The Microsoft Internet Security Framework

Version 1.0
June 3, 1996

Microsoft Corporation

Printed in the United States of America.

April 18, 2021

Printed on recycled paper.

The Microsoft Internet Security Framework

Overview

This paper is intended for corporate developers and consultants, independent software vendors (ISVs), network operators, and Webmasters who are interested in the convergence of the corporate intranet and the public Internet. It describes the Microsoft Internet Security Framework — a comprehensive set of public-key and password-based security technologies that give you the ability to:

- Exchange information securely across public networks.
- Control access from the public networks to the corporate network.
- Engage in electronic commerce.

The Microsoft Internet Security Framework accomplishes this without requiring you to replace your existing security model. Instead, it builds on the Windows® operating system security model and also provides an extensible architecture that integrates these security technologies into this model.

(Readers who are unfamiliar with public-key cryptography may first want to read the “Core Technology” appendix before continuing with this document.)

Introduction

The security paradigms in the world of the corporate network, or intranet, and the world of the public Internet have followed different paths. This is because of the differences in their computing environments. For example, an intranet typically has:

- A known number of users who are authenticated by the system.
- A trusted administrator who keeps information about the users.
- A central administration model with finely-tuned access controls.
- A set of administrative and user management tools for overseeing the entire network.
- A large investment in the existing security technology.

On the other hand, the Internet:

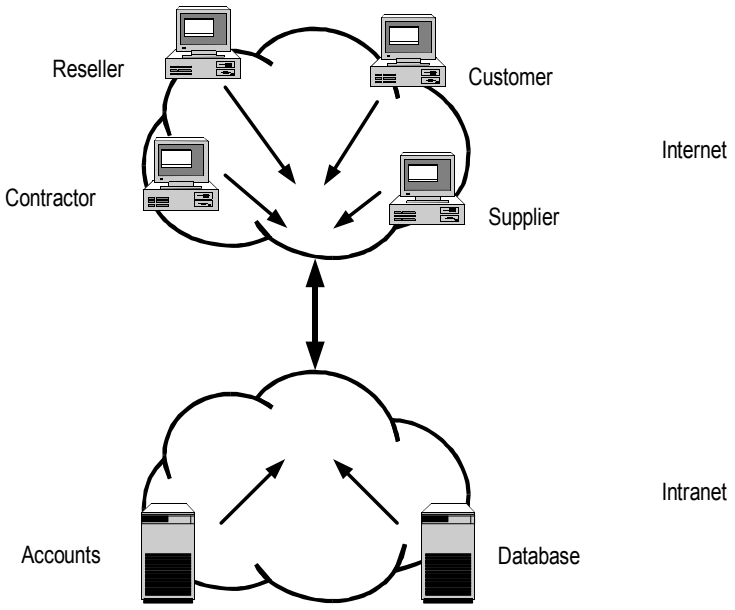
- Is used by a vast number of people who were previously unknown.
- Has no central administration to oversee access and security.
- Is a distributed, cross-platform network.
- Uses new, rapidly evolving technology.

Yet despite these difficulties, the pressures both for gaining access to the Internet, and for allowing access to the corporate networks from the Internet, are great. Many companies are already granting access to their networks despite the security problems. They do this because of the benefits they gain from this sort of cooperation with their customers. These benefits include:

- Increasing sales by creating new sales channels and reaching new customers.

- Forming closer relationships with customers, partners, and suppliers. This results in better products that are brought to market faster.
- Better customer service which means more business from customers.
- Better margins, which result from:
  - Lowering the costs of bringing a product to market.
  - Lowering the cost-per-sale for both repeat and new business.
- Gaining an edge by offering access before competitors do.
- Delivering faster, personalized service.
- Improving distribution.
- Improving the customers' experience.

When the intranet and the Internet converge, they form a single entity which can, for businesses, look something like this:



But currently, this convergence has created a place of confusion. Security tools are immature and complete solutions are difficult to create. This often results in security measures that are a laundry list of

- Firewalls.
- Proxy servers.
- Password-based security.
- Customized access controls.

And even worse, sometimes there is no security at all.

To take advantage of the potential the new world offers, businesses must be able to:

- Open up their corporate networks to the Internet while still maintaining control over who accesses their internal resources.
- Identify and authenticate customers who use the Internet to access their corporate networks. This includes customers using both e-mail and pipes.
- Ensure that private information sent over the public Internet can be transmitted securely.

Common Scenarios

In this section we describe several common situations where businesses would benefit from allowing their intranets to be accessed from the Internet and vice versa. We also list the technologies necessary to do this securely.

Development

A typical scenario is when a company decides to allow contractors and partners access to their intranet for cooperative design and development efforts. Often, this means they download software upgrades from their vendors. To do this securely requires:

- Client authentication, which means the server can identify and authenticate users.
- Certificates, which are needed for client authentication and access control.
- Access control, which determines which parts of the intranet a client can use.
- Code signing, which guarantees a known software publisher and intact code.

Electronic Commerce

Another common scenario is when a company decides to allow customers to set up an account and order parts on-line. To do this securely requires:

- Client authentication to make sure of the client's identity.
- Certificates for client authentication and access control.
- Secure messaging, which means a customer's order forms are confidential.
- Encryption, which is used by secure messaging to keep data confidential.
- Secure payment messages to keep financial information confidential.

Database Access

Another common scenario is when a company decides to allow resellers to access the internal customer information database to streamline the way leads are generated. To do this securely requires:

- Client authentication to make sure of the client's identity.
- Certificates for client authentication and access control.

- Integration with existing information systems.

Internet Service Provider

Another common scenario is when an Internet Service Provider offers a service including Internet access, electronic mail, news services, and access to a package of Web content that can include services on third-party sites. To do this securely and conveniently requires:

- Single logon client authentication both for the dialup network as well as applications such as the Web, e-mail, news and chat.
- Either password and certificate-based authentication with password protection over the Internet and from third-party sites.
- Distributed authorization services.
- Integration with Internet server applications.
- Integration with terminal server hardware residing at the dialup point-of-presence (POP).
- Integration with new and existing security databases.

The Goal

The goal is this: The environment created by the convergence of the public and private networks should be a place where systems can be extended to take advantage of new opportunities while still preserving investments in the existing systems. This environment must behave as an intelligent, secure network for distributing business-to-business applications. It must have:

- Secure exchange of information.
- Secure transactions to conduct electronic commerce.
- A way of controlling access to content.
- A distributed authentication technology based on passwords.

The following table summarizes the tools necessary for achieving this goal.

Goal	Mechanisms
Secure exchange of information	Authenticated connections, privacy (encryption), integrity, authorization.
Access control	Client authentication, certificates issued either by an employee's company or by a CA, password-based authentication.
Secure transactions	All of the above, as well as code signing, a secure place to store private information, a way to transport that information using off-line storage, and payment protocols.

The Microsoft Internet Security Framework Philosophy

The philosophy behind the Microsoft Internet Security Framework is to achieve this goal by using the best of existing technologies as a platform, and to extend them to encompass new technologies. This provides a comprehensive framework for secure on-line communications and electronic commerce. Issues of identity, authentication, and authorization are addressed using public-key and password-based technologies. These technologies can, when appropriate, be integrated with the Windows and Windows NT® operating system. The extensible security framework conforms to and takes advantage of Internet standards and protocols.

We will give an overview of the components that make up this philosophy.

The Foundation

The foundation of the Microsoft Internet Security Framework is the CryptoAPI, which provides APIs for both cryptography and certificate functions. In CryptoAPI 1.0, developers will find APIs for:

- Selecting and using a cryptographic service provider.
- Generating and exchanging keys.
- Data encryption and decryption.
- Hashing, creating and verifying digital signatures.

CryptoAPI 2.0 will add certificate functions and high-level calls. These functions will include:

- Generating requests to create certificates.
- Storing and retrieving certificates.
- Parsing and verifying certificates.
- Coding and decoding certificates for wire formats such as DER.
- Abstracting certificates into cryptographic messages, such as PKCS #7 format.

Higher-Level Security Services

A rich set of higher-level security services and mechanisms rest on top of the CryptoAPI. These include:

- Secure channels for private communications using SSL versions 2.0 and 3.0 and PCT.
- Certificate-based authentication.
- Password-based authentication.
- Code signing for ensuring that downloaded code has not been tampered with and has a known publisher.
- The SET protocol for secure credit card transactions.
- The Certificate Server for issuing and revoking certificates.
- The Microsoft Wallet for storing personal security information such as keys, certificates, and credit card information.
- The Personal Information Exchange (PFX) for moving personal security information between computers and platforms.

- Access Control

- Distributed authentication technology based on passwords, including integration with Internet protocols. This technology allows pass-through authentication with password protection, distributed authorization, and integration with Windows NT security. It support interfaces to scalable databases.

Standards

Microsoft® has made a strong commitment to supporting existing Internet standards such as X.509 and PKCS. Microsoft is actively participating in the Internet Engineering Task Force (IETF), World Wide Web Consortium (W3C), and other groups. Recent examples include the PFX protocol submitted to the W3C Digital Signature Initiative; the code signing proposal submitted to the W3C; and the Transport Layer Security (TLS) efforts through the IETF, aimed at creating a single secure channel standard.

Openness

All Microsoft Internet Security Framework specifications will be published and available for review. Some examples of this are the security design review which is planned for July 1996, as well as Microsoft's work with SET and JEPI.

Interoperability

The Microsoft Internet Security Framework is interoperable and designed to work with existing security models. You do not have to replace your existing systems to take advantage of its features. Here are some examples:

- The Internet Explorer 3.0 browser, using SSL, will communicate with a NetScape server that runs SSL.
- Microsoft is working with NCompass to create a code signing plug-in for NetScape Navigator.
- Windows NT servers can perform client authentication with NetScape browsers that use VeriSign certificates.
- Microsoft Certificate Server can issue certificates to other clients.
- Microsoft Internet Explorer can work with third-party certificate servers to obtain certificates for client authentication and code signing.

Cross-Platform

Microsoft's implementations of the Wallet, client authentication, distributed authentication, secure channel protocols, CryptoAPI, code signing and SET will all be made available, via the Internet Explorer, on Windows NT, Windows 95, 16-bit versions of Windows, Macintosh, and UNIX platforms. Also, Microsoft is working with Metroworks to develop code signing for the Mac.

Integration

The Microsoft Internet Security Framework integrates Internet technologies with your existing security models. This means you can unite the Internet and Intranet by leveraging your investment in existing tools and systems.

Developers can use their existing knowledge of Windows NT programming to create new applications. In particular, they will find that the CryptoAPI layer frees them from understanding the complexities of the underlying cryptography. Also, because of standard interfaces, developers are free to use whatever third-party services they choose. For example, developers can either use distributed password-based authentication or certificate-based authentication. Developers can also select among a variety of both cryptographic and certificate service providers simply by changing a few parameters in an API call. Finally, there is the assurance that applications using the CryptoAPI can be exported because the approved cryptographic service provider is included.

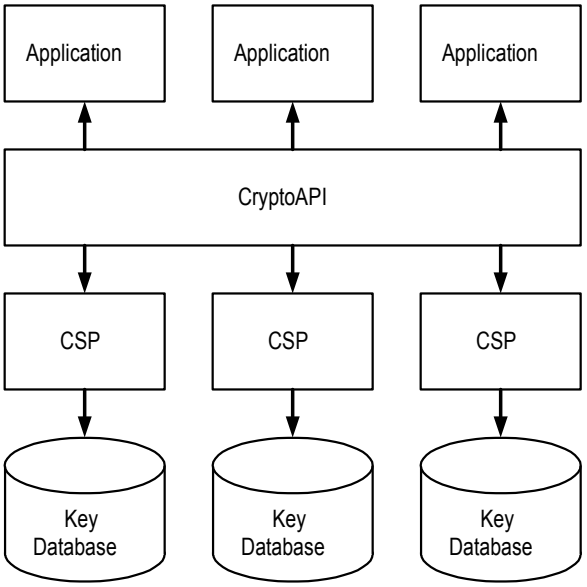
Webmasters will find that they can extend the familiar Windows NT security methods to embrace Internet security methods. Certificates, which are one of the major methods for authentication on the Internet, map to Windows NT users and user groups. The same methods and tools used for access control within the intranet can be used when granting access to a user on the Internet who wants to use the corporate network. This greatly simplifies administration and also means that only one logon is necessary when crossing between the intranet and Internet.

When the CryptoAPI is integrated with other products and higher-level security services such as Wallet and secure channel protocols, it is possible to provide authorizations and secure transactions.

CryptoAPI

Cryptography is the basis for enabling secure communications. The Microsoft Internet Security Framework provides a single interface to the unstandardized world of cryptography. Yet it is also open, which means the developer can use whatever cryptographic algorithm, strength of encryption, or certificate format that is required. This interface is called the CryptoAPI. As well as making it available for Windows, Microsoft intends to make it available on the Mac and UNIX.

Here is a diagram of the architecture.



Cryptographic Service Providers (CSP)

CSPs can either be software modules such as the RSA implementation supplied with Microsoft products or actual hardware devices such as the BBN SafeKeyper. Developers can have complete access to the services a CSP provides without having to learn how the cryptography is done. All keys are handled by a CSP, which is responsible for creating and storing keys, destroying them, and using them to perform a variety of cryptographic functions.

CryptoAPI, version 1.0

Version 1.0 of the CryptoAPI, which is incorporated into Windows NT 4.0, Microsoft Internet Explorer 3.0, and the Windows 95 OEM refresh, includes:

- Context functions.
- Key storage and exchange functions.
- Data encryption functions.
- Hashing, digital signatures, and signature verification functions.

We will discuss each of these functions briefly.

Context Functions

These functions are used by applications to connect to a CSP and establish a cryptography context.

Key Generation Functions

These functions allow applications to generate and customize cryptographic keys. Full support is included for changing chaining modes, initialization vectors, and other encryption features.

Key Exchange Functions

These functions allow applications to exchange or transmit keys. Also included are functions for storing session keys. You may want to store a key, for example, when you have encrypted a file using a key and want to decrypt the file at a later time. You exchange keys when you need to send a key to someone else. The key has to be exported from a CSP, transmitted by an application to the destination application, and then imported into the destination CSP. The CryptoAPI provides functions to easily perform all these tasks.

Data Encryption Functions

These functions allow applications to encrypt and decrypt data. Encryption is the process where data (called plaintext) is translated into something that appears to be random and meaningless (called ciphertext). Decryption is the process that takes ciphertext and converts it back to plaintext.

Hashing, Digital Signatures, and Signature Verification Functions

These functions allow you to create and verify digital signatures. Digital signatures enable a user to easily determine who sent the data and that the data hasn't been changed. To create a digital signature, you first create a hash value from the message. Hashing takes a large amount of data and transforms it into a very small amount of data. This hash value is then encrypted by a public key encryption algorithm, using the private key of the signer.

To verify a signature, both the original message and the signature are required. First, a hash value must be created from the message, in the same way as when the signature was created. Another hash value is generated by decrypting the signature using the public key of the signer. If both hash values match, you can be confident that the message is indeed the one the signer originally signed and that it has not been tampered with.

Version 2.0 of CryptoAPI

Version 2.0 of CryptoAPI will add certificate functions as well as simple, easy-to-use APIs to the cryptographic functions we described in the previous section. A beta version of CryptoAPI 2.0 is scheduled to be available in July 1996.

Both public-key cryptography and digital signatures rely on the use and availability of keys. This means that another possible security breach can be caused by an intruder who provides bogus public-key information. This is resolved by certificates.

A certificate is trustworthy data that contains the user's name and the user's public key. This data is considered trustworthy because it is signed by a trustworthy person or organization, known as a *certification authority* (CA). Certificates are verified through a hierarchy of these authorities. Each certificate is linked to the certificate of the authority that signed it. By following this hierarchy, or *verification path*, to a known, trusted authority, you can be assured that a certificate is valid.

Certificate Functions

These are the certificate functions that will be available in version 2.0 of the CryptoAPI:

- Generating requests to create certificates. (PKCS #10 and others)
- Storing and retrieving certificates.
- Parsing and verifying certificates.
- Coding and decoding certificates for wire formats such as DER.
- Abstracting certificates into cryptographic messages, such as the PKCS #7 format.

Just as CryptoAPI 1.0 provides extensibility by using modular CSPs, CryptoAPI 2.0 will do the same for certificate formats. The framework supports different types of certificates and formats, at different levels. The initial functionality will include certificates using X.509 version 3, ASN.1, and DER. A set of certificates using X.509, version 3, ASN.1, and DER, with certificates stored on a remote server, are supported with the same APIs as PGP certificates, which are stored on a PCMCIA token.

Easy-to-Use APIs

The easy-to-use APIs consolidate a number of the highly granular CryptoAPIs into single calls with only a few parameters. These calls allow developers to sign and encrypt data, as well as decrypt it and verify signatures.

Building on the Foundation

Resting on top of the CryptoAPI is a rich set of higher-level services for secure communication, secure transactions, and digital integrity. Also, because securing, transporting and displaying personal security information require some additional facilities, the Microsoft Internet Security Framework also provides a Wallet, and a Personal Information Exchange protocol.

We will first discuss how these particular features improve the user's experience. We will then discuss the other higher-level security services.

The User Experience

For users, the Microsoft Internet Security Framework makes the convergence of the intranet and Internet transparent. Users don't want to maintain different passwords and credentials, deal with unfamiliar software, or worry about whether systems are compatible. The Microsoft Internet Security Framework has:

- A single logon. Users only need one set of credentials (for example, a password or PIN number) to gain access to all keys and certificates. To the user, the intranet and Internet look like a single network.

- The Wallet, whose contents can be accessed by different browsers and applications in a controlled and protected manner. To the user, multiple systems look like a single system.

- Personal Information Exchange. This protocol allows users to transport information securely from one type of computer or medium to another. To the user, different computers behave like the same computer.

We will now discuss each of these features in more detail.

Logging On

The most important point about logging on is that logging on to the system enables access to both the intranet and the Internet. There will be no need for separate logons to separate services. Certificates and keys will automatically be accessed as they are needed. This means that a single logon will cover multiple credentials.

Storing Personal Security Information

It is extremely important that users feel comfortable storing personal security information such as credit card numbers and private keys, on their computers. It is also important that this information be available to different programs (subject to the user's approval) such as browsers and e-mail applications. To satisfy these requirements, the Microsoft Internet Security Framework provides the Wallet.

The Wallet

The Wallet resides on the user's computer (or another hardware device such as a smartcard). It securely stores personal security information such as private keys, credit card numbers, and certificates. Like all Microsoft Security Framework services, it has standard programming interfaces that make it accessible to other programs in a protected and controlled fashion. The contents of the Wallet can be transported across machines, platforms, and browsers using the Personal

Information Exchange (PFX) protocol, which we will discuss next. An access control policy determines who has access to the information. A simple certificate store is available in Internet Explorer 3.0 to support client authentication. The Wallet will initially ship in a release of Internet Explorer in 1996 (all platforms) and will eventually be integrated into the Windows operating system.

Personal Information Exchange

As a part of our comprehensive support for Internet standards, Microsoft has submitted the Personal Information Exchange (PFX) protocol to the W3C. This protocol enables users to transfer sensitive information from one environment or platform to another. For example, a user may have information such as certificates and keys stored on a PC in her office, but she also needs to securely transfer this information to her Macintosh at home.

To solve this problem, Microsoft has proposed the PFX protocol to the W3C. With this protocol, a user can export personal security information from one computer platform to another using some form of off-line storage such as a floppy disk.

Higher-Level Security Services

We will now discuss the services that provide secure communications, secure transactions, and digital integrity.

Secure Channel Services

A secure channel service provides privacy, integrity, and authentication in a point-to-point connection. An example of this is the connection between a Web browser, such as Microsoft Internet Explorer, and a Web server. An example of a secure channel is the Secure Sockets Layer (SSL).

SSL provides a security handshake that is used to initiate the TCP/IP connection. The client and server agree on the level of security they will use, and fulfill any authentication requirements. SSL also encrypts and decrypts the byte stream of the application protocol such as HTTP. This means that all the information, both in the HTTP request and response, are fully encrypted.

The Microsoft Internet Security Framework includes support for SSL versions 2.0 and 3.0, Personal Communications Technology (PCT) version 1.0, and will include support for the Transport Layer Security Protocol (TLS), which is being considered by IETF. This protocol will provide a single standard encompassing both SSL and PCT.

Client Authentication

Client authentication means that a server can identify and authenticate users. The Microsoft Internet Security Framework supports authenticating clients in a secure channel session, via public-key certificates, and also integrate them into the Windows NT security model. This means that access to information can be controlled by mapping certificate users to a Windows NT – based group or user account, and by assigning access control permissions to this group or account using familiar Windows NT administration tools such as File Manager and User Manager. In other words, network administrators can use the tools they know to control access from outside the corporate network. Their existing knowledge allows them to extend the capabilities of their network.

Client authentication using a secure channel and certificates requires the following:

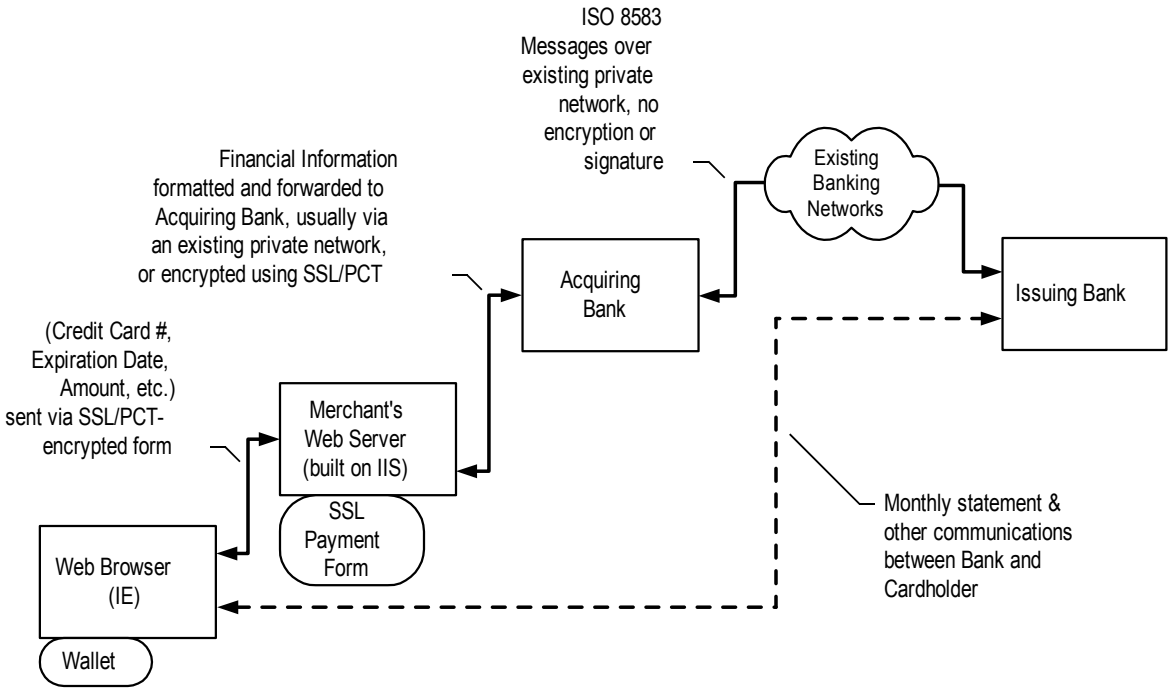
- The protocol must be able to handle certificates at both the client and server end. This includes handling the appropriate requests and replies.
- The client must be able to verify server certificates, request a certificate, and allow the user to present this certificate to the server when so requested. This means the client must support certificate storage and management.
- The server must be able to request a certificate, be able to verify client certificates, and map certificates received from the client to access controls on the server.

Client authentication will be integrated with Microsoft Internet Explorer 3.0. Users will be able to obtain certificates from different certificate authorities. Microsoft will ship an add-on module for Internet Information Server 2.0 that will perform authentication via certificates.

Payment Protocols

Merchants must be able to automatically collect and process payment information over the Internet. Microsoft is working with other parties and standards bodies to develop several ways of doing this. For example, Microsoft is working with IBM, Netscape, GTE, Visa, and MasterCard to develop the Secure Electronic Transaction (SET) protocol. The SET specification is expected to be completed in June 1996.

Until SET is available, Microsoft will provide tools for merchants to accept credit card transactions over secure channels such as SSL and PCT. The following diagram illustrates payment over a secure channel.



This method of payment is an interim solution until SET is finalized. It will work with any browser that understands SSL and PCT. However, because secure channels were not specifically designed for secure financial transactions, there are two limitations:

- There is no authentication. In other words, there is no way of proving that the person using the credit card is the person who should be using the credit card.

- There is some privacy, because the data is encrypted, but there is no way to control who has access to the information. For example, the merchant has access to all the credit card information. Ideally, only the bank needs to know the card number. The merchant needs to know only that you have enough credit to pay the bill.

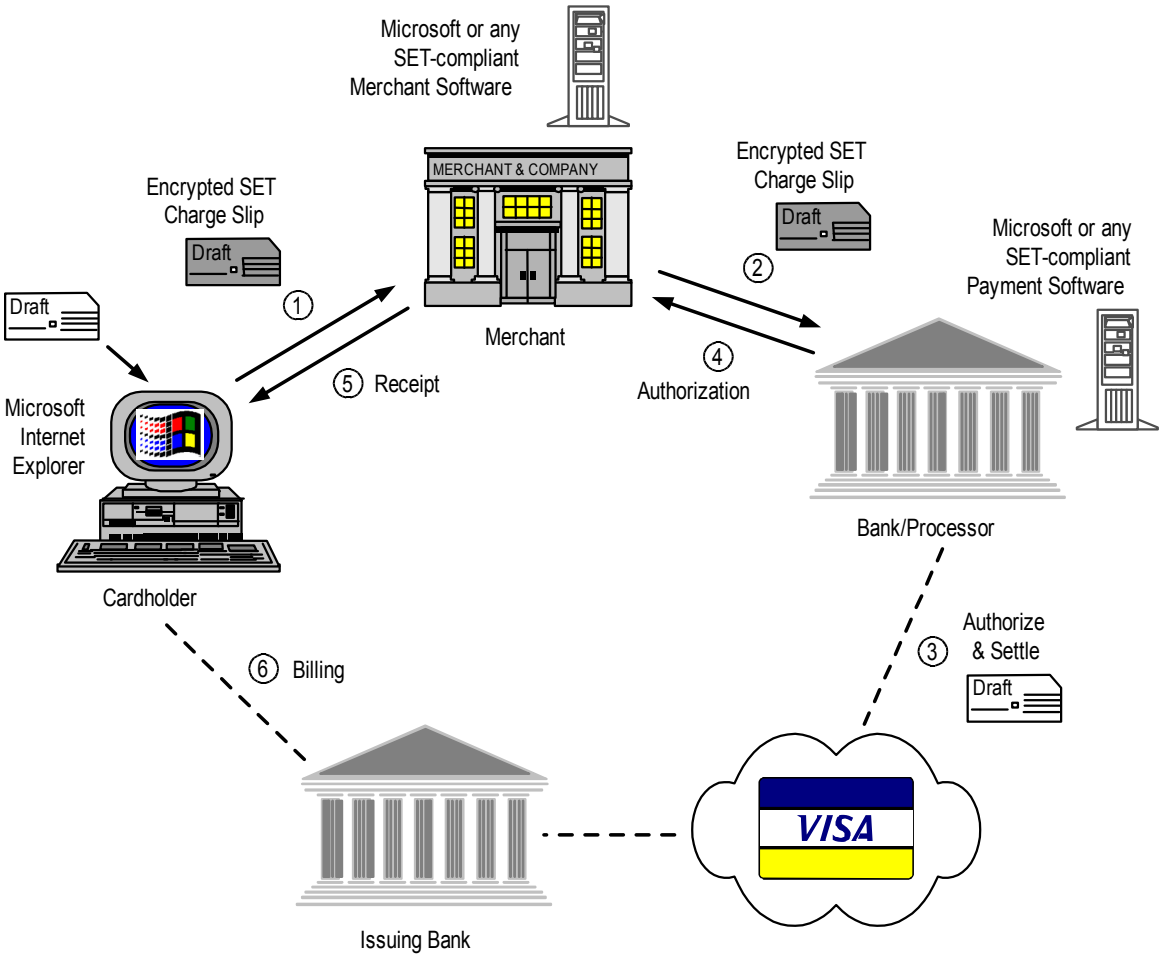
SET

SET is a secure message protocol for credit card transactions. It provides authentication for cardholders, merchants, and acquirers. SET preserves the confidentiality of payment data. It does not encrypt such information as order descriptions.

SET differs from payment over secure channels such as SSL in the following ways:

- SET uses 56-bit DES encryption.
- SET requires digital signatures to verify that the customer, the merchant, and the bank are legitimate.
- SET uses multi-party messages that allow information to be encrypted directly to banks. This avoids credit card numbers ending up in the wrong hands.
- SET requires integration into the credit card processing system.

The following diagram illustrates the way SET works. (In this example, any SET-compliant software will work.)



Once SET is available, Microsoft will deliver tools to aid merchants, acquirers, and payment processors to create SET-compliant applications.

JEPI

There will be a negotiation layer on top of SET and other payment protocols. The definition of this layer is the task of the Joint Electronic Payment Initiative project (JEPI), of which Microsoft is a member.

The JEPI project explores the technology required to provide a negotiation layer over multiple payment methods, protocols, and transports. Examples of payment methods include credit cards, debit cards, electronic cash, and checks. Payment protocols (for example, SET) define the message format and sequence required to complete the payment transaction. Payment transports are the mechanisms for message transmission. These include, for example, TCP/IP, SSL, and S-HTTP.

Code Signing

The Microsoft Internet Security Framework includes tools for signing code that are already available to developers. The ability to check for signed code is integrated into Microsoft Internet Explorer 3.0 beta. The Microsoft Internet Security Framework code-signing tools and APIs are, of course, open, which means other browsers can use them as well. This code signing strategy is meant to solve one of the larger questions facing the software industry today: How can users trust code that is published on the Internet?

Packaged software uses branding and shrink-wrapping to assure users of its integrity. The logo implies trust and reputation, while the shrink-wrap means no one has tampered with the contents since the box was sealed. However, code that has been transmitted on the Internet doesn't include these familiar assurances. To provide them, software publishers need a digital equivalent. Code signing provides this. It is the Internet version of branding and shrink-wrapping.

Code signing:

- Ensures authenticity, which means users know who published the code.
- Ensures integrity, which means users know the code hasn't been tampered with since it was published.
- Makes both corporations and individuals feel comfortable purchasing and installing code over the Internet.
- Is a W3C initiative supported by over 50 companies.

Signing Code

To sign their code, publishers first receive a certificate from a trusted third-party called a certifying authority. Then, when they are ready to ship their code, publishers sign it by encrypting their certificates into the code with their private keys. In other words, a digital signature is analogous to a person's signature on a legal document. A certificate is analogous to a notary seal on a document.

Downloading Code

When a user downloads the code, the browser or client-side application calls a function to verify signatures. This function uses the publisher's public key (verified by the certifying authority) to decrypt and check the signature. After the signature is checked, the following may happen, depending on the options selected by the user:

- If the code has not been signed, a warning (such as the one commonly seen today) is displayed and the user can decide whether or not to install the code.
- If the signature is invalid, or if the certificate has been revoked or has expired, then a strong warning appears, warning the user.
- If the certificate is valid, the user can decide to receive information about each piece of downloaded code, or simply let all signed code execute without first displaying any messages. The user can even decide to let a certain type of code run (based, for example, on the publisher or certifying authority), and flag any other type of code.

Certificate Server

The Certificate Server is designed to make certificate management tasks as simple as possible by issuing, managing, and revoking certificates. Because the Microsoft Internet Security Framework is an integral part of the Windows security model, certificates can be mapped to a Windows NT group. This means you can use the standard Windows NT administrative tools, such as access control lists, to handle Internet security matters. The Certificate Server will also be integrated with BackOffice.

The Certificate Server is integrated with an ODBC-compliant database such as SQL Server™ that holds such information as certificate revocation lists (CRLs), auditing information about certificates that were issued, and copies of certificates. Here are some key features of the Certificate Server.

Key Management

The most vulnerable point of any certification system is how private keys are protected. Because key management falls under the aegis of CryptoAPIs, the Certificate Server is isolated from these most confidential pieces of data. Also, as discussed in an earlier section, you are free to use anything from software modules to industrial-grade key engines to protect your keys.

Policy Independence

Certificates are granted according to some policy that defines what criteria must be met to receive a certificate. For example, one policy may be to grant commercial certificates only if applicants present their identification in person. Another policy may grant credentials based on e-mail requests. An agency that grants credit cards may first want to consult a database and make phone inquiries before issuing a card. The Certificate Server functions are isolated from any changes in policy an agency might make. Changes in policy do not mean changes in a server's code.

Transport Independence

Certificates can be requested and distributed via any transport mechanism. For example, the Certificate Server can post certificates to the applicant via HTTP, e-mail, Microsoft Exchange Address Book, or by some other custom transport.

High Reliability

The Certificate Server has extensive fail-safe mechanisms built in and leverages the reliability features incorporated into Windows NT Server.

Adherence to Standards

The Certificate Server accepts standard PKCS #10 requests and issues X.509 version 3 certificates. It will work with non-Microsoft clients and browsers. In addition, the Certificate Server will support different certificate formats.

SSPI

Extensibility is key to providing the best of both the Internet and the intranet. The Security Support Provider Interface (SSPI) provides this. It allows any application that uses SSPI calls to connect to any of the security modules available on a computer or network. Security providers or protocols grant applications (for example, Microsoft Internet Explorer or SQL Server) an authenticated connection. Authenticated connections prevent an impostor from posing as the real server.

For example, by using the SSPI calls, developers can make their browsers work with any of the available security modules on the server. Also, instead of needing to know all the details about how the module works, the browser only needs to make a single call, with a few parameters.

On the server side, there are also many benefits. For example, every application displays a well-defined, uniform behavior. Also, even if developers make extensive changes in the security modules code, they only need to make minor changes to the SSPI calls the module uses.

The calls defined by the SSPI fall into 4 major categories:

- Context management
- Credential management
- Message support
- Package management

Here is a general description of how an application uses the SSPI.

1. Using the package management APIs, the application lists the available service providers and selects the appropriate one.
2. Using the credentials APIs, the application acquires a handle to the user's credentials.
3. Using the context APIs, the application defines the level of security it requires.
4. Using the message APIs, it encrypts the data.

Distributed Security Based on Passwords.

Although public key security is gaining momentum, numerous Microsoft customers, (particularly Internet service and content providers) have stated the need for distributed password-based authentication that provides a single, secure user logon for both dialup connectivity and applications across multiple servers and sites. Microsoft is, therefore, providing password-based security technology for both network-level and application-level security that integrates with Internet protocols and Windows security for a common user and administrator experience.

This security technology provides distributed, scalable authentication, authorization, and billing services to Internet servers (and their respective clients) which are directly accessible to end users of Internet services. A distributed password-based authentication architecture builds on the classic three-entity distributed model of client, server and provider. Applications running on the server do not have to provide their own authentication, authorization, or billing

functions; nor are they limited to whatever services are offered by the local or network operating system. Services can be obtained from trusted third-party providers located anywhere on the Internet. The distributed password-based authentication optimizes its network communications for a typical Internet scenario (This is a small bandwidth connection between the client and an application server, but a large pipe between the server and providers.). All of the distributed, password-based authentication systems traffic occurs between the client and application server, or between the server and the provider.

Application Level Security

For application level security, this technology leverages the existing Windows NT security mechanisms and administrative tools so that ISVs do not have to build all new mechanisms into their applications, and content providers do not have to learn new tools and procedures to manage the security of their sites. It also provides a modular, extensible framework that could deliver a turnkey solution with existing services, yet also supports coexistence with and migration to public key technology. This means there is no need to rewrite all the client and server applications.

This interoperability is achieved through adherence to the published Security Support Provider Interface (SSPI), a standardized interface for client and server applications to obtain services from multiple providers. Under SSPI, security messages encapsulate the methods required to negotiate a given level of service, locate a service provider, and perform service-specific operations.

The distributed authentication service uses a derivation of the Windows NT authentication protocol which employs passwords and a trusted third-party security server. It never transmits clear text passwords to prevent them from being sniffed off the wire or exposed to application servers. The security server offers a published API which allows it to interface to potentially any Internet services' security database. Also, the security server provides distributed authorization services for controlling user access to content. User permissions- maintained in the account database of the online service(s) to which they subscribe- can be validated by application servers anywhere on the Internet against NT ACLs -maintained locally on the server. Authorization tokens provide a means to link authorized users to protected content, and thus allow service providers and content providers to be able to perform their required security administration independently of one another.

Network Level Security

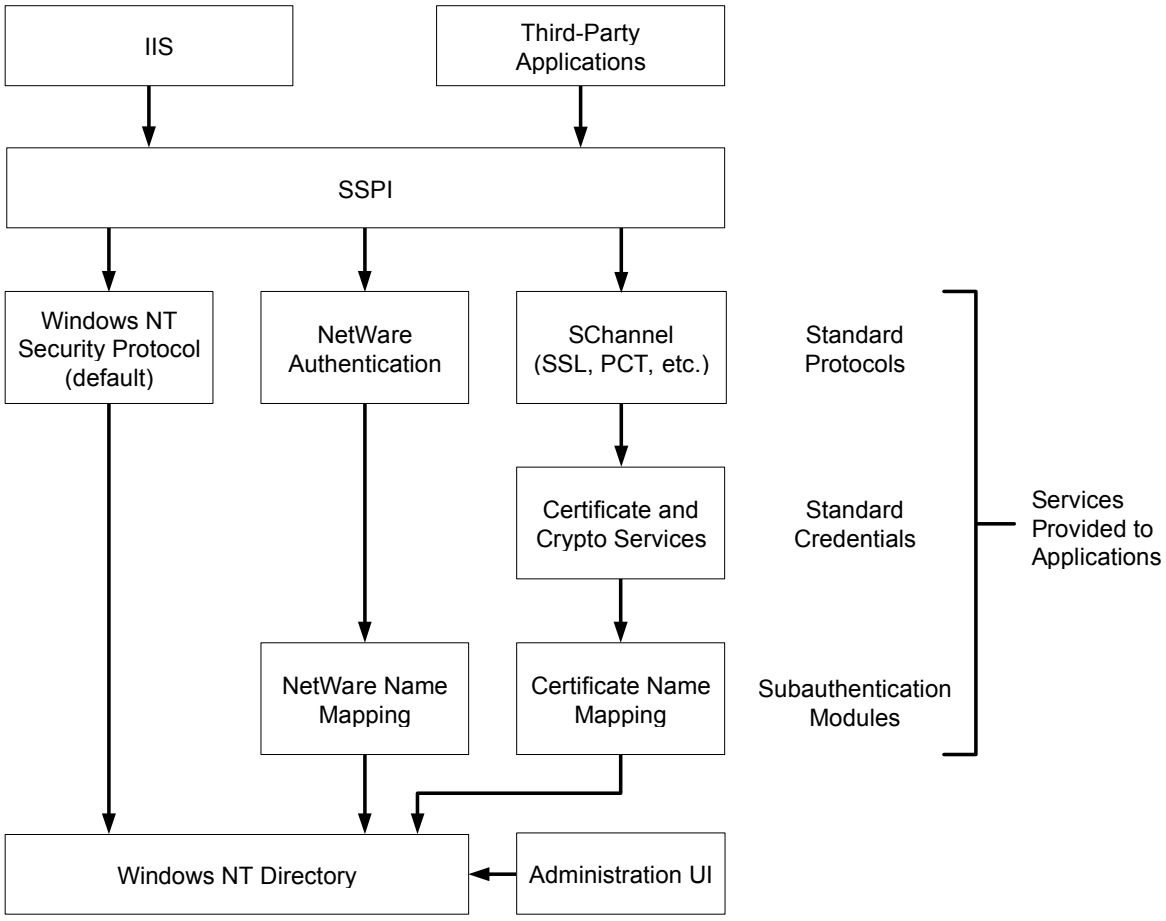
For network level security, this technology supports the standard CHAP/PAP protocols between users and terminal servers on dialup networks and the standard RADIUS protocol from the terminal server at a point of presence (POP) to the dialup provider’s security database. This technology includes a RADIUS Proxy which takes authentication requests from the terminal server and a RADIUS Server which authenticates users against a security database for proxies. This leverages the published security database APIs discussed above. The single logon experience is created for end users through a common password cache across the network and application logons.

Logon and Password UI

The Windows NT and Windows 95 operating systems accomodate methods of logging on that differ from the standard name and password based verification commonly used. Replaceable dynamic-link libraries (DLLs) can provide functionality which allows both alternative user interface, such as use of smartcards for login or more specific "branded" login prompts, and support for additional authentication protocols including a variety of both private key and password-based mechanisms. Windows provides a consistent user interface and single login process across all the login and authentication mechanisms configured for a system. The programming interfaces provided include the Graphical Identification and Authentication (GINA) interface, the Network Provider interface of the Win32 API, and specific APIs on Windows 95 for storing and changing passwords.

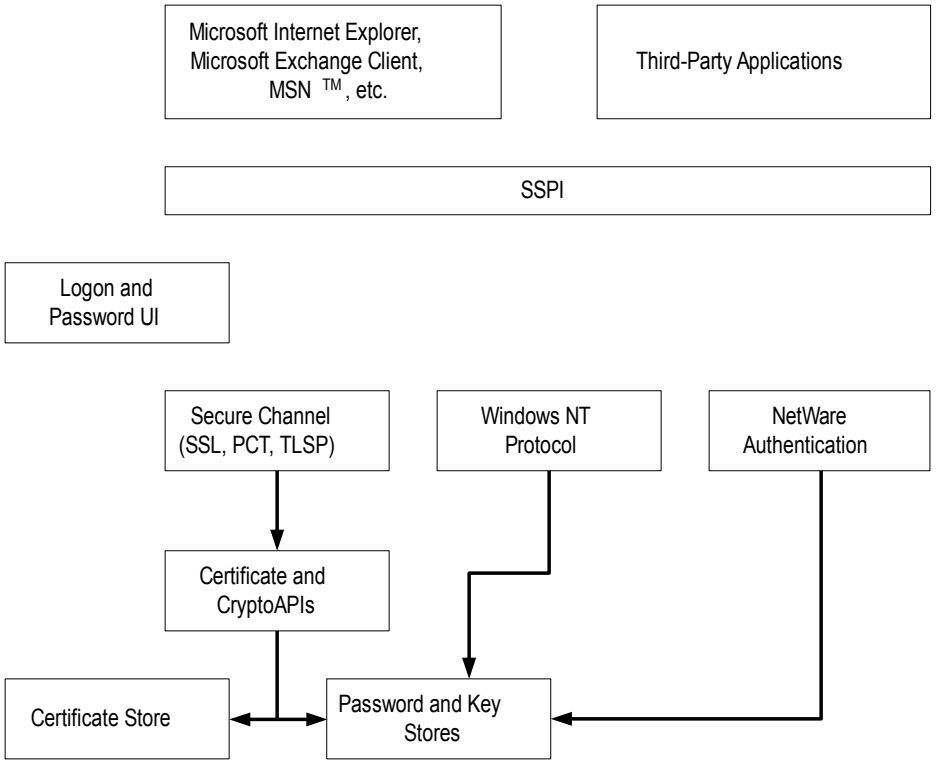
Building on Existing Systems

This section illustrates how the extensible architecture of Windows NT, along with the new public-key based technology, makes it possible for the Microsoft Internet Security Framework to integrate internal network security methods with Internet security methods.



This diagram shows how applications using Win32's SSPI interface support multiple authentication protocols. The picture above shows the existing Windows NT Security and NetWare authentication packages supported in Windows 95 and Windows NT today, and the new certificate-based authentication coming in the third quarter of 1996. Additional authentication packages can also be added under the SSPI interface, such as the scalable authentication used in Microsoft's "Normandy" components. The right-hand side illustrates how the certificate-based services integrate into the Windows NT Server. Essentially, it maps a certificate type into a Windows NT group. The certificate-based provider for SSPI includes a standard protocol such as SSL or PCT, support services for certificates and cryptography, and a certificate name mapping module. This module maps a certificate (initially by examining the certificate type and certifying authority) to a Windows NT group or to a specific user. The administrator can then assign access privileges to this user or group.

The next diagram illustrates the client-side architecture.



The client side is similar to the server side. The protocol and certificate services are integrated into Windows, beneath SSPI. Note the addition of the certificate store, as well as password and key stores. In CryptoAPI 2.0, extensible storage, for example, on a hard drive or a smart card, will be available.

Product Development

This example shows how the Microsoft Internet Security Framework helps businesses create better products faster. It also shows how a network administrator, using familiar tools and procedures, can extend the capabilities of a corporate network.

The Astro Mountain Bike Company

The Astro Mountain Bike Company in Seattle has decided to share its design diagrams with the engineers at Trey Research in Atlanta. Astro Mountain wants to work with Trey Research on several new projects. Giving them access to the corporate network is the best way to make sure the Trey Research engineers can get the information they need whenever they want it. Here is how it happens.

At Astro Mountain

Astro Mountain's network administrator:

1. Establishes a trust relationship with Trey Research. Together they determine the requirements necessary to qualify for a certificate.

2. Sets policy to make sure that certificates are current. This means there must be a certificate revocation list.

3. Creates an NT group called "Trey Research Engineers."

4. Maps everyone from Trey Research who has a certificate into this group.

5. Decides how much access to give each folder and document on Astro Mountain's network.

6. Gives the AT&D group the appropriate permissions.

At Trey Research

Trey Research must:

1. Certify its engineers.

2. Meet Astro Mountain's policy about certificate revocation lists.

Connecting Astro Mountain and Trey Research

When Trey Research engineers connect to Astro Mountain, they are:

1. Authenticated by certificates.

2. Given the appropriate level of access.

Delivering the Microsoft Internet Security Framework

The initial deliveries will be in the Internet Explorer and the Internet Information Server. Over time, technology in the Microsoft Internet Security Framework will be absorbed into the Windows 95 and Windows NT architecture. Here is a breakdown of Microsoft's plans for delivery.

Client Side

On the client side, the Microsoft Internet Security Framework will first be included with Internet Explorer, and then with Windows 95 and Windows NT. Internet Explorer 3.0 will include:

- Code signing.
- CryptoAPI 1.0.
- Secure channels (SSL version 2 and version 3, PCT)
- Client authentication.

Server Side

On the server side, the Microsoft Internet Security Framework will first be included with the Internet Information Server and later with Windows NT Server

and BackOffice. When the final version of Internet Information Server ships, the Microsoft Internet Security Framework will ship with a module for performing authentication via certificates.

Tools

Development kits for the services and mechanisms in the Microsoft Internet Security Framework will be available through the ActiveX SDK as well as through the Microsoft Developer Network subscription program.. Developers can already get CryptoAPI 1.0 and tools for code signing and secure channels.

Conclusion

Today, businesses are opening up their corporate networks, or intranets, to customers who will gain access by using the public networks, or the Internet. This means there is a strong demand for a set of security tools that will allow:

- The secure exchange of information across the Internet.
- Authorized access from the public networks to the corporate network..
- Businesses to conduct electronic commerce.

The Microsoft Internet Security Framework provides these tools. Its foundation is a set of APIs for cryptography and certificate management. These establish identity and provide authentication, data privacy, and data integrity. On top of these is a rich set of services and mechanisms to perform authorizations and transactions.

Microsoft Internet Security Framework is a comprehensive Security Framework that is open, interoperable, and integrated with the Windows security model. Incorporating Microsoft Internet Security Framework into your network means that you accomplish your goals by extending your existing systems, not replacing them.

Suggested Reading

CCITT, Recommendation X.509, *The Directory-Authentication Framework*, Consultation Committee, International Telephone and Telegraph, International Telecommunications Union, Geneva, 1989.

Microsoft Application Programmer's Guide (Microsoft CryptoAPI), Microsoft, 1995

RSA Laboratories, *PKCS #7: Cryptographic Message Syntax Standard*. Version 1.5, November, 1993.

Schneier, Bruce. *Applied Cryptography*, 2d ed. New York: John Wiley & Sons, 1996.

For information about Microsoft Internet Security Framework and other Microsoft technologies: http://www.microsoft.com/devonly/

For information about Microsoft's CryptoAPI:
http://www.microsoft.com/intdev/security/cryptapi.htm

For information about PFX:
http://www.microsoft.com/intdev/security/brink009.htm

For information about RSA: http://www.rsa.com

For information about Cylink another company which provides security technology: http://www.cylink.com

The Core Technology: Public-Key Cryptography

Appendix A

Introduction

Cryptography provides a set of techniques for encoding data and messages so that they can be stored and transmitted securely. This document introduces the basic terminology of cryptography and explains some of the most commonly used cryptographic methods. Here are some ways cryptography is used:

- To achieve secure communications, even when the transmission media (for example, the Internet) is untrustworthy.
- To encrypt sensitive files so that an intruder cannot understand them.
- To ensure data integrity as well as secrecy.
- To verify the origin of data and messages.

Cryptographic Methods

Cryptographic methods are a series of operations performed on data. The two fundamental ones are encryption (with decryption as its inverse) and signing (with verification of a signature as its matching operation). Encryption is analogous to enclosing data in an opaque envelope. Decryption is analogous to removing it from the envelope. Signing is similar to physically signing a document and initialing each section to show that no portion of the document has changed. Verification is roughly equivalent to matching the signature to a "signature on file" card, and verifying that no portion of the document has changed.

Odd as it seems, the only part of a cryptographic method that must truly be kept secret are the private keys. The algorithms, the key sizes, and file formats can be made public without compromising security.

This section will provide a brief description of:

- Functions.
- Encryption and decryption.
- Signature and verification of signatures.
- Certificates, certificate authorities (CAs) and certificate revocation lists (CRLs).
- Protocols.
- Measures of strength

Public-Key Cryptography and Functions

Public-key cryptography (as opposed to symmetric key cryptography) relies on one-way functions. These are functions which are easy to calculate but difficult to invert or reverse without some sort of prior knowledge. One example of a one-way function is factorization. It's often difficult to factor large numbers, but it's easy to verify a factorization. For example, it's harder to factor 4,399 than to verify that 53*83 = 4,399. Public-key cryptography exploits this asymmetry to create functions where:

- It is easy to perform one operation (for example, encryption or verification of a signature).
- It is *extremely* difficult to invert the operation (decryption or creation of a signature) without already having all the information.

Cryptography implements one-way functions by using two related, but different keys called a *key pair*. These keys are created at the same time. They are mathematically related in that the private key is required to invert operations performed with the public key, and the public key is required to invert operations performed with the private key.

A many-to-one function is when the public key is widely distributed and the private key is kept private. This means anyone can use the public key to perform

cryptographic operations, but only the person holding the private key can invert them. This is also a one-to-many function. It means the one person holding the private key can perform an operation that anyone holding the public key can invert. These two functions are used for encryption (many people can encrypt, only one person can decrypt), and signing (only one person can sign but many people can verify a signature).

Symmetric Algorithms

Symmetric algorithms are the most common type of encryption algorithm. They are called symmetric because the same key is used for both encryption and decryption. Unlike the keys used with public-key algorithms, symmetric keys are frequently changed. For this reason, they are called *session keys*.

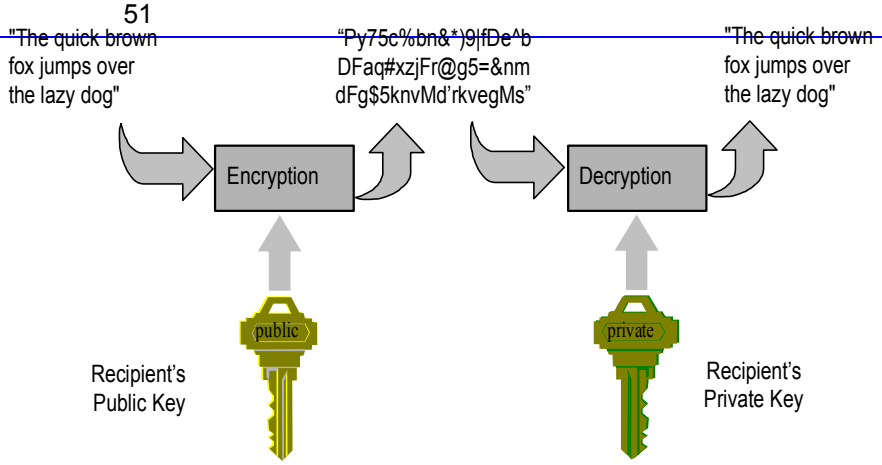
Compared to public-key algorithms, symmetric algorithms are very fast. Consequently, they are preferred when encrypting large amounts of data. Some of the more common symmetric algorithms are RC4, and the Data Encryption Standard (DES).

Many modern cryptographic protocols use a combination of public-key cryptography and symmetric cryptography to obtain the benefits of both. Public-key algorithms are used to exchange a symmetric key, which is then used to quickly encrypt or decrypt data.

Encryption and Decryption

With data encryption, a *plaintext* message can be encoded to look like random gibberish. It is very difficult to transform this encoded data back to the original message without a secret key. In this document, the term *message* refers to any piece of data. This message can consist of ASCII text, a database file, or any data you want to store or transmit securely. *Plaintext* is used to refer to data that has not been encrypted, while *ciphertext* refers to data that has.

Once a message has been encrypted, it can be stored on nonsecure media or transmitted on a nonsecure network and still remain secret. Later, the message can be decrypted into its original form. This process is shown in the following illustration.



Encryption and Decryption Using Public-Key Cryptography

When a message is encrypted, an *encryption key* is used. This is analogous to the physical key that is used to lock a padlock. To decrypt the message, the corresponding *decryption key* must be used. It is very important to properly restrict access to the decryption key, because anyone who possesses it will be able to decrypt all messages that were encrypted with the matching encryption key.

Surprising as it seems, data encryption and decryption is fairly straightforward. The difficulties lie in keeping keys safe and transmitting them securely to other users.

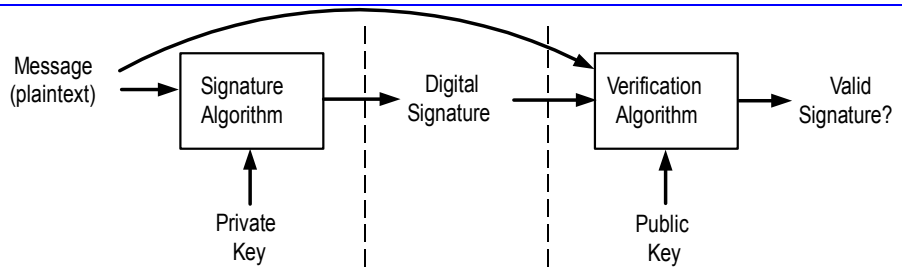
Hashing, Digital Signatures & Verification

A digital signature is used when you have plaintext to distribute and you want:

- The recipients to be able to verify that the message comes from you.
- The recipients to be able to verify that the plaintext hasn’t been tampered with since it left your hands.

Signing a message does not alter the message. It simply generates a digital signature string you can either bundle with the message or transmit separately.

Digital signatures are generated using public-key signature algorithms. A private key is used to generate the signature, and the corresponding public key is used to validate the signature. This process is shown in the following illustration:



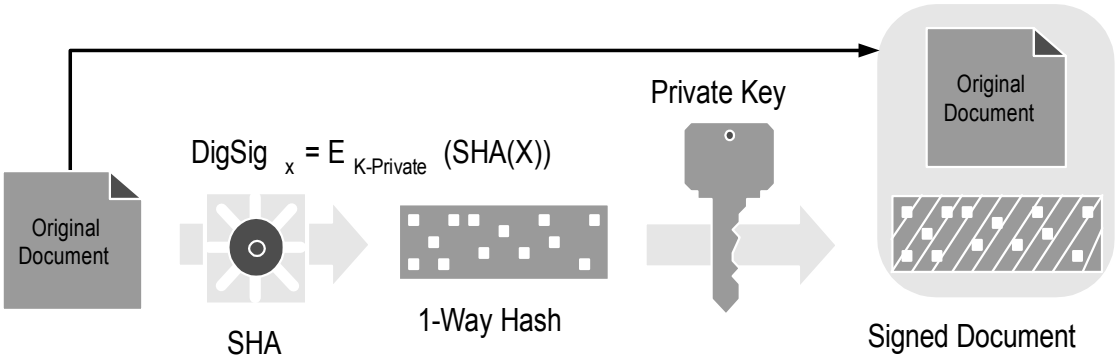
The Signature Generation and Verification Process

Digital signatures provide benefits separate from encryption. They allow users to verify that a document came from the holder of a private key, and that it hasn't changed since it was signed. (The document may or may not have been encrypted. It's a good cryptographic procedure to sign *before* encrypting — that way you know what you're signing. Imagine signing an envelope without knowing what's inside!).

Digital signatures are created by encrypting a hash of the document with a private key. The hash is essentially a miniature fingerprint of the document. The hashing functions used in cryptography are similar to the hashing functions used throughout computer science — functions which take a large amount of data and return a much smaller piece, usually of a fixed size .There are, however, a few key distinctions. In cryptography, hashing functions should be as "one-way" as possible. If you know the value of a hash (and even if you have the original document), it should be very difficult to create another document with the same hash value. It should be especially difficult to obtain the same hash value by simply altering a few characters .

A digital signature is a hash encrypted with a private signature key. Verifying a digital signature is done by decrypting the signature using the public signature key, and matching the result against a hash of the original document. (Good cryptographic procedure recommends using a different key specifically for signatures, rather than a general purpose key for both encryption or key exchange and signatures.)

The strength of a signature is dependent both on the quality of the one-way hash function, and on the strength of the encryption of that hash. If the one-way hash function can be subverted, then the original document might be changed. If the encryption isn't sufficiently strong, then the document might have come from someone other than the holder of the private key.



Hashing and Signing

So far, we know how to:

- Encrypt and decrypt documents
- Sign documents and verify signatures.

Both of these functions require our ability to distribute public keys and match them to the holder of the private keys. In other words:

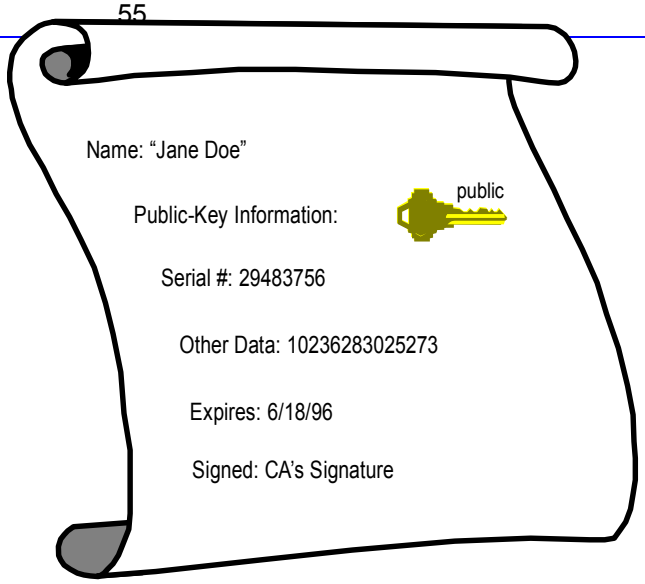
- If Alice wants to send Bob some encrypted data, she needs to know his public key.
- If Bob wants to verify that the digital signature on a document is Alice's, he needs to know her public key.

The question, then, is: How do you know who actually owns some public key? For example, if you received a key and were told that this was the key for your bank, would you believe it? One appropriate response might be "Says who?"

Certificates and Certificate Authorities

Certificates help answer this question. In essence, they are signed documents which match public keys to other information, such as a name or e-mail address. Certificates are issued and signed by certificate authorities (CAs). In essence, a certificate authority is a commonly-trusted third-party who verifies the matching of public keys to, for example, an identity, an e-mail name, or access privileges. Certificate authorities are similar to notary publics.

The benefit of certificates and CAs is that if two people both trust the same CA, then by exchanging certificates signed by the CA, they can learn each other's public keys. They can then safely use these keys to encrypt data they want to exchange, and to verify the signatures on document.



A Certificate

Certificate Creation

Certificate creation is straightforward, and has six steps:

1. Key generation. The individual requesting certification (this is the applicant, not the CA*)* generates key pairs of public and private keys.

2. Matching of policy information. The applicant packages up the additional information necessary for the CA to issue the certificate. This may be, for example, proof of identity, a tax ID number, or an e-mail address. The CA defines the actual requirements.

3. Sending of public keys and information. The applicant sends the public keys and information (often encrypted using the CA’s public key) to the CA.

4. Verification of information. The CA applies whatever policy rules it might have to verify that the applicant should receive a certificate.

5. Certificate creation. The CA creates a digital document with the appropriate information (for example, public keys, and the certificate's expiration date) and signs it using the CA's private key.

6. Sending or posting the certificate. Depending on what's appropriate, the CA can send the certificate to the applicant or post it publicly.

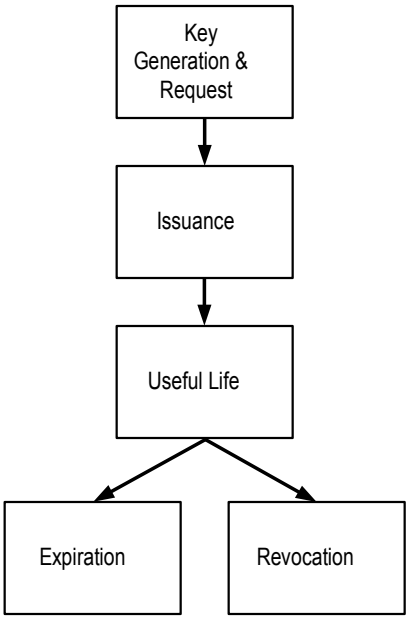
To verify a certificate, all that is necessary is the public key of the CA (and possibly a check against a revocation list). Certificates and CAs reduce the public-key distribution problem from one of verifying and trusting one (or more) public keys per individual to simply verifying and trusting the CA's public key and relying on that to allow verification of others.

Multi-Level Certificate Authorities

Certificate authorities can also certify sub-authorities who can issue their own certificates. These "trees of trust" reduce the burden on a centralized server. An example is a large corporation with four divisions. The main CA would certify four sub-CAs (one per division) that would issue certificates for everyone in each division. Cross-divisional certification works as long as everyone has the public key of the main CA. This is used to verify the credentials of the sub-CAs.

The Life of a Certificate

Certificates have a limited life. They are requested, created, and then, at some point, either expire or are revoked. Expiration is important because advances in computing power, and the potential for the discovery of holes in algorithms or protocols, can make certificates unreliable. Revocation is important if private keys are compromised, or there has been a change in status or policy. For example, a certificate indicating that TomC is an employee of Netwerks Corporation should be revoked if he leaves the company.



The Life of a Certificate

A revoked certificate should be entered on a certificate revocation list (CRL). This is similar to what happens when a credit card is revoked. A bank cannot force people to cut up their credit card, just as a CA cannot force the destruction of all copies of a certificate. However, in the process of requesting authorization for a purchase above some minimum value, a credit card is checked against a card revocation list to make sure that it is still valid. Someone who is going to use a certificate might want to check against a CRL to ensure the validity of the certificate.

Protocols and Examples

A cryptographic protocol is a combination of

- Encryption and decrypting.
- Hashing.
- Signing and verifying signatures.
- Issuing, checking, and revoking certificates.

Examples of protocols include:

- Key exchange protocols. These allow people who want to communicate, but either don't already have each others' public keys or have certificates, to safely exchange keys.
- Secure channel protocols. These allow people to have private, authenticated communications over a public network. An authenticated connection means that an impostor can't intrude and pretend to be one of the participants. An example of a secure channel protocol is the Secure Sockets Layer (SSL).

Strength

The strength of cryptographic methods depends on a number of things. First and foremost is that private keys be kept private. Here is a summary of some additional measures:

- Public-key encryption depends on the strength of the algorithm (such as RSA) and the length of the key.
- Symmetric-key encryption depends on the strength of the algorithm (such as DES and RC4) and the length of the key.
- Hashes depend on the strength of the hash algorithm.

- Digital signatures depend on the hash algorithm, the encryption algorithm used to sign the hash, and the length of the key.

- Certificates depend on the choice and implementation of policy by the CA, on the digital signature algorithm, on the revocation policy, and on the availability and use of CRLs.

The Need for Replaceable Security

Cryptography is vulnerable to both increases in computing power and discoveries of weaknesses in the algorithms. To be truly valuable, cryptographic engines must be easy to upgrade or replace.

For example, Moore's law (which states that computing power doubles every 18 months) shows that a cryptographic algorithm which would have taken 16,000 years to invert using brute-force techniques on a single PC in 1996 might take only 1,000 years to invert on a single PC in 2002. The availability of cycles on networked PCs only exacerbates this situation.

In addition, algorithms which were once thought to be secure may have holes. For example, it was initially believed the MD4 algorithm (a hashing algorithm) was difficult to subvert. It has, however, been proven to be insecure.

Protocols which relied upon key lengths or hashing algorithms may be fine in and of themselves, but be vulnerable because of insufficient key lengths, or a poor choice of algorithms. Rather than rewriting a protocol or solution from scratch, systems should allow key lengths to be increased or strong algorithms to be substituted when a weakness is found.