Communications Security Principles

This lecture describes the principles used to ensure the secrecy,integrity, and authenticity of communications.

Introduction

Goals

Data communication security has the following goals:

secrecy ensures that the data being communicated are not disclosed. Information *about* the contents, including its existence, is known as "metadata'. Metadata must often be protected as well.

authentication identifies the parties communicating. The goal is to prevent one party from impersonating another.

integrity tries to ensure that the data is not modified by the communication system. This could be intentional or unintentional.

Exercise 1: How important are each of the above goals for the following applications: access to a news web site, downloading free software, web access to a banking system, deciding to allow access to an ISP's network.

Exercise 2: Can you think of an example where metadata might need to be protected?

Security Layers

Security can be applied at different layers of the protocol stack:

link layer provides security between two physical devices. An example is the WPA2 protocol for WLAN security.

network layer provides security between two networks or hosts. An example is a virtual private network (VPN)

application layer provides "end-to-end" security between two applications. A common example is the TLS protocol (formerly SSL) used for the HTTPS scheme (secure HTTP).

Each of these can, but need not, provide secrecy, authentication and integrity.

Which of these is depends on the expected threat. For example WLAN link-layer security protects against eavesdropping by nearby devices but does not protect data after it leaves the wireless LAN. Network layer security can protect against interception or modification as packets travel over a WAN. End-to-end security protects against all of these threats but requires that applications and application-layer protocols include security features.

Limitations

Physical security, competent, trustworthy people and bug-free software are a pre-requisite. Security methods described below cannot protect from various other threats such as compromised hardware or software, buggy software or human error or wrongdoing. If the hardware (e.g. storage media, firmware or hardware) or the software can be modified by an attacker the security protocols can be rendered ineffective. Similarly, buggy software can be exploited. Often the simplest and least expensive way to circumvent security is to bribe, threaten or use "human engineering" on personnel.

All security is eventually broken. For various reasons it's usually impossible to prove, in a mathematical sense, that a communication system is secure. Because the information being communicated is often valuable, there is a constant incentive to find ways to subvert communication security. Over time, all security systems have been "broken" and there is no reason to expect this trend will end. Most attacks also try to make it difficult to determine that the security mechanisms have been compromised (they "cover their tracks"). So you can never be sure that you system is truly secure.

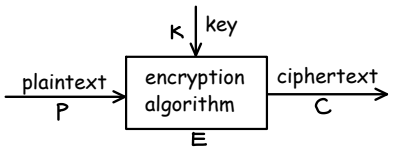
Avoid ad-hoc solutions. Since so much effort has been devoted to subverting security mechanisms, it

is important to choose well-tested security methods and to follow "best practices." Ad-hoc solutions are likely to be easily ineffective.

Cost vs benefit. Security mechanisms always have a cost. It may be in the form of lower reliability, more expensive hardware, more processing time, or inconvenience for users. Selection of appropriate security mechanisms therefore requires a cost-vs-benefit analysis to ensure the cost is appropriate for the value of the information being protected.

Encryption

The basic tool for communication security is encryption. Modern encryption techniques use a publicly-known algorithm to convert "plaintext" into "ciphertext" using a secret "key":



If implemented correctly, the ciphertext cannot be decrypted without knowing the key.

We always assume the encryption algorithm is known to everyone and only the key is secret. The use of secret (and thus untested) encryption algorithms is known as "security by obscurity" and is poor practice.

Cryptanalysis is the process of trying to deduce the key and/or plaintext from the ciphertext. Since in many situations the plaintext will be known, encryption algorithms are designed to prevent cryptanalysis with "known-plaintext" attacks.

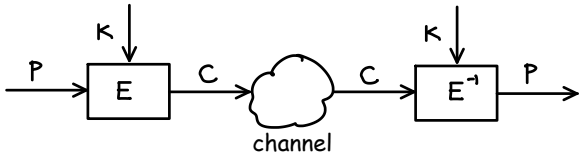
The simplest and often the only known cryptanalytic attack is to try all the possible keys until a recognizable message is decrypted. For these types of attacks the difficulty of decrypting the message increases exponentially with the length of the key.

Exercise 3: If you could test one key per nanosecond, how long would it take, on average, to find the key if it was 32 bits long? 128 bits? 2048 bits?

In practice, security systems are most often subverted by the flaws in the software (e.g. bugs or malware), hardware (information leakage) or people (disgruntled or dishonest employees).

Symmetric Encryption

In symmetric encryption the same key is used for encryption and decryption:



The main problem with symmetric encryption is the need to securely distribute keys. Often, a master key is distributed and is used to periodically transmit a separate "session" key. The session key is used to encrypt the bulk of the communications. This reduces the amount of ciphertext encrypted with the master key and the likelihood of it being discovered through cryptanalysis.

To avoid known-plaintext attacks the encryption algorithm must make it difficult to work backwards and derive the key from the ciphertext and plaintext.

The only type of encryption that can be proven to be unbreakable is a "one-time pad." In this case the key is a truly random sequence of bits the same length as the plaintext. The key is then exclusive-or'ed with the plaintext. This is only feasible for low-volume and high-value messages such as master keys.

The most common symmetric encryption techniques are block ciphers such as DES (Data Encryption Standard), adopted in 1997 and AES (Advanced Encryption Standard), adopted in 2001 and currently the most commonly used symmetric encryption algorithm. DES is now considered susceptible obsolete because advances in cryptanalytic techniques and computational hardware have made it practical to break.

AES encrypts data in blocks of 128 bits under control of keys of 128 or 256 bits. The algorithm was designed for efficient implementation in software. It consists of multiple "rounds" that rearrange (permute) and re-map the values of the 16 bytes in each block.

A symmetric encryption algorithm is a function with two inputs (plain text block and key) and one output (ciphertext). The algorithm can be used in different cipher "modes" to make certain types of cryptanalysis more difficult and suit different applications. Common cipher modes include:

electronic codebook mode (ECB) The simplest approach. The key is used to encrypt the data. The

drawback is that there is a 1:1 mapping between plaintext and ciphertext which makes cryptanalysis easier.

cipher block chaining (CBC) To avoid this problem, the plaintext is first xor'ed with the previous ciphertext before being encrypted. An "initialization vector" (IV, transmitted in plain text) is used with the first plaintext block.

stream cipher mode Also known as output feedback (OFB). This technique generates a sequence of bits which are xor'ed with the plaintext. The bits are generated by recursively encrypting the previous block of bits. The process begins with an IV.

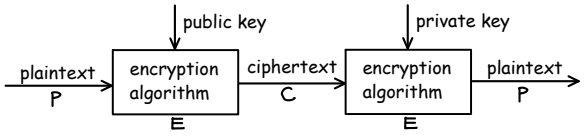
counter mode (CTR) An IV plus a counter (block index) value are encrypted and the result is xor'ed with the data. This allows starting the encryption or decryption at any point in the stream.

Different modes are suitable for different applications. For example, CBC and OFB require decrypting a message sequentially from start to end. A CBC stream cannot be decoded past the first error in the ciphertext.

Public Key Encryption

A problem with symmetric encryption is that the number of keys that need to be exchanged grows as the square of the number of parties that need to communicate with each other. Clearly this is not practical for systems with a large number of users.

Public key encryption is a technique that solves this problem by using different keys for encryption and decryption. One key, the public key, is assumed to be known to everyone. The second key, the private key must be kept secret.



Public key algorithms simplify the problem of key distribution because the public keys can be exchanged between the two parties at the start of a session. Public keys have been widely adopted.

However, public key encryption (PKE) algorithms are much slower. Therefore typically PKE algorithms are only used to transfer a session key which is then used for symmetric encryption of the bulk of the data. Two devices that want to communicate securely first exchange their public keys and then use these public keys to transfer a randomly-chosen session key.

Public key algorithms are designed so that the public and private keys cannot be (easily) derived from each other. Several public key algorithms have been developed, the most popular of which is known as RSA after the initials of the inventors.

Hash Algorithms

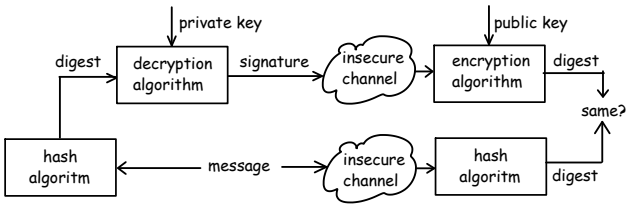
It is often useful to be able to compute a "hash" value that is similar to a CRC in that it is a function of the contents of a message but has the additional property that the function cannot be easily reversed – it should be very difficult to create a document that has the same hash value as another (this is not the case with a CRC). These hashes are called cryptographic hashes or message digests and can be used to verify that the document has not been altered. The advantage of a hash is that it is only necessary to protect the hash value which can be much shorter than the message being protected.

The most common hash algorithm is SHA-1. It produces a 160-bit hash value from a document.

Authentication

Authentication is verification of identity. For example, authenticating the author of software, an e-mail or the company responsible for a web site.

Public key algorithms are used for authentication. Some public key algorithms (such as RSA) have the property that the encryption and decryption can be done in reverse order and still produce plaintext. A message (or, more typically, a cryptographic hash or digest of the message) can be "signed" by first "decrypting" it with the private key. A recipient can use the public key to encrypt the signature and check to see if the value matches the document hash. If so, then the document must have been signed by someone in possession of the (secret) private key corresponding to the public key:



Because of its similarity to a physical signature, a hash value that has been encrypted with the author's private key is called "signature". Signed messages can be authenticated by decrypting the signature with the purported author's public key and comparing the result to the hash value of the signed message.

Exercise 4: Why does the signing key have to be kept private?

Public Key Infrastructure

The use of public keys for authentication introduces another problem: how do we authenticate the public keys themselves?

Authentication of public keys is done with "certificates". A certificate is a document that contains identifying information (e.g. a personal, company or domain name) along with that author's public key. To ensure the authenticity of a certificate, they must be signed by a Certificate Authority (CA).

The certificate authority is a party who is trusted to have verified the identify of the entity named in the certificate before signing the certificate with the CA's private key. A receiver can then check the signature of a certificate to make sure it

A CA also publishes a certificate with its public key so the certificates that it signed can be authenticated. One CA can also sign another CA's certificate. Ultimately this "chain of trust" must lead to a certificate that cannot be authenticated by means of its signature. These certificates are typically supplied by a trusted party such as a software manufacturer who has verified the certificates in some other way. For example, a browser comes configured with several dozen CA certificates, all of which are implicitly trusted to authenticate other certificates.

The system of signing certificates that establish this chain of trust between CAs is called a public key infrastructure (PKI).

X.509 is the ISO-standard certificate format that has been widely adopted. Certificates often include additional information such as an expiry date, the level of identity validation performed by the CA (e.g. control of an e-mail address or government identity documents), and the allowed uses of the certificate (e.g. signing e-mail, e-commerce, etc.).

Exercise 5: What three things must any certificate include?

In some cases a certificate must be revoked. For example, a user's private key may have been disclosed. It is necessary to publicize that the certificate is no longer valid so that those who obtained the private key cannot impersonate the original owner. A CA can publish a certificate revocation list (CRL) with a list of certificates that have been revoked. The CRLs can be downloaded periodically and used to verify certificates before being used for authentication.

It is up to each user to decide which CAs to trust. Web browsers come with a set of several dozen default CAs which software companies (Microsoft, Mozilla, Google, etc) believe use good practices in signing certificates for authentication.