SOME SECURITY ISSUES FOR E-COMMERCE

Fred Piper and Chris Mitchell
Information Security Group
Royal Holloway
University of London
Egham, Surrey TW20 0EX, UK.
{F.Piper,C.Mitchell}@rhbnc.ac.uk

Introduction

Conducting business over open, insecure networks, such as the Internet, has introduced a number of potential security hazards. In the pre-computer era, those involved in buying and selling relied on a number of social 'mechanisms' for security. These included, for instance, face-to-face recognition of one's colleagues, reliance on the introduction of strangers by people we know, assurance obtained from the use of hand-written signatures, faith in the privacy of letters inside sealed envelopes, and the concept of holding a secret conversation behind closed doors. When e-commerce transactions are conducted over open networks then these social mechanisms disappear and many new opportunities for fraud are created. For example frauds may occur in transaction systems by people impersonating legitimate users and gaining improper access, or by hackers gaining access to critical data which is held on-line and, depending on the environment, either reading, altering or destroying it. Similarly, unless appropriate precautions are taken, anyone storing information on a database may not be able to control who else has access to that information. Thus anyone sending information over a public network or storing it on a database should ask themselves:

- Am I happy for other people to read the contents?

If the answer is YES then there is no problem, but if it is not then they may need to ask:

- How can I stop them?
- What does it cost to stop them?
- Is there any legal restriction on what I can do to stop them?
- Do I need acknowledgement of delivery?

Similarly there are a number of questions which the recipient of transmitted information may need to ask him or herself. They include:

- Am I confident I know the identity of the sender?
- Am I happy that the message received is identical to the one sent by the originator?
- Am I concerned that the sender may later deny sending the message and/or claim to have sent a different one?

Clearly the precise security needs of any particular company depend on its perceptions of its vulnerabilities and how it manages the risks. However, in general

terms, information security relates to the provision of CIA: Confidentiality, Integrity and Availability.

Confidentiality is concerned with preventing unauthorised users reading information to which they are not entitled. Integrity is about ensuring that things are as they should be. In particular, the provision of data integrity usually implies providing a method of detecting unauthorised modifications of data, e.g. changes, deletions, additions and replays. Availability is about ensuring that a system's services are accessible on demand by authorised entities. One particular aspect is the prevention of denial of service attacks.

In the context of e-commerce transactions, users may wish their credit card details and/or information about what they are buying to be protected from eavesdroppers, merchants may not want anyone to know the discounts offered to various customers, and financial institutions must protect the account details of their customers. Thus there are certainly some confidentiality issues which need addressing. However, the major concerns are probably related to checking the identify of users, the availability of networks, and protecting the integrity of information.

Cryptographic solutions

Various security mechanisms are available for the provision of confidentiality and/or data integrity and a number of these involve the use of cryptographic algorithms. We now give a brief introductory overview of cryptography without restricting ourselves to e-commerce applications.

Cryptography is a useful tool in the provision of security and, in addition to the confidentiality and data integrity, cryptographic algorithms can be used to provide user authentication and non-repudiation. Furthermore, cryptography based challenge-response protocols are now commonly used to improve access control by the introduction of, for example, dynamic passwords.

The effectiveness of each of these cryptographic services is, of course, dependent on the strength of the algorithm, together with the physical security of the hardware. However, the accompanying key management procedures are equally important.

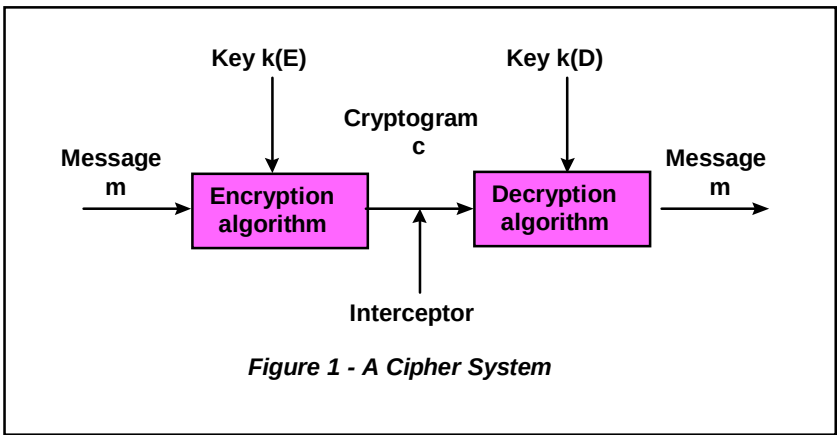

Figure 1 - A Cipher System

The idea of a cipher system, shown diagrammatically in Figure 1, is to disguise confidential information in such a way that its meaning is unintelligible to an unauthorised person. The information to be concealed is called the **plaintext** (or just the **message**) and the operation of disguising it is known as **enciphering** or **encryption.** The enciphered message is called the **ciphertext** or **cryptogram**. The

message is scrambled by using an encryption algorithm and the cryptogram will depend on both the message and an **enciphering key k(E)**. In order that the recipient can obtain the message from the cryptogram there has to be a **deciphering algorithm** which, when seeded by the appropriate **deciphering key k(D),** reproduces the plaintext from the ciphertext.

Even if they know the deciphering algorithm, any third party that intercepts the cryptogram will not, in general, know the deciphering key and it is this lack of knowledge that, it is hoped, will prevent him from knowing the plaintext.

In practice most attacks involve trying to determine the deciphering key because, if successful, the interceptor will then have the same knowledge as the intended recipient and will be able to decipher all other communications until the keys are changed. However there may be instances where an attacker's sole objective is to read a particular message.

One important fact should already be clear from our introduction. That is, knowledge of the deciphering key is not necessary for generating the ciphertext from the message. This simple observation has led to a natural division into two types of cipher systems.

A cipher system is called **conventional** or **symmetric** if it easy to deduce the deciphering key k(D) from the enciphering key k(E). However if it is computationally infeasible to deduce k(D) from k(E) then the system is called **asymmetric** or a **public key system**. The reason for distinguishing between these two types of system should be clear. In order to prevent an interceptor with knowledge of the algorithm from obtaining the plaintext corresponding to intercepted ciphertext it is essential that k(D) should be secret. Whereas for a symmetric system this necessitates that k(E) should also be secret, if the system is asymmetric then knowledge of k(E) is of no practical use to the attacker. Indeed it can be, and often is, made public.

Symmetric Algorithms

Although the statements made in the last paragraph may appear to be simple and self-evident, their consequences are far reaching. In Figure 1 our diagram assumes that the sender and recipient have a matching pair of keys. It may, in practice, be quite difficult for them to reach this situation. In fact the general problem of key management, which includes key generation, distribution, storage, change and destruction, is one of the most difficult aspects of obtaining a secure system. The problems associated with key management tend to be different for symmetric and asymmetric systems. If the cipher is symmetric then there is a need to able to distribute keys while keeping their values secret. If the cipher is asymmetric then it is possible to avoid this particular problem. However it is then replaced by a different, but not necessarily easier, problem. When an asymmetric cipher is used to provide confidentiality, it is absolutely crucial that the sender is sure that the enciphering key is authentic, in the sense that they have complete confidence in the identity of its owner. The problem of guaranteeing the authenticity of enciphering keys for a public key system should not be underestimated and frequently involves the use of so-called trusted third parties.

One consequence of accepting that an attacker will know the algorithm is that we have to assume that the only information which distinguishes the genuine recipient

from the interceptor is knowledge of k(D). Thus the security of the system is totally dependent on the secrecy of the deciphering key. This reinforces our earlier assertion about the importance of good key management.

We must stress that assessing the security level of a cipher system is not an exact science. All assessments are based upon assumptions, not only on the knowledge available to an attacker but also on the facilities available to them. The best general principle is, undoubtedly, when in doubt assume the worst and/or err on the side of caution. It is also worth stressing that, in general, the relevant question is not 'is this an exceptionally secure system?' but, rather, 'is this system secure enough for this particular application?' It is also important to consider how else the information might be exposed when trying to assess the level of cryptographic protection it warrants. A frequently quoted illustration concerns credit card details which are frequently transmitted in clear over telephone links and are entrusted to shop assistants, waiters etc.

Key Searches

If we assume that our deciphering algorithm is known then there is one obvious method of attack available to the interceptor. They could, at least in theory, try each possible deciphering key and hope that they identify the correct one. Such an attack is called an **exhaustive key search**. Of course such an attack cannot possibly succeed unless the attacker has some way of recognising the correct key or, as is more common, is at least able to eliminate some obviously incorrect ones. If, for instance, some corresponding plaintext and ciphertext is known then it is clear that any choice of k(D) which does not give the correct plaintext for all the corresponding ciphertext cannot possibly be the correct key.

We are already in a position where we can begin to give some very basic criteria for assessing the suitability of a given cipher system for any particular application. The designers will (or, at least, should) know the size of their key space and, if they make assumptions about the speed with which an attacker could try each key, they can estimate the expected time for an exhaustive key search to reveal the key. If this latter time is unacceptably short then the system is clearly too weak.

As an example of the time needed for key searches we note that it is possible to build, for a cost of about US $130,000, a purpose-built machine that will complete a 56-bit key search, (for a specific symmetric algorithm known as DES), in less than 10 days. Anyone who intends to use DES needs to be aware of these facts and decide whether they make DES too weak for their application.

Practical Systems

There is a theoretically unbreakable cipher. It is called the **one-time-pad** and can be used to protect any message. However it has the property that the key has to be as long as the message and each key should be used only once. This forces very strict key management control restrictions and makes it unusable for most practical systems with a large number of users.

Since the one-time-pad is (essentially) the only unbreakable cipher system but is unusable on large networks, it is an inescapable fact that almost all cryptographic systems in use today are **theoretically** breakable. Fortunately this does not mean that messages cannot be transmitted secretly, and the reason is the emphasis on the word

theoretically. Clearly it does not matter if an attacker can determine your secret deciphering key in, say, a year if you only require secrecy for a day or so. Unfortunately it is difficult to ascertain whether or not this second criteria is satisfied. This is why designing good encryption algorithms is difficult.

Digital Signatures

The fundamental idea behind a public key cipher system is that each user has a secret key that is available only to them and a public key that is known by everyone. Earlier we assumed that the public key would be used to encrypt messages that only the owner of the corresponding secret key can decrypt. However, it is also possible to use a public key system to provide digital signatures. A **digital signature** is a cryptographic checksum which can be appended to a message to assure the receiver of the identity of the sender and that the message has not been altered in transit. The sender must be the only one able to generate a valid signature for a message, whereas all users, or at least all users within a defined group, must be able to check whether or not a signature is valid.

A digital signature can also be retained by the recipient as evidence to convince a third party that the message originated from the alleged sender. Since a digital signature has to identify the user uniquely it must depend on a secret parameter available only to that user. This requirement makes it natural to try to use the secret key of a public key system as that parameter.

Can Public Key Systems be Secure?

The theory behind public key systems often baffles the non-mathematician who asks 'how is it possible to know the public key used for enciphering but for it to be computationally infeasible to reverse the process to decipher the message?' Although this may sound implausible there is an analogy which may help explain it. Suppose that you and a friend were in a closed room with no telephone. Suppose also that you both have copies of the London telephone directory. Your friend takes the directory and looks up someone's telephone number. They then tell you the number and ask you the name and address of the person. You know exactly what they have done. Nevertheless you might find the prospect of reversing the process somewhat daunting and might even be prepared to regard it as infeasible. Of course, you know exactly how to reverse the process. You could, for instance, start at the first page and work your way through all the names until you found the correct number. It is not the theoretical difficulty that deters you but the sheer magnitude of the task. Indeed, if your friend had used the directory for a small company, you might have accepted the challenge with optimism.

A mathematical function is said to be **one-way** if it is easy to perform but computationally infeasible to reverse. The one way functions used in public key cryptosystems tend to involve number theoretic concepts. The simplest example of such a function is the multiplication of two integers. While it is straightforward to compute the product of any two integers it is much harder to factor a large number, i.e. to find the numbers which multiply together to give this larger number. Precisely how difficult it is, is not relevant to an article like this. The relevant facts are that factoring is easy for small numbers and, in some sense, gets more difficult as the number gets larger. In theory we know how to factor all integers but the time

required for a computer to perform the task becomes astronomic as the size increases. (Note that this is precisely the same situation as for the telephone directory analogy.)

The Use of Number Theory

The two most commonly used public key algorithms are known as **RSA** and **El Gamal**. We will not discuss them in detail. However, they both use number theoretic techniques and this has a number of practical consequences. The first is that they need to operate on large block sizes to be secure. As an example, the block size for an RSA application is influenced by the likelihood of an attacker having the resources and capability to factor a number of that size and being willing to commit them to the task. Clearly the precise size of the block depends upon the application and the value of the data being protected. However, it is frequently very difficult to arrive at realistic assessments. The last fifteen years have seen unprecedented advancements in factorisation.

Most RSA systems have block sizes of at least 512 bits. Many people feel that, for high value transactions, this is dangerously small and block sizes of 640, 768, 1024 or even 2048 are not uncommon. Not surprisingly, increasing the block size increases the computations involved and public key systems tend to be significantly slower that their symmetric counterparts.

As a result they are not usually employed to encrypt long passages of text. Their main applications are digital signatures and as key encrypting keys for systems which use symmetric ciphers to encrypt data.

Attacks on Digital Signatures

The basic principle of a digital signature scheme is that each user has a secret value that only they can use, and whose use is accepted for identifying them. However, corresponding to each secret key is a public key that can be used to confirm that their secret key was used. If an attacker wishes to impersonate a genuine user then two obvious methods are to attempt to:

obtain the use of that user's secret key

substitute their own public key for that of the genuine user.

In the first scenario they are forging a signature in the sense that they are producing the same signature value that the genuine use would have produced. In the second scenario they are attaching their own signature but that signature is accepted as genuine because verifiers will use the wrong public key.

Obtaining use of the secret key

If an attacker is to obtain the use of someone else's secret key then they must either gain knowledge of that secret key or obtain use of a signing device which contains it.

The most obvious attack is probably that of trying to compute the secret key directly from the public key. Using suitably chosen algorithms and large keys prevents this. However, the key generation process may be an easier target for would-be attackers, and the importance of this process should not be overlooked.

There are various levels of physical security that can be provided for secret keys, and these may include tamper-resistant features of the devices in which the keys are stored and/or physical features of the locations where the devices are stored.

Most of the problems associated with protecting the secret key are those that face all users of cryptography and, as a result, are not unique to public key systems.

Substitution of public key

In order to provide users with the ability to obtain authentic copies of other users' public keys most systems rely on Certificates and Certification Authorities.

Certification Authorities

A **Certification Authority** (abbreviated to **CA**) is a trusted entity that issues certificates which bind an entity *E* to its public key value PK_E. Each certificate consists of a message containing *E*'s identity and *E*'s public key value, digitally signed by the CA. Anyone who is in possession of *E*'s certificate and the CA's public key can now obtain assurance of the authenticity of *E*'s public key by verifying the CA's signature on the certificate. If someone claims to be *E* then this can be verified by sending a 'challenge' to *E*, and where the response from *E* is a signed message containing the challenge. It is important to notice that it is not the possession of the certificate which establishes *E*'s identity but, rather, the use of the secret key which matches the public key value in the certificate.

This solution to the secure public key distribution problem is attractive because the overhead involved for users is minimised: the secure distribution of all users' authentic public keys is reduced to the secure distribution of one CA's authentic public key. However, users are required to trust the CA to sign only bona fide certificates. Provided this trust is justified they can have confidence in the identity of the owners of public keys and thereby avoid impersonation attacks.

Revocation

In most systems, it is unreasonable to expect that the private keys of users will never be compromised or that there will not be other reasons for wishing to invalidate an existing certificate. Therefore it is necessary to have a mechanism in place so that users can revoke their certificates.

Since the CA is a trusted entity and is responsible for producing certificates, the burden for maintaining this revocation mechanism usually falls on the CA. One revocation notification mechanism typically takes the form of a certificate revocation list (abbreviated to CRL) – this is simply a list of the revoked certificates all signed by the CA. The list is either stored in a public directory maintained by the CA or distributed directly to users.

Each time a user wants to rely on a certificate, it must check that it is valid. Thus, if CRLs are implemented, users may need to access them frequently.

Cross Certification

Typically, the CA's public key will be given to *E* at the same time that *E* gets a certificate. However, in systems where more than one CA is operating, *E* may also need access to the public keys of the other CAs.

In practice it will be impractical for *E* to visit all the other CAs to collect their public keys. Instead *E's* CA may **cross certify** the public keys of the other CAs.

In this approach, the CA obtains the public keys of the other CAs and gives copies of the keys to *E*, either by handing them over in person, or by issuing special authority certificates binding each CA to its public key.

A related approach is to implement a hierarchy of CAs. Here high level CAs certify lower level CAs. Now all *E* needs to verify any entity's certificate is the public key of the highest level 'root' CA and a trail of authority certificates leading to the entity's certificate.

Some Problems

The theoretical model works provided that each CA is trustworthy. However, a number of obvious liability issues arise. Who is liable if, for instance, a fraudster manages to con a respectable CA into issuing him with a certificate? (Note that many people who place reliance on the certificates will have no direct relationship with that CA.)

An even more fundamental issue concerns the legal status of a digital signature. In many situations there is no point in relying on digital signatures unless they represent a binding commitment by the signer. Most countries are now contemplating the introduction of some form of digital signature legislation. However, there is no guarantee that their solutions will be mutually compatible.

An Application

In order to illustrate the potential of certificates we will now show how two people with public key certificates from the same CA can establish a common secret session key for a symmetric algorithm such as DES.

We assume that two users, whom we call *A* and *B*, generate their own public, secret key pairs for a public key system. We denote *A's* key pair by PKA and SKA with a similar notation for *B*.

Each of them goes to the certification authority and, after establishing their identities, is given a copy of the CA's public key plus certificates, $CERT_A$ and $CERT_B$ respectively. Each of these certificates contains the respective user's identity and their public key value and is, of course, signed with the CA's secret key.

If *A* now wishes to establish a DES key with *B* then they might follow the following protocol:

- *B* sends $CERT_B$ to *A*, who uses the CA's public key to check the signature and, assuming it is correct, now has an authentic copy of *B*'s public key.
- *A* generates a DES key and encrypts it with *B*'s public key. *A* then signs this encrypted value, together with the identity of *B*, with its own private key. The result is sent to *B* together with a copy of $CERT_A$.

B uses their copy of the CA's public key to check the signature on $CERT_A$ and, provided it agrees, now has an authentic copy of *A*'s public key. *B* uses this to verify the signature on the encrypted key value and, provided it agrees, is now confident that the encrypted key came from *A*. *B* then uses his own secret key to decrypt the received value and obtain the key. Clearly *B* now knows the key and is confident that it came from *A*. However, *A* used *B*'s public key to encrypt the key, so *A* is confident

that only *B* can decrypt it. Thus, at the end of this exchange of messages, both parties share a common key and are confident that they are the only two people who know the key.

However, one thing that is not provided is 'freshness', i.e. *B* has no way of knowing whether the message from *A* is newly generated or is a replayed copy of an old message. An interceptor who has obtained the key it contains might replay an old message – the interceptor will then be able to read secret messages sent by *B* encrypted using this 'old' key. This problem can be avoided in a number of ways, e.g. including the date and time in the encrypted message sent by *A*, although this is just one aspect of the much larger topic of secure protocol design.

Conclusion

In this short article we have provided a basic introduction to cryptography, illustrated a number of practical applications and hinted at some of the problems associated with establishing a secure network. Information Security is an area of tremendous interest whose importance increases daily. It is likely to impact on the success of e-commerce.

Further reading

For an introduction to cryptography see:

- R.E. Smith, *Internet Cryptography*, Addison Wesley, 1997.

For a much more detailed treatment of the subject, by far the best book is:

- A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.