# Chapter 1 E-commerce and Cryptographic Techniques

## 1.1 E-commerce and digital payment systems

Information technologies are significantly changing the ways we store, distribute and access information. Financial information is also greatly impacted by these technologies. In recent years, electronic commerce (*e-commerce*) has created quite a buzz in the world of finance, commerce and trade as well as IT industry. So what is e-commerce?

E-commerce doesnπt always mean buying and selling goods and services on the Internet. In fact, it has been found [ITAA98] that the e-commerce activities are split between business-to-customer transactions (such as sales and customer service), and business-to-business transactions (such as those supporting sales, order processing, and resource management). Also it is discovered that the latter even has made more significant gains recently. Therefore, any activities involving financial, commercial or business information exchanges via telecommunication means such as telephone and computer networks may be regarded as e-commerce activities. In this thesis, the discussion is focused on one aspect of the business-to-customer transactions, the digital payment systems.

It is widely accepted that the Internet will be the main media and means for e-commerce because of its ever-increasing popularity. At the time of writing, it has been estimated that world total population online is 158 million [Nua]. However, among them, it is found that 82.5% do one, less than one or none purchase each month [GVU10]. And it is also discovered [GVU10] that only 4.3% are not at all concerned about security when making purchases or banking over the Internet. It is estimated [ITAA98] that more and more transactions are either solely Internet-based or based on a mix of proprietary networks and the Internet. Current e-commerce doesnπt fulfill peopleπs expectation for a secure and privacy-respecting e-commerce infrastructure. This state is in a large degree due to the vulnerable characteristics of Internet itself. We must accept the fact that e-commerce is still under definition and construction, and a lot of work is needed to do to make the Internet a mature digital market.

Digital payment systems have been the prime target for research in e-commerce for over two decades. This should not be a surprise. Commerce always involves a payer, (who is called Alice in this thesis), a payee, (who is called Bob) and at least one financial institution, (who is named Cyber Banque). Analogous to the current finance and trade markets based on traditional payment instruments such as cash, checks, credit cards, debit cards, etc, a digital market must be based on a robust, secure and efficient digital payment scheme. Only after such a digital payment system is built up, a digital market with all the traditional commerce elements such as shopping, bidding and bargaining, brokerage, and delivery may be constructed.

## 1.2 Cryptography

Security is the most concerned in a digital payment system. Cryptographic techniques are the most often applied in various payment systems for security purposes. This section starts the introduction on some basic techniques in cryptography by addressing variant security attacks against the communication across an open network

### 1.2.1 Security Attacks

In the context of the communication across a network, the following attacks can be identified:

- ← Disclosure: Release of message contents to any unauthorized person.
- ← Traffic analysis: Discovery of the pattern of traffic between parties.
- ← Masquerade: Insertion of messages into the network from a fraudulent source.
- ← Content modification: Changes to the contents of a message, including insertion, deletion, transposition, and modification.

- ← Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
- ← Timing modification: Delay or replay of messages.
- ← Repudiation: Denial of receipt of message by destination or denial of transmission of message by source.

### *1.2.2 Security Services*

Four general security services encompass the various functions required for an information security facility.

- ← *Confidentiality* ensures that the information is protected against unauthorized disclosure. In a digital payment system, disclosure of critical transaction information such as credit card numbers to illegitimate parties may cause a disaster.

- ← *Authentication* ensures that the origin of the information or the identity of a participant is correctly verified.

- ← *Integrity* ensures that the information is protected against unauthorized modification, and indicates whether or not such modification has occurred. In a digital payment system, attackers could attempt to change the transaction information by deleting, replaying or altering a message. For example, a malicious attacker could intercept the payment information from Alice to Bob, change some critical data such as the payment amount, and then retransmit it to Bob.

- ← *Non-repudiation* ensures that any party involved in the information transmission cannot deny either the participation in or the content of that transmission. In a digital payment system, for example, Alice could fraudulently claim that she did not, (in fact she did), authorize a payment to Bob, which could cause profit loss to Bob.

### 1.2.3 Cryptographic techniques

The first concern of cryptography is how to keep communications private. *Encryption* ensures privacy by keeping information hidden from anyone for whom it is not intended. Decryption is the reverse of encryption; it transforms encrypted data back into an intelligible form.

However, nowadays cryptography is more than encryption and decryption. Authentication is as important as privacy when information is communicated electronically. Digital signatures and digital certificates are among the cryptography mechanisms to offer authentication. In this section, these cryptographic techniques are discussed that are quite critical for developing digital payment systems.

#### 1.2.3.1 Encryption and Decryption

How to render a message incomprehensible to an unauthorized reader is the traditional aim of cryptography. The words, characters, or letters of the original intelligible message contribute the *plain text*. The words, characters, or letters of the secret form of the message are called *cipher text*. The process of converting plain text into cipher text is *encryption*. The reverse process of converting cipher text into plain text is *decryption*. An encryption/decryption algorithm and the associated encryption key and decryption key are required for the processes. (See Figure 1.) The algorithm is a set of mathematical rules to determine the transformation process of encryption and decryption. The keys act as a parameter of encryption/decryption algorithm and control the transformation processes. If $P$ is the plain text message, and $E$ and $D$ are the encryption and decryption algorithms respectively, it is required that

$$D(E(P)) = P$$

That is, decrypting the encrypted message results in the original plain text message.

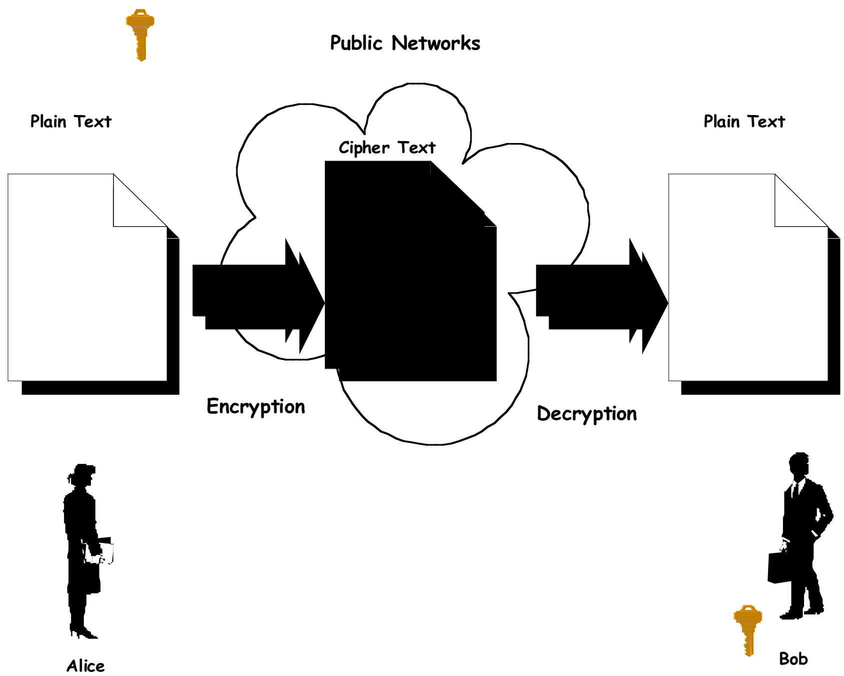


**Figure 1 - Encryption and decryption**

Confidentiality of transaction information in a digital payment system can be achieved by applying encryption techniques. Alice and Bob can agree on a certain method of encryption and decryption prior to the transmission of transaction data to ensure that the data is not understandable to others.

Note that, while modern cryptography is growing increasingly diverse, it is fundamentally based on problems that are difficult to solve. The problems are hard because they are intrinsically difficult to complete. Prime factorization of a large integer is an example of such problems. Given a large integer $n$, it is not easy to find the prime factors $p$ and $q$ such that $n = pq$. In fact, factoring is the underlying, presumably hard problem upon which several cryptography algorithms are based, such as the *RSA* algorithm, (refer to section 1.2.3.4). The security of the *RSA* algorithm depends on the unproved assumption of factoring problem being difficult and the presence of no other types of attack[1]. The theory of computational complexity is relevant here, since it classifies algorithms according to their difficulty. Difficulty in this case refers to the computational requirements in finding a solution. If an encrypted message cannot be decrypted by an attacker within practical time and with acceptable amount of resources, it loses its value for the attacker. In other words, whether a cryptography system is secure or not is relative. For example, in 1997, it was found that, a *RSA* 512-bit key might be factored for less than $1,000,000 in cost and eight months of effort [Rob95]. Therefore, an *RSA* 512-bit key at that time was not secure enough for corporate use, while it might be secure enough for casual personal use. Today, it is recommended the key sizes of 768 bits for personal use, 1024 bits for corporate use, and 2048 bits for extremely valuable keys [RSA98].

Encryption/decryption algorithms fall into two categories: symmetric algorithms and asymmetric algorithms.

← ***Symmetric encryption***

Symmetric encryption, also referred to as secret-key encryption, makes use of an algorithm that applies a unique key for both encryption and decryption. The key is often referred to as secret key.

With the plain text $P$ and the encryption key $K$ as input, at the side of the sender Alice, the encryption algorithm computes the cipher text $C$ denoted as

$$C = E(P, K)$$

The intended receiver Bob, in possession of $K$, is able to invert the transformation:

$$P = D(C, K)$$

[1] There has been some recent evidence that breaking RSA is not equivalent to factoring [DB98].

The fact is the encryption algorithm $E$ and the decryption algorithm $D$ are publicly known. The attacker Jack, observing $C$ and having knowledge of $E$ and $D$, but not possessing $K$ or $P$, will find that it is impractical to recover $P$ or $K$ or both $P$ and $K$.

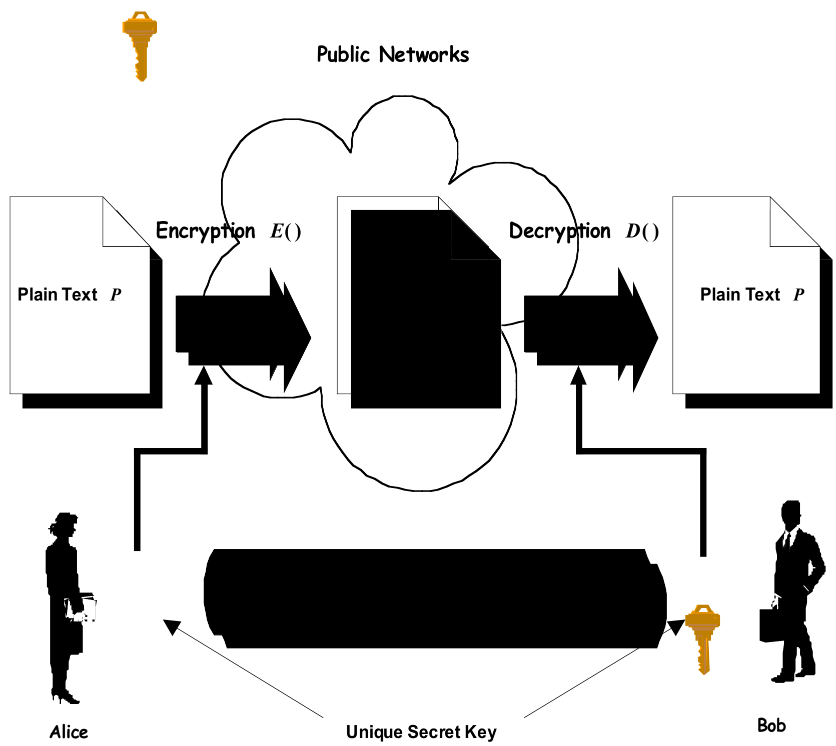


**Figure 2 - Encryption and decryption with symmetric algorithms**

The security of a symmetric encryption is based on the assumption that the sender and the receiver exclusively share the key. Therefore, when deploying symmetric encryption, the principal security problem is maintaining the concealment of the key. If Alice generates the key, she must also transmit it to Bob by means of some secure channel, such as a confidential mail. (See Figure 2.) Alternatively, a third party, trusted by both Alice and Bob, could generate the key and securely deliver it to both Alice and Bob.

The most widely used symmetric encryption algorithm is Data Encryption Standard (*DES*).

### ← *Asymmetric encryption*

Asymmetric encryption algorithms are also referred to as public-key algorithms. Instead of using the same key, an asymmetric algorithm makes use of two separate but mathematically related keys for encryption and decryption respectively. The creator of the key pair maintains one key secret (the *private key*) and makes the other key (the *public key*) known to anybody he or she may want to correspond with. A message encrypted with one key of the pair can only be decrypted by the other. It is not computationally possible to deduce one key from the other given knowledge of the algorithm. Thus, a message encrypted by Alice with Bobπs public key can only be decrypted by Bob using his private key. No other people can decrypt the message because only Bob knows his private key. (See Figure 3.)

With this approach, all participants have access to public keys of the correspondents, and private keys are generated and kept locally by each participant and therefore never need to be distributed. As long as the private keys are protected from disclosure, the communication is secure.

In this thesis, a public key is represented as the letter $K$ with a subscript naming the owner; for instance, $K_{\mathrm{Bob}}$ stands for Bobπs public key. The associated private key of Bob is represented as $K^{-1}_{\mathrm{Bob.}}$

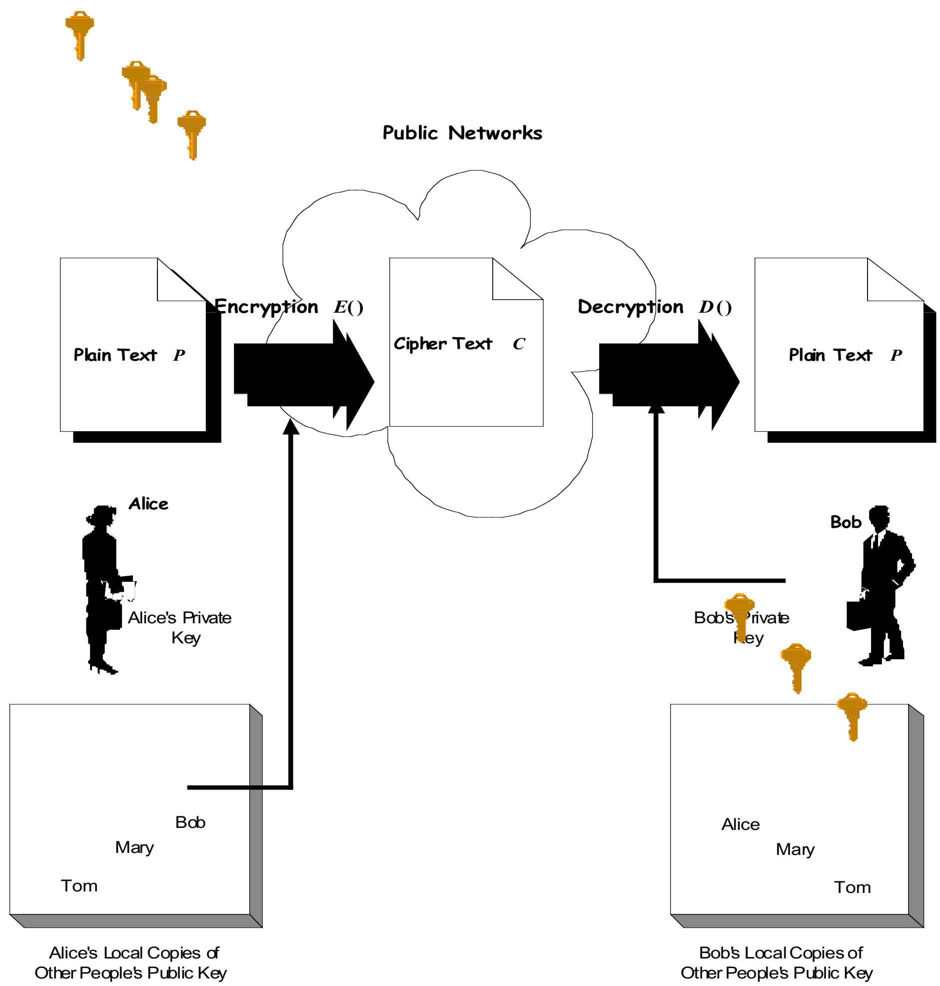


**Figure 3 - Encryption and decryption with asymmetric algorithms**

With the plain text message $P$ intended for Bob and Bobπs public key$K_{\mathrm{Bob}}$ as input, Alice forms the ciphered message $C$ with the encryption algorithm $E$:

$$C = E(P, K_{\mathrm{Bob}})$$

Bob, in possession his matching private key $K^{-1}_{\mathrm{Bob}}$, is able to invert the transformation to recover the plain text with the decryption algorithm $D$:

$$P = D(C, K^{-1}_{\mathrm{Bob}})$$

The attacker Jack, observing $C$, having access of $K_{\mathrm{Bob}}$ and having knowledge of $E$ and $D$, but not possessing $K^{-1}_{\mathrm{Bob}}$ or $P$, will find that it is impractical to recover $P$ or $K^{-1}_{\mathrm{Bob}}$ or both $P$ and $K^{-1}_{Bob}$.

The *RSA* algorithm is the most widely accepted and implemented general-purpose approach to asymmetric encryption. The *RSA* algorithm will be discussed at the end of the chapter.

#### 1.2.3.2 Digital signatures

Digital signatures are fundamental in authentication, authorization, and non-repudiation. Although asymmetric cryptography is the mostly employed means for digital signatures, symmetric cryptography can also be used to authenticate a message.

← ***Message authentication codes (MACs)***

A message authentication code (*MAC*) is an authentication tag (also called a checksum) sent along with the original message. A *MAC* is computed as a function of the message $M$ and a secret key $K$ shared by the sender and the receiver. That is, $MAC = f(M, K)$.

The receiver, knowing $f$ and $K$, can also compute the *MAC* and compare it with the received one. If they match, the messageπs integrity and the senderπsidentity are correctly verified.

It should be noted that a sender could repudiate a previously authenticated message by claiming the secret was somehow compromised by the recipient who shares the secret. There is no means within *MACs* to resolve this dispute.

← ***Hash functions and message digests***

A hash function $H$ is a transformation of the form:

$$h = H(x)$$

where $x$ is a variable-length message and $h$ is the fixed-length hash value. The hash value is also called the message digest. A hash function $H$ employed in cryptography is usually chosen to have also the following properties:

← $H(x)$ is easy to compute for any given $x$ ,

← $H(x)$ is one-way,

← $H(x)$ is collision-free.

For any given code $h$, if it is computationally infeasible to find $x$ such that $H(x) = h$, the function $H$ is referred to as one-way. For any given message $x$, if it is computationally infeasible to find $y$ … $x$such that $H(y) = H(x)$, $H$ is referred to as weak collision-free. If it is computationally infeasible to find any pair $(x, y)$ such that $x$ … $y$and $H(x) = H(y)$, $H$ is referred to as strong collision-free.

Two well-known hash functions are Message Digest 5 (*MD5*) and Secure Hash Algorithm (*SHA-1*).

One of the main roles of a cryptographic hash function is in the provision of digital signatures.

← ***Digital signatures***

It is a good solution to authenticate messages using a secret key if the parties involved are confident in the keyπs secrecy and if no disputes can arise. However, such a solution can never prove that a message came from the sender. Since both the sender and the receiver know the key, either one could have sent the message. Verifying the senderπs identity and resolving disputes require digital signatures.

A digital signature of a message is a block of data dependent on some secret known only to the signer, and, additionally, on the content of the message being signed. A digital signature is verifiable; if a dispute arises as to whether Alice signed a message, an unbiased third party should be able to resolve the matter equitably, without requiring access to Aliceπs secret information.

Thus, a digital signature scheme consists of a signature generation algorithm and an associated verification algorithm. A digital signature generation algorithm is a method for a signer to produce a digital signature on a particular message. A digital signature verification algorithm is a method for verifying that a digital signature is authentic, (i.e., was indeed created by the specified signer).

A digital signature generation algorithm (the signing process) must use some secret information that is only accessible to the signer. A digital signature verification algorithm (the verifying process), in contrast, uses some public information about the signer to verify the signature. Some public-key cryptography algorithms, among other mechanisms, can be deployed as digital signature schemes: the signer signs a message with his or her private key, while anyone else may verify the signature using the public key of the signer. It should be noted that with some public-key cryptography algorithms such as *RSA*, signing a message is actually employing the encryption algorithm on the message with the signerπs private key, and verifying a signature is actually employing the decryption algorithm on the signature with the signerπs public key. But this is not always the case for all public-key systems. The following discussions on digital signature mechanisms are based on public-key cryptography algorithms that are the likes of *RSA* algorithm.

Suppose Alice wants to send a signed message $M$ to Bob (see Figure 4). Generally the first step is apply a one-way hash function $H$ to the message, creating a message digest $h$.

$$h = H(M)$$

To create a digital signature $S$, the signing process usually signs the message digest $h$ instead of the message itself. In this way, it saves a considerable amount of time, because a message digest is shorter compared to the message itself.

Next, by encrypting the message digest $h$ with her private key $K^{-1}_{\text{Alice}}$, Alice actually creates a digital signature $S$ on $h$ with a secret of herself:

$$S = Sign(h, K^{-1}_{\text{Alice}})$$

Alice sends Bob the signature $S$ together with the message. In order for Bob to verify the signature, he must first apply the same hash function $H$ as Alice did to the message $M$ she sent him:

$$h_1 = H(M)$$

Then he verifies Aliceπs signature $S$ using her public key:

$$Verify(S, K_{\text{Alice}}, h_1)$$

The verification process above is actually accomplished in two steps. First, it decrypts $S$ with Aliceπs public key $K_{\text{Alice}}$ to get a value $h_2$ Second, it compares $h_1$ and $h_2$: if they are the same, it means that the signature is successfully verified; otherwise, the verification fails, that may suggest that either someone is trying to impersonate Alice, or the message itself has been altered since Alice signed it, or an error occurred during the transmission.

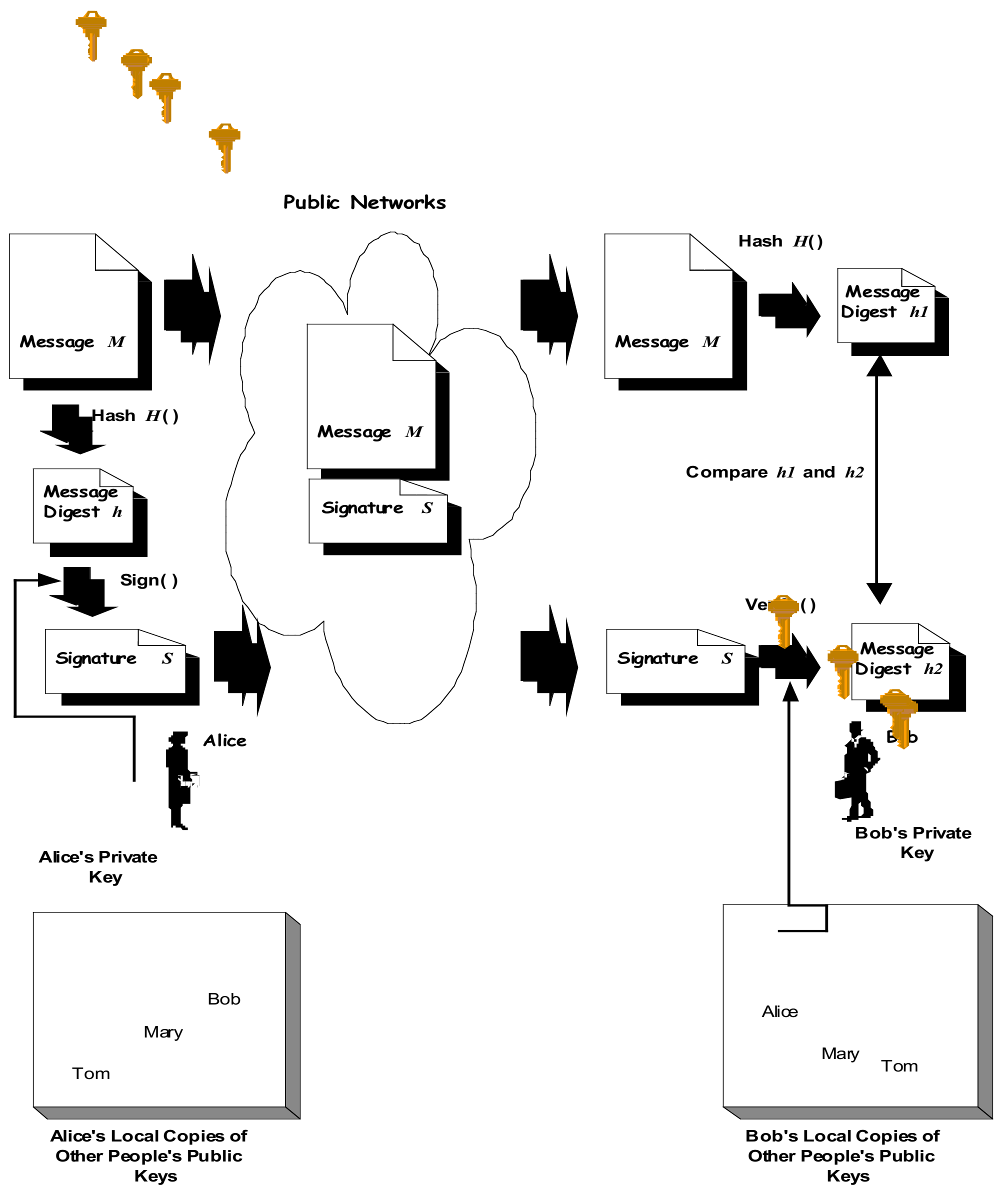


**Figure 4 - Digital signatures with public-key algorithms**

### *1.2.3.3 Certificates and certification authorities*

Public keys are sometimes considered simpler to manage because they can be distributed across insecure channels. However, for example, when Alice wants to communicate with Bob she must be assured that she is using Bobπs correct and authentic public key, and if she receives a signed message from Bob she must have a way to check that the signature is authentic and still valid.

The usual solution is for users to register their public keys with a trusted registry or certification authority (*CA*). The CA distributes certificates, which are public keys carrying the digital signature of the CA. To verify a certificate, a user needs only the CAπs public key that can be published in many ways and places.

A certificate contains more than just the public key. The userπs public key is combined with other significant information before signed by the CA. Such information may include the name of CA, validity period (start and finish dates) and the name of the user.

#### *1.2.3.4 RSA*

*RSA* is an asymmetric cryptography system that offers both encryption and digital signatures [RSA78]. In this thesis, *RSA* will be employed to offer some most important features of the presented digital cash system, so in this section, a brief description of the algorithm is given.

*RSA* works as follows:

1. Randomly generate two large primes, $p$ and $q$.
2. Compute their product $n = pq$; $n$ is called the public modulus.
3. Select a number, $e$, less than $n$ and relatively prime to $(p-1)(q-1)$, which means $e$ and $(p-1)(q-1)$ have no common factors except 1.
4. Compute another number $d$, the multiplicative inverse of $e$ modulo $(p-1)(q-1)$, which means $(ed - 1)$ is divisible by $(p-1)(q-1)$. ($d$ is also denoted as $e^{-1}$ or $1/e$.)

The public key is the pair (*n, e*); the private key is *d*. The value *e* is called the public exponent; the value *d* is called private exponent. The factors *p* and *q* may be secretly kept with the private key, or destroyed.

← ***RSA Encryption***

Suppose Alice wants to send Bob a message *p*, and she is in possession of Bobπs public key (*n*, *e*). She creates the cipher text *c* as:

$$c = p^e \bmod n$$

She sends *c* to Bob. By decrypting it with his private key *d*, Bob get the plain message:

$$p = c^d \bmod n$$

The relationship between *e* and *d* ensures that Bob correctly recovers *p*. Moreover, since only Bob is in possession of *d*, no one else other than Bob can decrypt this message.

← ***RSA Digital Signature***

Suppose Alice wants to send Bob a digitally signed message *p*, and her public key (*n*, *e*) is known to Bob. Alice creates a digital signature *s* on *p* with her private key *d*:

$$s = p^d \bmod n$$

She sends *p* and *s* to Bob. To verify the signature, Bob checks if the following is satisfied:

$$p = s^e \bmod n.$$

# Chapter 2 Digital Payment Systems

In Chapter 1, several cryptographic techniques, encryption/decryption, digital signatures and digital certificates, are discussed respectively. In this chapter, applications of these techniques on various digital payment systems are addressed. Digital payment schemes generally have three models: *secure credit card systems*, *credit-debit systems*, and cash-like system, (also called *digital cash systems*).

## 3.1 Secure credit card systems

This model enhances the traditional credit card payment system by securely protecting the credit card number from eavesdropping. This model is anticipated to be the most acceptable to all entities because it is based on a mechanism that both consumers and merchants are familiar with, and it maintains the benefits of the government and the traditional financial institutes. One proposal of this payment model, *iKP*, is shown in the following subsection, and other well-known proposals are addressed at the end of the section.

### 1.3.1 iKP

The *iKP* protocols are a family of protocols ≠ *iKP* (*i = 1, 2, 3*) ≠ for secure digital payments over the Internet. The protocols implement credit card-based transactions between customers and merchants using the existing financial networks for clearing and authorization. [iKP95]

The reason *iKP* was focused on the credit card payment model is that its developers believed that this model ≥is anticipated to be the most popular in the future.≤ However at the same time, it is claimed that they ≥can be extended to apply to other payment models, e.g., debit cards and electronic checks≤. [iKP95] *iKP*s make use of public-key cryptography. As the increase of the number of parties that possess their own public/private key pairs, indicated in the name of each protocol by *i*, the *iKP* protocols offer increasing level of security and complexity.

- *1KP*, the simplest in the family, requires that only the ≥acquirer gateway≤ (the *iKP* term for a bank) possess public/private key pair. Customers and merchants need only to possess the authentic public key of the gateway, or the authentic public key of an ≥authority≤ that validates the gatewayπs public key via a signed certificate. [iKP95]

  When Alice makes payment to Bob, she encrypts her credit card number and possibly associated PIN with the public key of Cyber Banque, and binds them with the relevant purchase information. She then sends the message to Bob, who immediately forwards this message to Cyber Banque for verification. Cyber Banque decrypts the message with its private key, checks the card number and possibly the PIN. If they are valid, the bank will authorize the transaction by digitally signing it with its private key, then return it to Bob. It is easy to see that customers of *1KP* are actually authenticated based on their credit card numbers, and possibly associated secret PINs. In addition, since Aliceπs card number and PIN are encrypted with Cyber Banqueπs public key, therefore no one except Cyber Banque will be able to uncover the secrets.

  *1KP* does not offer non-repudiation for messages sent by customers and merchants. There is no mechanism for the acquirer to get a proof of transaction authorization by merchants. The proof of transaction authorization by customers are the credit card numbers and PINs, which are deniable proof since the disclosure of these secrets is not impossible.

- *2KP* requires that merchants should, in addition to acquirer gateways, possess public/private key pairs and key certificates. Non-repudiation is consequently offered for messages originated by merchants. This is achieved by requiring the merchants digitally signing the transaction data with their private keys. Moreover, *2KP* enables customers to verify that they are dealing with genuine merchants. This is achieved by checking the merchantsπ certificates issued by a central certification authority (CA). In *2KP*, still no mechanism for authenticating payment orders originated by the customers, therefore non-repudiation is not offered for messages proposed from the customer.

← *3KP* requests that each involved party has a public/private key pair, therefore it provides full multi-party security. It achieves non-repudiation for all messages of all parties involved. Payment orders from the customer are authenticated both by the credit card number (and PIN), and a digital signature of the customer generated with her private key. Merchants consequently can authenticate the customer with her public key. A CA is required to provide certificate of the customerπs public key as before.

The developers expect that, as public key technology becomes more pervasive, more and more parties will hold public/private key pairs. A gradual deployment of the *iKP* protocols thus makes sense: begin with *1KP*, then move to *2KP* and finally to *3KP*. [iKP95]

*iKP*s provide a customer some privacy or anonymity against the merchants in the sense that ≥the customer uses a pseudo-identity ä which is different in each transaction≤. However, since the system follows the credit card-based payment model, it requests the customers to reveal their identities for the credit card number verification, so it doesnπt offer customers anonymity against the payment system provider.

### *1.3.1 Other systems in this category*

There are many other digital payment proposals which fall into the category of secure credit card systems, such as *SET* (Secure Electronic Transaction) [SET] and *CyberCash* [CyberCash]. One of the common features among them is that they all deploy cryptography techniques for security reasons, and as more parties involved possess a public/private key pair, more security features are achieved, just like *iKP*.

## 3.2 Credit-Debit Systems

The common feature of payment systems in this category is there is a centralized accounting server who transfers balances between two accounts. There are no separate, identifiable digital tokens that can be stored on a hard drive or in the memory of a smart card, and subsequently transferred to others in payment without going through the centralized server. In other words, one can send a digitally signed payment message to the accounting server which results in a balance transfer, but one doesn't receive from the server a digital token or piece of digital cash carrying the server's signature, and promising to work as payment instruments on demand. A typical proposal in this category, *NetBill*, is described in this section.

### NetBill

*NetBill* is designed by Carnegie Mellon University and Mellon Bank Corp. for micro-payments, especially for information delivered over the Internet, such as payment of a few cents for access to a Web page or an electronically archived research journal. The system attempts to guarantee that customers receive the information they request, and that merchants receive payment for goods delivered. [NetBill]

*NetBill* uses an account server, which maintains accounts for both customers and merchants, linked to conventional financial institutions. The *NetBill* server acts as an aggregator to combine many small transactions into larger conventional-sized transactions.

The payment structure is designed for the easy construction of pseudonyms to allow buyers to protect their identities.

The basic steps in the payment protocol are the following:

1. Alice requests a price quote from Bob.

2. Bob responds with a quote to Alice.
3. Alice accepts or rejects the price quote.
4. If she accepts, Bob delivers the information in encrypted form.
5. Alice sends an electronic payment order to Bob.
6. Bob sends the electronic payment order and key to the *NetBill* server.
7. The *NetBill* server sends a receipt to Bob.
8. Bob sends a receipt to Alice, allowing Alice to decrypt the information.

In step 1, Alice and Bob authenticate each other using public-key certificates. They establish a symmetric session key to encrypt subsequent messages. Bob's price quote (step 2) will be based on Alice's identity (to allow for price discrimination). If Alice accepts the quote (step 3), Bob sends the purchased information to her in an encrypted form while withholding the encryption key *K* (step 4). Alice makes up an electronic payment order that describes the transaction and includes a message digest of the information from Bob in step 4. The payment order is signed with the private key of Alice and sent to Bob (step 5). Bob verifies the message contents using Alice's public key, appends the key *K*, endorses the order with his digital signature, and sends it to the *NetBill* server. The *NetBill* server checks Alice's account balance, debits this balance and credits Bob's balance, and sends a digitally signed receiptãwhich includes the key *K*ãback to Bob (step 7). Bob forwards the receipt to Alice (step 8), who now has the key *K* to decrypt the purchased information.

There is little personal anonymity in this process: the *NetBill* server keeps transaction records of purchases made from different merchants, and this potentially allows all information requests to be linked.

## 3.3 Digital cash systems

In a digital cash system, there exists at least one bank, (a cash server usually), minting digital cash which a customer can withdraw from and deposit to the bank, keep in his or her digital purse, (which is usually a smart card or a PC hard disk), and pay to others.