examples of the kind of names that may be assigned to SET entities. The overall root of the hierarchy will be an organization that is at arms length from any of the card associations operating SET payment systems. The attributes specified in its name will be the country, the name of the organization, and a unique identifying string. Other examples shown in the figure are for a cardholder who has been issued with a "European Express Card" from the Bank of Ireland, where the local promotional name used is "Fun Card." The common name attribute is a blinded version of the card number. This cardholder is shown interacting with a merchant called "Wolf a Pizza," whose acquiring organization is the Wells Fargo Bank. The payment gateway is operated by the same financial institution.

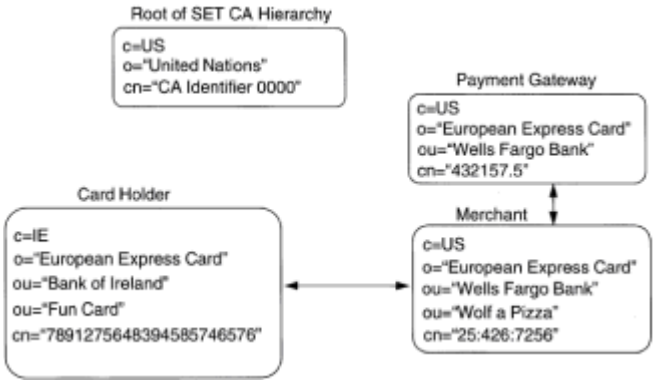


Figure 4.29 Some typical names assigned to SET entities.

4.9.2 SET message structure

The SET protocol consists of request/response message pairs, such as the PReq and

PRes messages shown in Figure 4.30. In this section, the contents and flow of the messages required to complete a purchase transaction are presented. To allow interoperability, the messages are defined in a machine-independent format in the specification. This will allow clients produced by one software company to perform a SET transaction with a server developed by a completely different company.

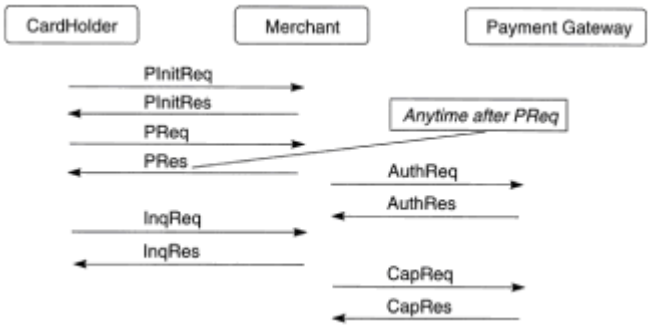


Figure 4.30 Steps in a SET transaction.

Encryption is performed on parts of certain messages. This end-to-end solution allows information contained with the message to be selectively revealed to parties as required. For example, the financial data about a credit card is not revealed to the merchant, and data about the purchased product is concealed from the acquirer. Point-to-point encryption of a connection link would not allow such selection to occur.

The messages needed to perform a complete purchase transaction (Figure 4.30) usually include

- Initialization (PInitReq/PInitRes);
- Purchase order (PReq/PRes);
- Authorization (AuthReq/AuthRes);
- Capture of payment (CapReq/CapRes);
- Cardholder inquiry (InqReq/InqRes) [optional].

These messages are not necessarily in the above order. The contents and function of each pair is now examined in turn.

4.9.3 Payment initialization (PInitReq/PInitRes)

The SET payment starts after the cardholder has been presented with a completed order form and approved its contents. The way in which goods are selected and presented in an order form is outside the scope of SET, but it is likely that shopping protocols will be designed to handle this.

The PInitReq message is sent to the merchant to indicate that the cardholder is ready to pay for the goods. As shown in Figure 4.31, it contains

- Brand of card that will be used in the payment, such as VISA or MasterCard (Brand_ID);
- A local ID for the transaction (LID_C);
- Optional list of certificates (Thumbs) already stored by the cardholder software. This list consists of a thumbprint (SHA hash) of each certificate held;
- Challenge variable (Chall_C), which will be used in the merchant's response, to guarantee the freshness of the communication.

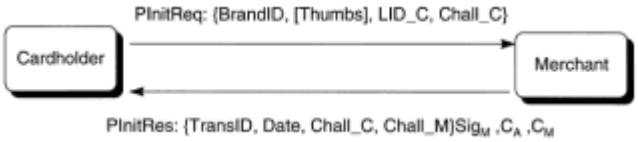


Figure 4.31 Initialization messages.

Upon receipt of the PInitReq, the merchant generates a globally unique ID that is combined with the LID_C to form the complete transaction ID (TransID). This is used to identify a specific purchase from other purchase messages received.

The merchant's response contains this TransID along with certificates and the current date. The cardholder's challenge is included along with a new merchant challenge (Chall_M). The certificates include those keys that are needed in the payment and that the cardholder doesn't already hold, such as the merchant and acquirer public keys.

Since the cardholder has now received an appropriate response to his challenge, he can be confident that the merchant is an accredited retailer.

4.9.4 Purchase order (PReq/PRes)

The purchase order messages fulfill the actual purchase by the cardholder from the merchant. It is the most complex message pair in the payment protocol. The cardholder sends two elements, the order information (OI) and payment instructions (PI), to the merchant, as shown in Figure 4.32:

Figure 4.32 Purchase request.

- The OI holds data that identifies the order description at the merchant.
- The PI contains the actual card data, purchase amount, and order and transaction identifiers. The PI is encrypted with the acquirer's public key so that the merchant cannot view its contents. It is forwarded to the acquirer later as part of the authorization.

4.9.4.1 Order information (OI)

The construction of the OI is shown in Figure 4.33. OIData consists of data from the initialization phase. The merchant challenge, Chall_M, is returned, demonstrating the freshness of the message to the merchant. ODsalt is a nonce that is used when creating a hash of the order description. By including this random nonce within the hash, dictionary attacks are prevented. That is, the nonce stops an attacker from guessing the hash value, *H*(OIData), by trying all possible combinations of dictionary words in the order description.

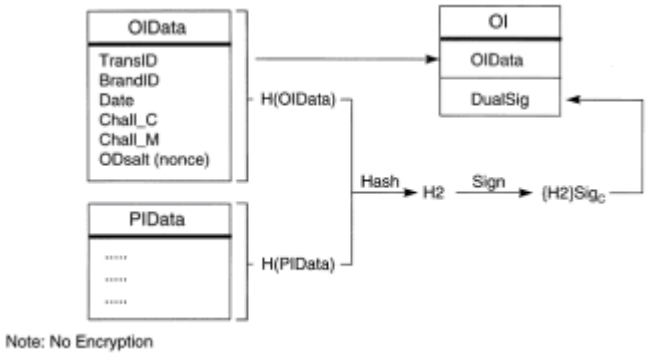


Figure 4.33 Construction of the order info (OI) element.

A dual signature is created using the hashes of OIData and PIData (the data contents of the PI). The signature links OIData and PIData, which links the cardholder's order to the authorization payment instructions by showing that they were signed together. Anyone possessing either OIData or PIData and the dual signature can verify the signature without having to know the other. The dual signature is also a speed optimization since only one signature is needed instead of having to separately sign OIData and PIData. The cardholder signature certificate is also included with the signature so that the

merchant can verify it.

4.9.4.2 SET dual signatures

While the dual signatures in SET are created in the normal way, as described in Chapter 3, a further optimization is applied. The result of XORing the hash of message 1 with the hash of message 2 is included with the signature. The signature can be verified with this value and either message 1 or message 2. This prevents having to include *H*(M1) in one copy of the signature, and *H*(M2) in the second copy of the signature when distributing to the two parties involved, as is normally the case. Instead, these values can be extracted from the XOR value and the single version of the dual signature verified. The following example clarifies this:

- Take two messages, M1 and M2.
- Create a dual signature in the normal way:
  *H*2 = *H*(*H*(M1), *H*(M2)). Then Sign H2.
- Normal dual signature:
  {H2}, Sig_X , *H*(M1) for holder of M2.
  {H2}, Sig_X , *H*(M2) for holder of M1.
- But now include *Y* = *H*(M1) XOR *H*(M2) in signature.
- SET dual signature: {*H*2}Sig_X, *Y* for all parties.
- The holder of M1 verifies the dual signature by extracting *H*(M2) from *Y*:
  H(M1) is first obtained by applying the hash algorithm to M1)
  H(M2) = *H*(M1) XOR *Y*
- Having obtained *H*(M2) from *Y*, the dual signature is validated in the normal way:
  H(M1) and *H*(M2) are concatenated, and the result then hashed.
  *H*3 = *H*(*H*(M1), *H*(M2)). The resulting hash, *H*3, should match
  *H*2 in the signature.
  *H*3 = *H*2 if signature valid.

The SET dual signature optimization allows the same signature material to be sent to both parties ({*H*2}Sig_X, *Y*), instead of having to replace *Y* with different values for each receiver. From the SET dual signature message, both parties can verify the signature that links message 1 and message 2 together.

4.9.4.3 Payment instructions (PI)

The PI is constructed as shown in Figure 4.34. Remember, the contents of the PI are never seen by the merchant, but are forwarded to the acquirer. The actual credit card data is included with nonces to defeat playback and dictionary attacks. The card data is protected by using extra-strong encryption as described in the following section. A hash of the order, *H*(Order), is included, which identifies the unique order the cardholder is referring to without giving away what that order is.

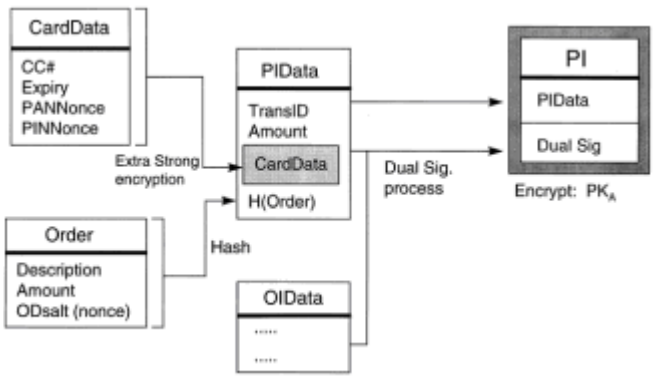


Figure 4.34 Construction of payment instructions (PI).

The same dual signature already created for OI is used as the signature on the PI. The PI is encrypted with the public-key exchange key of the acquirer. This prevents the merchant, or anyone else, from viewing its contents.

The payment request embodies the actual credit card payment from the cardholder's point of view. It is at the core of the SET payment protocol and once sent, the cardholder has shown an agreement to pay that cannot be easily reversed.

4.9.4.4 SET extra-strong encryption

Instead of encrypting with a symmetric DES key and then encrypting this symmetric key using RSA, the data is directly encrypted with RSA. This is illustrated in Figure 4.35 where extra strong encryption is used on the card data and is much stronger than the normal SET encryption.

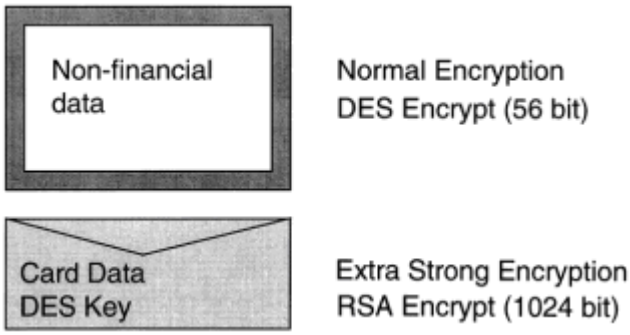


Figure 4.35 SET extra-strong encryption.

4.9.4.5 Processing PReq

When the merchant receives a purchase request from a cardholder, the OI and PI parts are extracted. The merchant verifies the cardholder's dual signature on OI using the cardholder's certificate by traversing the certificate trust chain to the root.

Before sending a purchase response (PRes) to the cardholder, the merchant will normally perform the authorization and perhaps capture steps of the payment. Authorization and capture are described in the following sections. However, PRes may be returned before capture or before both capture and authorization. The option chosen will affect the contents of the messages.

If authorization is delayed, the merchant will return a purchase response indicating that the cardholder should inquire later about transaction status. The cardholder inquiry messages are described in Section 4.9.7.

When the merchant does send the purchase response to the cardholder, it will contain the transaction status and any result codes available. The message format is shown in Figure 4.36. The CompletionCode indicates whether the authorization or capture steps have been completed. Results contains the authorization or capture codes for the transaction if these steps have been performed. These codes were generated in the financial bankcard network to authorize and clear the transaction and may appear on the cardholder's monthly bill.

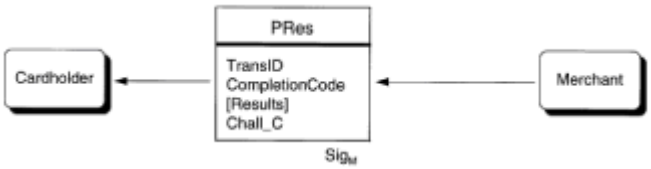


Figure 4.36 Merchant's purchase response message.

The purchase order messages form the actual purchase by the cardholder from the merchant. When the cardholder receives the merchant's response she knows that either the payment is complete or that the transaction is waiting to be processed by the credit card financial network.

4.9.5 Authorization (AuthReq/AuthRes)

This process allows the merchant to verify that the cardholder has credit for the purchase and to obtain permission to charge the transaction to her credit card. In the authorization request, the merchant sends data about the purchase, signed and encrypted to the

acquirer. The PI from the cardholder is also forwarded in this request.

The data sent includes a hash of the order details as shown in Figure 4.37. If this matches the *H*(Order) present in the PI, the acquirer knows that the merchant and cardholder are in agreement about the ordered goods and purchase amount. The dual signature in the PI proves that it came from the cardholder. *H*(OIData) in the merchant request shows knowledge of OIData that is signed by the dual signature, showing agreement on the order data without revealing it. Thumbs is an optional list of relevant certificates held by the merchant to prevent the acquirer sending them in the response. The cardholder billing address (obtained outside the SET protocol) and other merchant details such as business type are also expected.

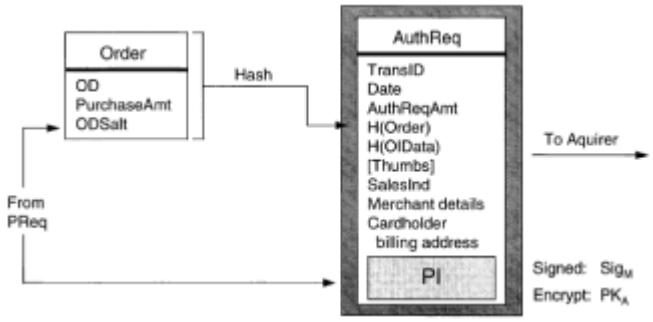


Figure 4.37 Requesting authorization of the purchase.

Both authorization and capture can be performed in a single request, known as a sales transaction. SalesInd is used to indicate if the merchant wishes to do this.

Upon receiving AuthReq, the acquirer decrypts the message parts, verifies signatures, and checks for consistency between the purchase details sent by the merchant and those in the PI. If the merchant's request (AuthReqAmt) is not the same as the PurchaseAmt, it is checked that this difference is acceptable to policy. Next, the acquirer obtains the authorization through the existing financial network as shown in Figure 4.38.

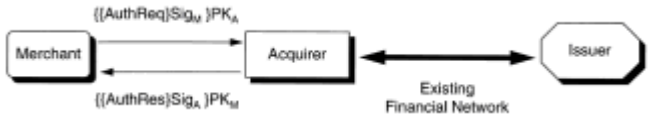


Figure 4.38 Obtaining authorization for a purchase transaction.

Having received a good authorization from the card issuer, an authorization response (AuthRes) is returned to the merchant with the authorization code from the issuer. Illustrated in Figure 4.39, it also contains a capture token, signed and encrypted, which is later used by the merchant to capture the payment. Only the acquirer can decrypt the token, keeping the capture data hidden from anyone else.

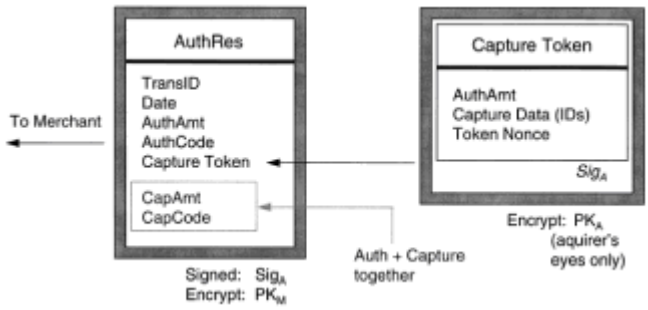


Figure 4.39 Authorization response with capture token.

If capture was performed with the authorization (sales transaction), then the capture code and amount are returned instead of the capture token. Upon receiving a good authorization, a merchant can ship the goods purchased. A good authorization indicates that the card issuer has verified the card details and credit limit, and given the go-ahead for the purchase.

4.9.6 Capture of payment (CapReq/CapRes)

After processing an order, the merchant needs to request the payment previously authorized to be transferred to his account. The total payment for several authorizations can be captured in a single batch request as shown in Figure 4.40. A merchant might accumulate many capture tokens (from authorization) throughout the day and then request reimbursement for these at the end of the day.

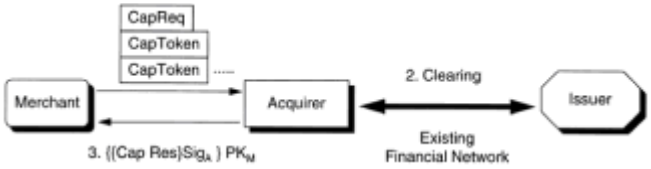


Figure 4.40 Capturing multiple authorized payments in a single request.

The contents of a capture request (CapReq) are shown in Figure 4.41. Many capture tokens from different transactions can be included in a single request for efficiency. For each token, the corresponding authorized amount and transaction ID are included. These should match the data encrypted within the capture token itself. CapID is just a unique value used to identify this capture from others. The request is signed and encrypted by the merchant.

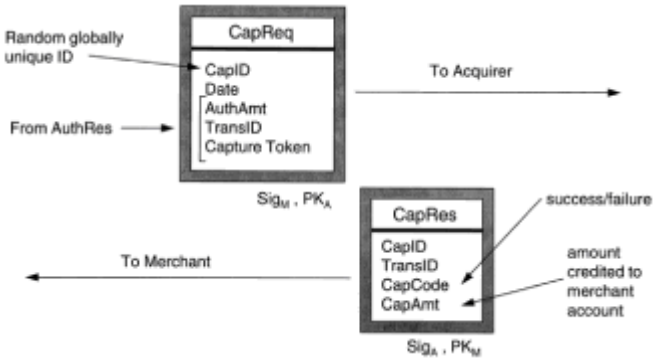


Figure 4.41 Capture request/response messages.

After verifying the capture request, the acquirer credits the merchant's account. Transaction fees may be deducted at this stage. The capture response (CapRes), signed and encrypted, contains a success indication, the settled amount, and a capture code from the financial network.

After a successful capture, the merchant has received the actual payment of money from the cardholder's purchase. If the merchant has not already sent the purchase response to the cardholder, this is now done.

4.9.7 Cardholder inquiry (InqReq/InqRes)

The inquiry messages allow a cardholder to check the status of a transaction. An inquiry can be sent any time after the purchase request, and cardholders can only inquire about their own purchases. The inquiry can be performed several times for a single transaction.

The message contents are shown in Figure 4.42. The inquiry request contains the transaction identifier and includes a new challenge variable. This should be unique to each invocation, since the inquiry may be sent repeatedly. The inquiry is signed to prove that the request comes from the correct cardholder.

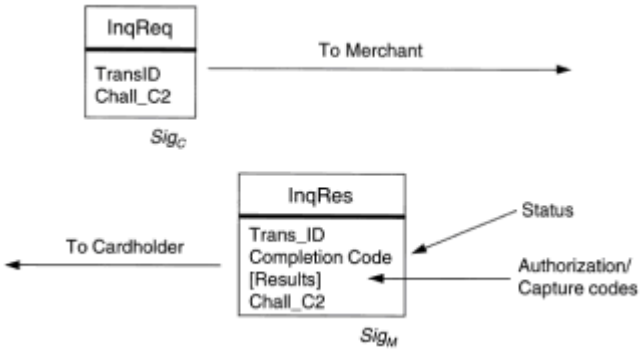


Figure 4.42 Inquiry request/response messages.

The inquiry response returned by the merchant is very similar to PRes. It contains the transaction status and any result codes (authorization or capture) available. Having received an inquiry response, the cardholder can be sure as to how a specific purchase with an accredited merchant is proceeding.

4.9.7.1 Cardholder registration

All parties involved in a payment need key certificates to send SET messages. When cardholders want to start using SET to make network payments, they must obtain a public key certificate from a certification authority (CA).

Every cardholder needs a signature certificate containing their public signature key so that messages signed by them can be verified. An encryption certificate, with a public-key exchange key, is optional since none of the SET payment messages sent to the cardholder are encrypted in the current version of SET.

The method for obtaining a CA signed certificate is outlined in Figure 4.43. Whether the cardholder is obtaining one certificate or multiple certificates, the procedure is much the same.

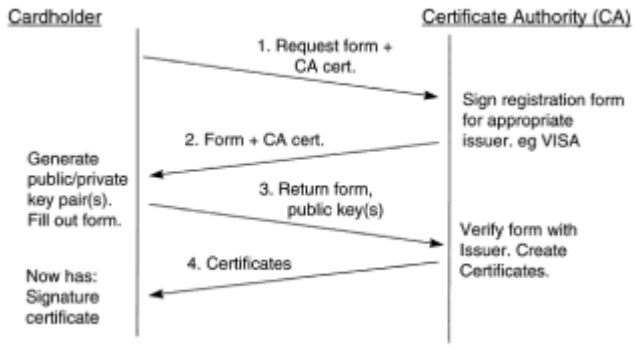


Figure 4.43 Cardholder registration.

Certificates have an expiry date, which usually matches the credit card expiry date. The procedure for renewing certificates is also very similar to that shown. Similar certificate registration and renewal processes exist for merchants and acquirers.

The registration process starts when the cardholder sends a request (CInitReq) to the CA asking for:

- CA's key exchange certificate. This will allow later messages sent to the CA to be encrypted with its public-key exchange key. Clearly, to verify this certificate, all users must have a copy of the root public key.
- An electronic registration form from the cardholder's financial institution (issuer).

The contents of CInitReq are shown in Figure 4.44. The request type indicates whether the cardholder seeks a signature certificate, an encryption certificate, or both. The BankID (BIN) is the first six numbers on the card and uniquely identifies the issuing bank. The Language field is used to request the language on the returned registration form. A thumbprint (Thumbs) of current certificates held is also sent.

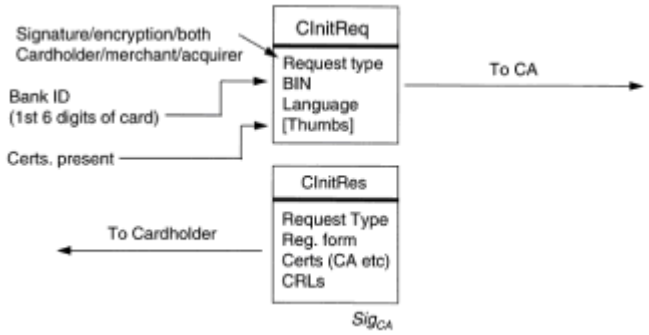


Figure 4.44 CInit request/response messages.

The CA's response (CInitRes) holds the appropriate registration form, the CA's certificates and those needed to verify them, and a list of relevant revoked certificates (CRLs), all signed by the CA.

The cardholder now validates the certificates by traversing the trust chain and removes any certificates listed in the revocation list. The cardholder can now be confident that she has received a valid registration form from an accredited certificate authority. The registration form is filled out and includes the credit card number and expiry date. In order to send this sensitive data to the CA, it is encrypted directly with the CA's public RSA key exchange key (SET extra-strong encryption).

The cardholder nonce (Nonce_CARD) will be combined with a CA-generated nonce (Nonce_CCA) at the CA to form PANNONCE. This nonce is used in the payment protocol (see Figure 4.34). *H*(Card data,PANNONCE) will also appear in the cardholder's certificates.

The cardholder signs the public keys that are to be certified and includes these with the registration form to construct the certificate request message (CertReq). The request is encrypted before sending it to the CA, as shown in Figure 4.45.

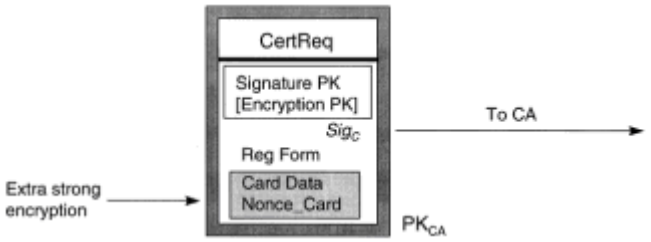


Figure 4.45 Certificate request message.

When the CA receives the certificate request, it authenticates the card with the issuing bank using an existing financial network. If the card data is valid, it will generate and sign certificates for the cardholder's public keys. The contents of these certificates are discussed in Chapter 3.

The certificate response from the CA (CertRes) contains the new certificates and the strongly encrypted Nonce_CCA, as shown in Figure 4.46. The cardholder verifies the certificates. She also combines the decrypted CA nonce (Nonce_CCA) with the cardholder nonce (Nonce_CARD) to form PANNONCE.

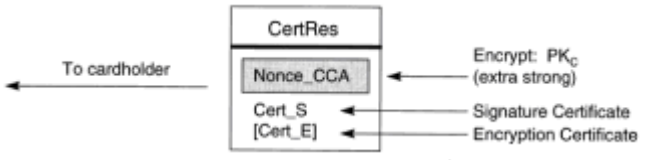


Figure 4.46 Certificate response message.

PANNONCE = *H*(Nonce_Card XOR Nonce_CCA)

It can then be verified that *H*(CardData,PANNONCE), which forms the globally unique cardholder ID, appears in the new certificates:

Unique Cardholder ID = *H*(CardData,PANNONCE)

The cardholder now has the signature certificate required to make a purchase, and can start making credit card payments to SET merchants.

4.9.8 Developing SET applications

SET is exclusively a payment protocol. Supporting electronic commerce protocols such as those for shopping, price negotiation, payment selection, and information delivery will often need to be incorporated with a SET implementation to provide a useful electronic commerce application. Similarly, configurable, simple-to-use graphical user interfaces need to be provided so that the purchase steps in a SET transaction are transparent to the user. An application using SET will typically contain much more functionality than just the SET payment protocol alone, as shown in Figure 4.47.

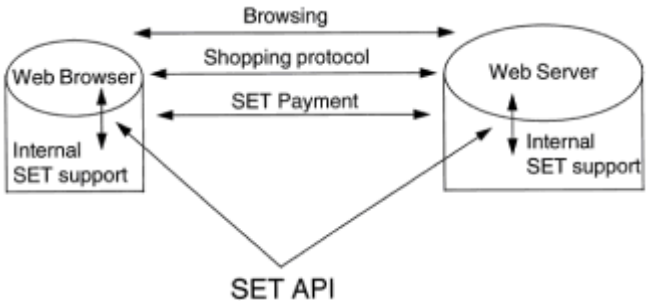


Figure 4.47: Integrating SET into end-user applications.

In order to prevent every SET application developer having to implement SET for integration into the application, several SET implementations, APIs, and toolkits are available for this purpose. This has the advantage of allowing applications to be developed independently and yet use the same SET implementation. MasterCard and VISA provide a reference implementation of SET that is freely available for noncommercial use.

4.9.9 Evolution of the SET standard

Table 4.7 shows how the SET standard has evolved. The first live SET transaction was performed in Denmark on the last day of 1996 in an IBM pilot scheme. SET pilots are planned for around the world during early 1997 but it will probably be late 1997 or early 1998 before SET gains widespread use. In the meantime, both VISA and MasterCard discourage the use of credit cards over open networks such as the Internet. They request that a cardholder use the telephone or the mail to get a card number to a merchant.

Table 4.7 Progression of the SET Standard

Date	Documents Released
Jan. 1996	Press release announcing SET
Feb. 1996	Draft for Public Comment Book 1: Business Description (68 pg.) Book 2: Technical Specification (269 pg.)
June 1996	Draft for Testing Book 1: Business Description (71 pg.) Book 2: Programmer's Guide (431 pg.) Book 3: Formal Protocol Definition (135 pg.)
Aug. 1996	Book 3 revised and updated Book 3: Formal Protocol Definition (146 pg.)
Early 1997	Piloting activity begins

4.10 Summary

In this chapter we have outlined a number of network payment methods using payment cards as the underlying method of achieving the financial transfer. The simpler methods, such as FV and CARI, involve no cryptography whatever, while SSL uses quite sophisticated cryptographic techniques to secure the link between customer and merchant. More complex three-way methods such CyberCash , *i*KP, and SEPP have laid the foundations for the SET protocol which, at the time of writing, is beginning to be deployed. It is expected that this protocol will supplant other schemes, and become the basis of all network transactions involving payment cards in the not too distant future.

References

[1]

Information Technology Partners, *Collect All Relevant Information (CARI)*, Milford, CT, 1995, http://www.netresource.com/itp/cari.html

[2]

Sandberg, R., *The Sun Network Filesystem: Design, Implementation and Experience*, Sun Microsystems, Inc., 2550 Garcia Ave, Mountain View, CA 94110, 1983, http://www.sun.com/solaris/networking/nfsindex.html

[3]

Bagwill, R., et al., “Improving the Security of NFS,” *Security in Open Systems -NIST Special Publication 800-7*, July 1994, http://csrc.ncsl.nist.gov/nistpubs/800-7/node148.html

[4]

Netresource, *Netresource Web Server*, http://www.netresource.com/itp/

[5]

Hickman, K., *The SSL Protocol*, Netscape Communications Corp., 501 E. Middlefield Rd., Mountain View, CA 94043, Feb. 1995, http://home.netscape.com/newsref/std/SSL.html

[6]

Kocker, P., A. Freier, and P. Karlton, *The SSL Protocol Version 3.0*, Netscape Communications Corp., March 1996, http://home.netscape.com/eng/ssl3/index.html

[7]

Simon, D., *The Private Communication Technology Protocol*, Microsoft Corporation, April 1996, http://www.microsoft.com

[8]

CyberCash, *CyberCash Web Server*, Reston, VA, 1996, http://www.cybercash.com/

[9]

Eastlake 3rd, D., et al., *CyberCash Credit Card Protocol Version 0.8*, RFC 1898, Feb. 1996, ftp://nic.ddn.mil/rfc/rfc1898.txt

[10]

Bellare, M., et al., “iKP - A Family of Secure Electronic Payment Protocols,” *Proc. 1st USENIX workshop on Electronic Commerce*, New York, NY, July. 11–12, 1995, http://www.zurich.ibm.com/Technology/Security/extern/ecommerce/iKP.html

[11]

MasterCard, *Secure Electronic Payment Protocol*, Oct. 1995.

[12]

MasterCard and VISA Corporations, *Secure Electronic Transaction (SET) Specification - Book 1: Business Description*, June 1996, http://www.mastercard.com/set http://www.visa.com/

[13]

MasterCard and VISA Corporations, *Secure Electronic Transaction (SET) Specification - Book 2: Programmer's Guide*, June 1996, http://www.mastercard.com/set http://www.visa.com/

[14]

MasterCard and VISA Corporations, *Secure Electronic Transaction (SET) Specification - Book 3: Formal Protocol Definition*, Aug. 1996, http://www.mastercard.com/set http://www.visa.com/