Tema 4: Seguridad y privacidad en aplicaciones telemáticas

Índice

Seguridad en email y herramientas

El **correo electrónico** es la aplicación más ampliamente utilizada en la gran mayoría de los entornos distribuidos. El crecimiento en su uso ha conllevado una mayor necesidad de seguridad, y más concretamente, de la integridad de servicios de **autenticación y confidencialidad**. Entre las soluciones de seguridad en email disponibles en la actualidad, hay dos que destacan por su amplio uso: PGP y S/MIME.

PGP (Pretty Good Privacy)

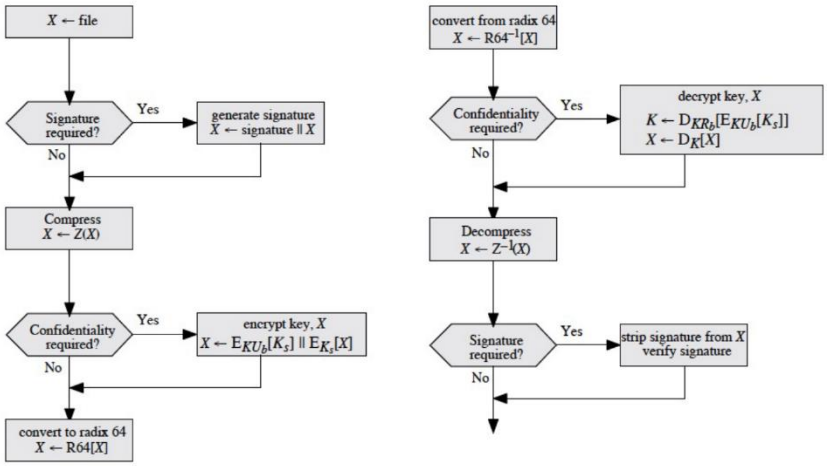
PGP es una solución diseñada por Phil Zimmerman en 1991. Básicamente, PGP consiste en seleccionar algoritmos criptográficos ya existentes e integrarlos en una única aplicación software independiente del sistema operativo. Integra los algoritmos RSA, DSS, Diffie-Helmann, CAST-128. IDEA, 3DES y SHA-1.

Proporciona servicios de **autenticidad y confidencialidad** que se pueden usar para **aplicaciones de email** y **almacenamiento de ficheros**. Existen versiones libres para Windows, UNIX y Mac, además de versiones comerciales.

Las operaciones de PGP incluyen, además de autenticación y confidencialidad, las operaciones de compresión y compatibilidad de email.

Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed, for storage or transmission, using ZIP.
Email compatibility	Radix 64 conversion	To provide transparency for email applications, an encrypted message may be converted to an ASCII string using radix 64 conversion.

El esquema de transmisión y recepción de mensajes PGP es el siguiente:



Nótese que se puede elegir entre **firma**, **confidencialidad (encriptar)** o **ambos.**

PGP proporciona para cada usuario U dos estructuras de datos:

1. ***Private-key ring**:* para almacenar los pares <clave pública, clave privada> del propio usuario U.

Private-Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
⋮	⋮	⋮	⋮	⋮
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
⋮	⋮	⋮	⋮	⋮

2. ***Public-key ring:*** para almacenar las claves públicas de los otros usuarios con que U se comunica.

Public-Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
T_i	$PU_i \bmod 2^{64}$	PU_i	$trust\_flag_i$	User i	$trust\_flag_i$		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Cada línea de la tabla de *public-key ring* puede considerarse en sí misma como un tipo de **certificado digital** (certificado de clave pública) **no estándar**. PGP no usa el estándar X.509, sino un formato propio. Tampoco basa su funcionamiento en la existencia de una PKI jerárquica de Autoridades de Certificación (CA), sino en un modelo PKI en malla. Esto conlleva que:

- **No existen CAs** (autoridades de certificación).
- **Cada usuario** del sistema **puede emitir certificados** al respecto de las claves públicas de los demás usuarios. De ahí los valores de confianza (*trust*) incluidos en el *public-key ring*.

OpenPGP

El **OpenPGP Working Group** se creó en 1997 y gracias en parte a *Internet Engineering Task Force* para definir el estándar. OpenPGP es una aplicación estándar y libre que permite cifrar mails usando criptografía de clave pública y un formato específico. Está basado en PGP original.

GnuPG

GnuPG es una implementación del estándar OpenPGP que deriva del software criptográfico PGP original. Con GnuPG se pueden realizar las siguientes acciones:

- **Gestionar y generar claves** públicas y privadas.
- **Visualizar y distribuir las claves** públicas, exportándolas e importándolas.
- **Cifrar y descifrar** documentos.
- **Firmar y validar** documentos.

S/MIME (Secure/Multipurpose Internet Mail Extension)

S/MIME es una mejora en el ámbito de seguridad del formato MIME para correo electrónico, el cual a su vez es una mejora del SMTP (*Simple Mail Transfer Protocol*). Aunque tanto PGP como S/MIME están envías de llegar a estándar, todo apunta a que **S/MIME se va a consolidar como estándar para uso comercial**, mientras que PGP quedará para uso personal.

En términos de funcionalidades, S/MIME es similar a PGP en el sentido de que ambos ofrecen la posibilidad de firmar y/o cifrar mensajes. S/MIME usa **certificados de clave pública con formato X.509v3**, con un modelo de PKI híbrido entre la jerarquía estricta de CAs y el modelo en malla. Utiliza los algoritmos criptográficos de la siguiente tabla:

Function	Requirement
Create a message digest to be used in forming a digital signature.	MUST support SHA-1. Receiver SHOULD support MD5 for backward compatibility.
Encrypt message digest to form a digital signature.	Sending and receiving agents MUST support DSS. Sending agents SHOULD support RSA encryption. Receiving agents SHOULD support verification of RSA signatures with key sizes 512 bits to 1024 bits.
Encrypt session key for transmission with a message.	Sending and receiving agents SHOULD support Diffie-Hellman. Sending and receiving agents MUST support RSA encryption with key sizes 512 bits to 1024 bits.
Encrypt message for transmission with a one-time session key.	Sending and receiving agents MUST support encryption with tripleDES. Sending agents SHOULD support encryption with AES. Sending agents SHOULD support encryption with RC2/40.
Create a message authentication code.	Receiving agents MUST support HMAC with SHA-1. Sending agents SHOULD support HMAC with SHA-1.

Operaciones:

- Crear hash del mensaje para usarlo en firma digital: SHA-1.
- Encriptar hash del mensaje para formar firma digital: DSS, RSA.
- Encriptar clave de sesión: Diffie-Hellman, RSA.
- Encriptar mensaje: TripleDES, AES, RC2/40.
- Crear código de autenticación del mensaje: HMAC, SHA-1.

Conexión remota segura

Telnet

Telnet (puerto 23) facilita el acceso remoto a otros sistemas y sobre TCP, de forma que el terminal local aparenta ser terminal del sistema remoto.

FTP (*File Transfer Protocol*)

FTP (puerto 20/21) permite la transferencia de ficheros entre diferentes recursos remotos.

Ambos presentan un **grave problema**, la información transferida y las acciones remotas se realizan en claro, por lo que no se recomienda su uso.

SSH (*Secure Shell*)

SSH es una herramienta similar a Telnet funcionando en el puerto 22. Permite el acceso remoto sobre TCP a otros sistemas usando el concepto de cliente/servidor pero usando las transacciones usando criptografía pública. Funcionamiento general:

- **Capa de aplicación**: gestiona la autenticación del cliente haciendo uso de un usuario/contraseña o aplicando criptografía de clave pública.
- **Capa de transporte**.
  - Gestiona o intercambia las claves iniciales.
  - Establece los modos de cifrado y de compresión.
- **Capa de red.**
  - Establece una "conexión directa" entre el cliente-servidor y restringe el tráfico entre esos puntos de conexión. **Modo túnel** (en base a cifrado simétrico).

Además mitiga o evita ataques específicos:

- **Spoofing de IP o suplantación de identidad**: nodos remotos intentan suplantar la identidad de otro nodo de la red.
- **Spoofing de DNS:** en donde el atacante trata de suplantar el nombre del servidor.

SFTP (*SSH File Transfer Protocol*)

SSH puede ser usado también para transferir ficheros como una alternativa a FTP, conocido como SFTP (*SSH File Transfer Protocol*) y SCP (*Secure Copy*). Funcionamiento de SFTP:

- Se puede basar de diferentes **modos de autentificación** para conectar con el servidor SFTP:
    - **Modo básico**: usuario y contraseña.
    - **Modo avanzado:** usando las claves públicas de SSH, previamente generadas, y compartiendo dichas claves públicas con el servidor SFTP. De esta forma, cuando el cliente quiere establecer conectividad con el sistema remoto, el proceso software del cliente tendrá que transmitir su clave pública al servidor para su autenticación. Todas las conexiones SFTP están cifradas.

FTPS (*File Transfer Protocol Secure*)

FTPS proporciona un soporte adecuado para TLS (*Transport Layer Security*) y SSL (*Secure Sockets Layer*). Funcionamiento:

- Se basa de un usuario, contraseña y certificados, de forma que:
    - Las credenciales de seguridad (usuario y contraseña) son cifrados a lo largo de la conexión FTPS.
    - Para ello, el cliente primero verifica que el certificado del servidor es correcto y de confianza para hacer uso de su clave.

Herramientas de cifrado

TrueCrypt

Herramienta de cifrado de discos duros disponible para Windows y Linux, haciendo uso de AES-256, Blowfish, CAST5, Serpent, Triple DES y Twofish. También permite ocultación de particiones haciendo uso de cifrado y aleatoriedad de la información.

DiskCryptor

Similar a TrueCrypt pero con capacidad de cifrar dispositivos de almacenamiento externo USB. También incluye algoritmos de cifrado como AES, Twofish y Serpen.

VeraCrypt

Similar a TrueCrypt pero con la diferencia que incluye un número específico de iteraciones para el cifrado, incrementando la lentitud del sistema durante los procesos de lectura y escritura en el disco. Aplica AES, Twofish y Serpen.

OpenStego

Aplica técnicas de **esteganografía** (del griego “cubierto”, “oculto” y “escritora”) para ocultar información haciendo uso de imágenes o cualquier archivo multimedia.

OpenPuff

Es similar a OpenStego para Windows con soporte para BMP, JPG, MP3, WAV y MPG4.

Salt

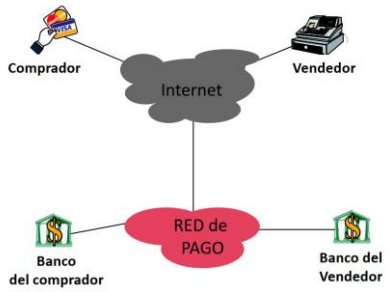
Las cuentas almacenadas en un disco duro de un sistema y protegidas con contraseñas, suelen tener asociado un hash a dichas contraseñas, **nunca se deben almacenar las contraseñas en claro**. Cuando el usuario quiere entrar al equipo se le pide la contraseña, se hace hash a dicha contraseña y se compara con el hash almacenado.

Esto presenta un **problema,** si alguien roba el fichero con los hash puede hacer fácilmente un **ataque de diccionario**. Para dificultar los ataques de este tipo se usa una **sal**, valores aleatorios que se asocian al hash. Los principios que debemos seguir al utilizar la sal son:

- **Nunca reutilizar la sal**, crear una sal aleatoria por cada contraseña, usando un generador aleatorio criptográfico (*Os.urandom()*).
- Usar **sal suficientemente larga**: 10 caracteres [a-z A-Z 0-9] 57 bits.
- Utilizar **funciones hash lentas**.

Seguridad en pagos electrónicos

En el ámbito del *ecommerce* se han desarrollado esquemas de pago electrónico que proporcionan en el mundo digital la misma **heterogeneidad** que los sistemas de pago tradicionales. En la mayoría de los sistemas de pagos electrónicos disponibles, los pagos se realizan a través de redes abiertas como internet. Pero la correspondencia entre los pagos electrónicos y las transferencias del valor real es realizada y garantizada por los bancos a través de los **sistemas financieros de compensación**. Estos sistemas utilizan para su funcionamiento las redes cerradas de las instituciones bancarias, las cuales son consideradas corporativamente más seguras.



En general, los pagos electrónicos involucran a un comprador y a un vendedor, adicionalmente existen sistemas que involucran a terceras partes confiables. Más aún, puede existir algún tipo de entidad que ejerza de **mediador** (TTP) para la **resolución de disputas**. Por lo general, las disputas se resuelven fuera del sistema de pago y en muchos casos el protocolo ni siquiera especifica cómo gestionarlas.

Clasificación pagos electrónicos

Existen varias formas de clasificar los sistemas de pagos electrónicos, dependen de:

- **Cuándo contacta el vendedor con el banco** para verificar el proceso de pago.
  - **Online**: antes de enviar el producto, el vendedor contacta con la entidad financiera para verificar la validez del pago del comprador.
  - **Offline**: cierto tiempo después de que el vendedor haya aceptado el pago y enviado el producto realiza el depósito de dinero que le ha dado el comprador. Para que la entidad financiera lo verifique y lo ingrese en su cuenta; es decir, el vendedor no contacta con el banco en el proceso de compra-venta.
- **Cuándo procede el comprador con la transacción y carga de dinero** en la cuenta del vendedor.
  - **Sistema de prepago:** el comprador ve decrementada su cuenta bancaria antes de realizar su compra. Este método se correspondería con los sistemas de **monedero electrónico** y **tarjetas telefónicas**. Este sería el sistema más análogo al papel moneda tradicional.
  - **Sistemas de pago instantáneo:** cuando al comprador se le realiza el cargo en cuenta justo en el momento de realizar la compra. Se correspondería con los sistemas actuales de pagos con tarjeta de débito.
  - **Sistemas de pospago:** cuando Alice realiza la compra, el Banco asegura al vendedor que se le hará efectiva la cantidad acordada. Pero Alice solo verá decrementada su cuenta cierto tiempo después de haberse realizado la compra.
- **La cantidad de dinero** implicada en cada transacción.
  - **Macropago:** mayor de 10€.

- **Pago:** entre 1 y 10€. Problema de **coste de implementación** (no tendría sentido utilizar un sistema de pago cuyo coste económico sea de orden de magnitud superior al importe de la transacción).
- **Micropago:** inferior a 1€.

Hay muchos protocolos que gestionan el proceso de pago online, tales como:

- Online y trazables: *First Virtual, CyberChash, iKP, SET*.
- Micropagos: *Paypal, Google Checkout, Amazon Payments, iTunes Store…*

Estos medios de pago presentan los siguientes **problemas:**

- Ataques de **escucha**
- **Suplantación de identidad**
- Generación de dato falso
- Modificación del dato

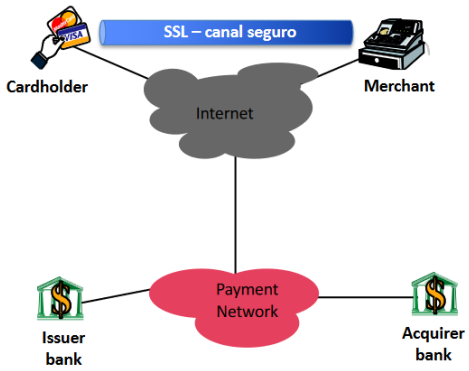
Soluciones para evitar estos problemas:

- Mecanismos criptográficos
- Mecanismos de autenticación de usuarios
- Firmas digitales
- Certificados digitales
- Y protocolos específicos que implementen estas soluciones.

Protocolos originales

SSL (*Secure Sockets Layer*), el protocolo para establecer conexiones seguras, **no es un protocolo de pago**, pero se usaba por seguridad. La última versión SSL v3 ya está obsoleta.

Entonces surge, **TLS** (*Transport Layer Security*), que utiliza el mismo formato para la cabecera de los paquetes que SSL, pero difiere en: número de versión, código de autenticación del mensaje (MAC), función pseudo-aleatoria, códigos de alerta, lista de algoritmos de cifrado, mensajes de verificación del certificado y de finalización, algunas partes del algoritmo criptográfico.



SSL aplica: **criptografía asimétrica** (certificados digitales, RSA o Diffie-Hellman) y **criptografía simétrica** (DES, 3DES, RC2, RC4 o IDEA). SSL **autentifica al servidor**, utilizando certificados digitales X.509v3, opcionalmente también puede certificar al cliente. Además, **asegura la integridad** de los datos, mediante códigos de autentificación de mensajes (MAC) y una clave secreta, MD5 o SHA-1. También proveía **confidencialidad** en el pago electrónico y garantizaba la creación de un canal seguro entre cliente y servidor. Sin embargo presentaba algunos **problemas:**

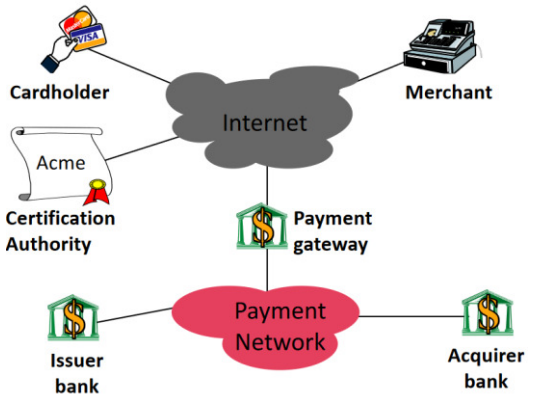
- **Solo protege transacciones entre dos puntos**, mientras que una transacción electrónica basada en una tarjeta de crédito involucra al menos a un banco.
- **No protege al comprador frente al vendedor**, el vendedor puede obtener información de la tarjeta que podría utilizar en un futuro de forma ilícita.
- **No hay mecanismos de autentificación de tarjetas**, no es la entidad bancaria quien autentica la entidad interesada.
- **No hay mecanismos de facturación o gestión de recibos**, cualquier reclamación queda a la buena voluntad del vendedor.

Por esto, se requerían otros tipos de protocolos más específicos.

Protocolo Set (*Secure Electronic Transactions*)

Protocolo desarrollado en 1996 por VISA y Mastercard (y American Express), en colaboración con IBM, Microsoft, Verisign, RSA, Netscape, GTE… Tenía el **objetivo** de proporcionar **seguridad** a las transacciones electrónicas basadas en **tarjetas de crédito**. De esta forma se reduce el **fraude mercantil** y se **garantiza el pago** a través de esas mismas redes.

La arquitectura de SET es la siguiente:



A diferencia de SSL fue diseñado para el comercio electrónico, sin embargo no es un sistema de pago en sí mismo, sino **un conjunto de protocolos de seguridad** y de formatos estándar que permiten a los usuarios usar de una forma segura a través de Internet la infraestructura ya existente de tarjetas de crédito.

Servicios

SET proporciona:

- **Confidencialidad** en las comunicaciones entre las entidades que intervienen en la transacción.
- **Autenticidad** a través del uso de certificados digitales X.509
    - Todas las entidades (cliente, vendedor, pasarela de pago… han de tener certificados X.509)
    - Es necesario el servicio de una o más **autoridades de certificación**.
- **Privacidad**, porque la información solo está disponible para las diferentes entidades cuando y donde es necesario.
- **Integridad**
- **Reduce las disputas debido al no repudio**
- Autorización de pago
- Confirmación de la transacción
- Garantía de pago del vendedor

Pasos de una transacción

Los pasos de una transacción son los siguientes:

1. **Petición de producto** (entre cliente y vendedor).
2. **Inicialización**: envío de certificados (entre cliente y vendedor).
3. **Información del pedido e instrucciones de pago**: descripción de la compra (entre cliente y vendedor).
4. **Petición de autorización** (entre vendedor, pasarela de pago y banco del comprador).
5. **Aprobación de autorización** (entre vendedor, pasarela de pago y banco del comprador).
6. **Finalización** (compensación entre la red de pago)

Firma dual

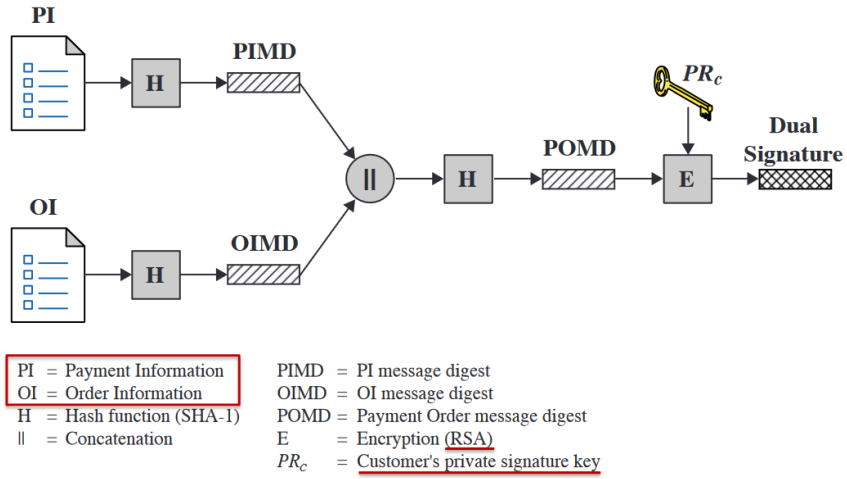
SET introduce el concepto de la **firma dual**. El propósito de este tipo de firma es enlazar dos mensajes que han de ir a receptores diferentes. En este caso es el cliente el que quiere enviar:

- **la información del pago** (*Payment Information*) al banco
- **la información del pedido** (*Order Information*) al comerciante.

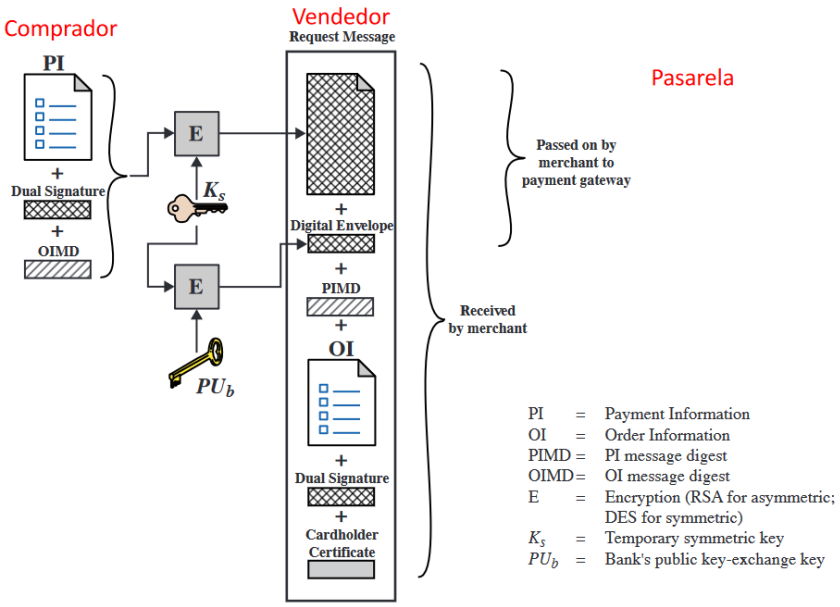
Se ofrece mayor **privacidad** al cliente si ambos ítems se mantienen por separado:

- ni el comerciante necesita conocer el número de tarjeta del cliente
- ni el banco necesita conocer los detalles del pedido del cliente.

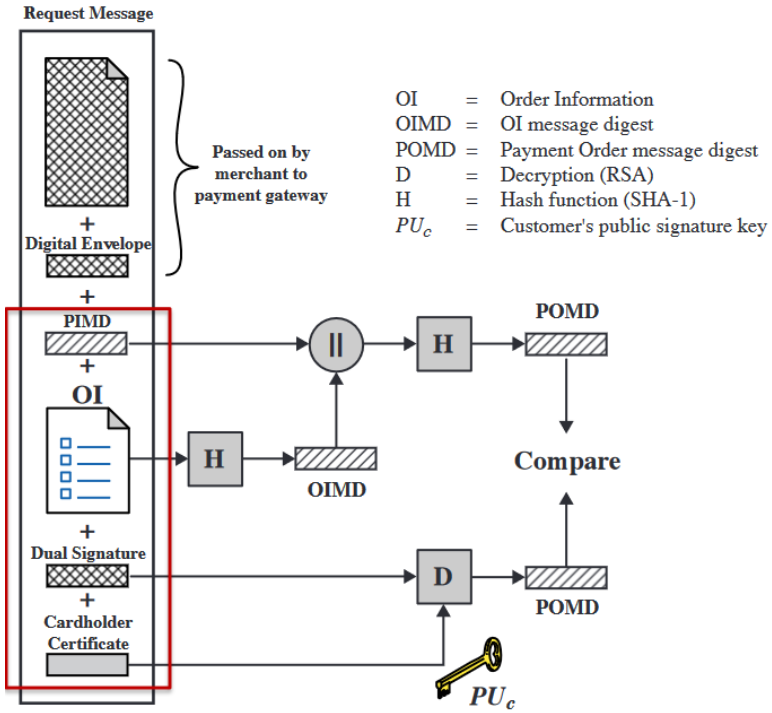
Pero también es necesario que ambos ítems queden entrelazados de alguna forma, para una posible **resolución de disputas**.



El comprador envía:

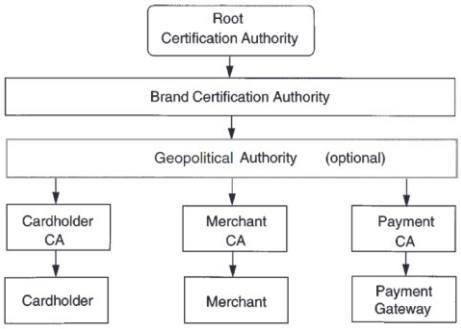


El vendedor comprueba:



PKI de SET

La PKI de SET se compone de las siguientes autoridades de certificación:



Las **ventajas** de SET son:

- Muy seguro y bien diseñado
- Garantiza autenticación, confidencialidad, integridad
- Garantiza no-repudio: si el banco del comprador autoriza el pago, el vendedor tiene la garantía de ese pago.
- Garantiza privacidad:
    - Evita que el vendedor acceda a los datos de la tarjeta.
    - Evita que el banco acceda a la información de los productos comprados.

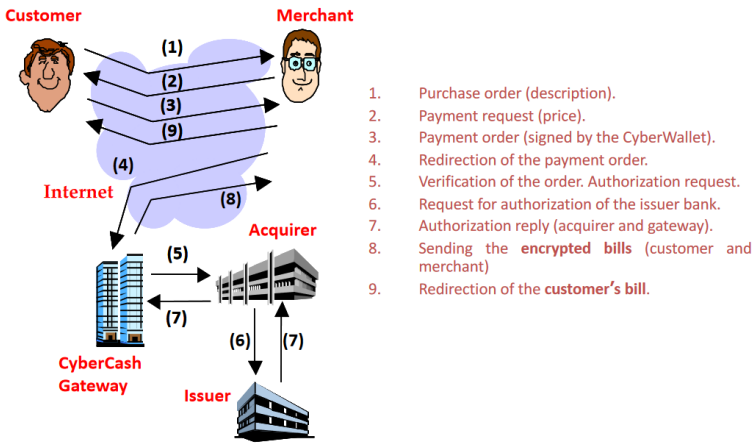
Las **desventajas** de SET son:

- Es dependiente de **algoritmos específicos** (RSA, DES, SHA1)
- Gestión de **certificados digitales**: fuerte esfuerzo para la implantación, especialmente para el vendedor.
- No adaptado a micropagos.

Protocolo Cybercash

El **protocolo Cybercash** se basa en el uso de una **pasarela propia** que gestiona los pagos electrónicos, permite el uso de cualquier tarjeta. Integra el software de cliente (*cyberwallet*) con la red financiera del banco del comprador. Se realiza la autenticación de todas las entidades y el cifrado de los datos relativos al pago.

El proceso de compra es el siguiente:



Algunos **problemas** que presenta son:

- Parte de la información del cliente es conocida por la pasarela, por lo que se pueden analizar los hábitos del cliente (problema de privacidad que no existía en SET).

- Uso de DES (56 bits) y RSA (1024 bits).

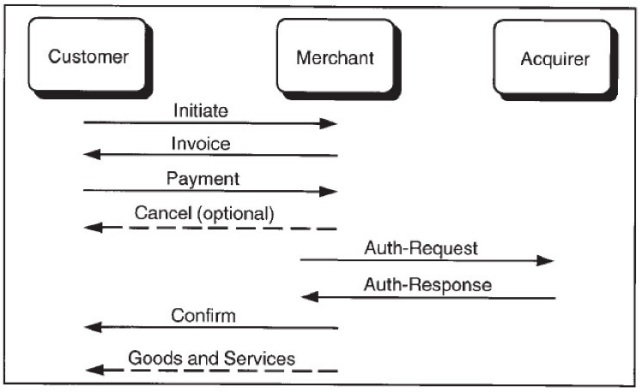
Tras caer en banca rota, Verisign adquirió los derechos de la marca y el protocolo, posteriormente fue comprado por PayPal.

Protocolo iKP

iKP (i = 1, 2, 3) es una **familia de protocolos** de pago, **basado en criptografía de clave pública,** y desarrollado por IBM. Estos protocolos se diferencian entre sí en el número de entidades que poseen su propio par de claves públicas-privadas, cuanto mayor sea el número de entidades que posean un par <clave pública, clave privada>, mayor es el nivel de seguridad proporcionado.

La implantación de los protocolos 2KP y 3KP se realiza de forma gradual para conseguir un **pago seguro multiparte** completo, requiriendo una **infraestructura de certificación** avanzada.

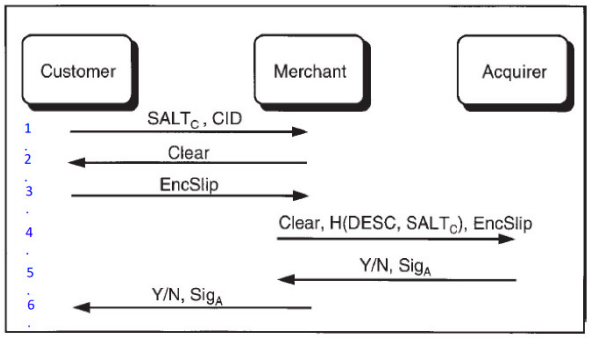
Cada uno de los protocolos consta de 6 pasos (varían según *i*).



El protocolo **1KP** es el más básico. Solo el *Acquirer* necesita poseer y distribuir $CERT_A$. La información de partida es:

Actor	Information Items
Customer	DESC, CAN, PK_{CA}, [PIN], $CERT_A$
Merchant	DESC, PK_{CA}, $CERT_A$
Acquirer	SK_A, $CERT_A$

Y los pasos del protocolo son los siguientes:



1KP también tiene **desventajas:**

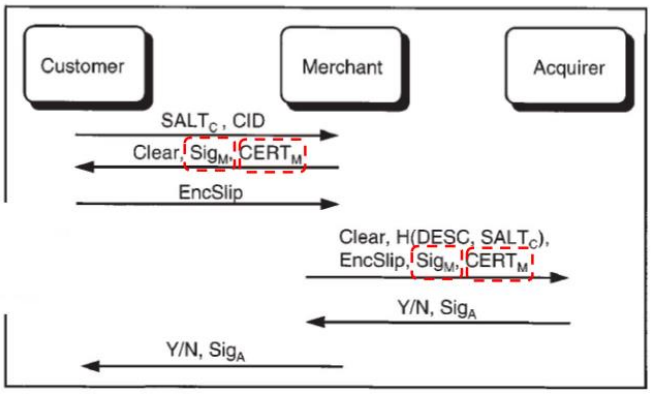
- El cliente se autentica utilizando solo un número de tarjeta y opcionalmente un PIN.

- El vendedor no se autentica ante ninguno.
- Ni el vendedor ni el cliente proporcionan evidencias de intervención.

En el protocolo **2KP** cada MERCHANT (vendedor) está obligado a distribuir su certificado $CERT_M$ al CLIENT y al ACQUIRER. La información de partida es:

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

Los pasos del protocolo son:



Millicent

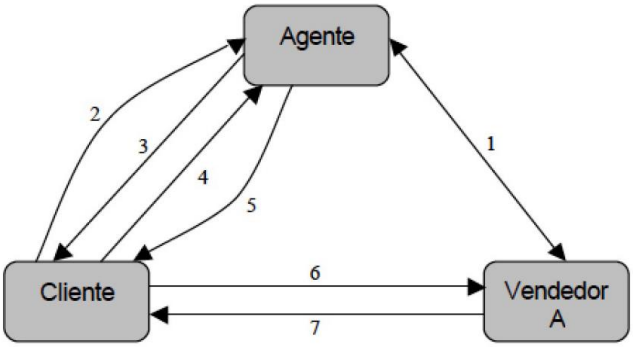
En algunos escenarios hay que transferir una cantidad muy pequeña (**micropago**), y por ello hay que buscar la forma más **eficiente y económica** posible. Minimizando el tráfico y los recursos utilizados, para que los costes de realizar el pago sean mínimos en comparación con el pago en sí mismo.

Para reducir los costes operacionales se utilizan varias soluciones:

- Servicios de prepago
- Autorizaciones off-line
- Agrupación de la facturación de los micropagos en lotes
- Reducción del coste computacional
    - Por tanto, la **criptografía asimétrica** no es adecuada, y la **simétrica** cuestionable. Cada vez se aplican más las **funciones hash**, pero conllevan falta de **no-repudio**.

Una solución es Millicent, basado en el cifrado simétrico y no utiliza procesamiento online. Además de clientes (*clients*) y comerciantes (*merchants*) existe la figura del **agente de negocios** (institución financiera). El sistema utiliza una forma de moneda electrónica, el **script** ("cupones electrónicos" que representan dinero, con los que el comprador obtiene la mercancía del vendedor).

Para un cliente no sería eficiente comprar lotes de scripts a cada uno de los potenciales vendedores del sistema. Se puede suponer que, durante un periodo, las compras de un cliente a varios comerciantes alcanzarán un importe equivalente a un macropago. La **función del agente de negocios** es vender a cada cliente, dentro de un mismo lote mixto, scripts de distintos vendedores. El modelo es el siguiente:



1. Compra-Venta de scrips de A
2. Compra de scrips de agente (macropago)
3. Envío de scrips de agente
4. Compra de scrips de A (micropago mediante scrips de agente
5. Envío de scrips de A
6. Petición producto + micropago mediante scrips de A
7. Envío de producto

El **modelo a tres bandas** (cliente, vendedor y bróker) ayuda a tener **anonimato** por parte del comprador. El agente conoce la identidad del comprador y su número de tarjeta de crédito pero nunca llega a conocer qué compra. El vendedor sabe lo que el cliente compra pero desconoce su identidad.

Privacidad de usuarios

Conceptos generales

La **privacidad** puede definirse como: "el derecho de los individuos y entidades de proteger, salvaguardar y controlar el acceso, almacenamiento, distribución y uso de información sobre su propia persona". **La confidencialidad no es equivalente a la privacidad**, es relativa a los datos, mientras que la privacidad es relativa a las personas. La información necesaria de proteger **depende de lo que el usuario considere privado**, identidad, localización, preferencias, rutinas…

Las propiedades de la privacidad son:

- **No-vinculación:** incapacidad de un atacante para relacionar dos mensajes o entidades observadas.
- **No observabilidad**: imposibilidad de distinguir la presencia de mensajes, ni la identidad de entidades.

Estrechamente relacionado está el concepto de **anonimato**: "es el estado de no ser identificable entre un grupo de sujetos, conocido como conjunto de anonimato". Las técnicas de anonimato tratan de hacer **indistinguible a un individuo** entre un conjunto suficiente de identidades con unas características similares. El anonimato depende de la técnica empleada y se distinguen 4 tipos:

- **Pseudónimos**: ocultar la identidad a través de pseudónimos. El uso continuado de pseudónimos puede ser vinculante, es decir, se puede derivar la identidad del usuario y todas las operaciones previas.
- **Anonimato rastreable**: se centra en ofrecer anonimato, pero en caso de necesidad se puede revelar la identidad del usuario. Por ejemplo entre un vendedor y un banco, dependiendo de la honestidad de las partes se puede o no revelar la identidad del usuario principal.
- **Anonimato no rastreable:** trata de resolver el problema del anonimato pero garantiza que la identidad de los usuarios no se va a revelar.
- **Anonimato no rastreable y no vinculante**: además de garantizar que la identidad de los usuarios no se puede revelar, la condición de vinculante asegura que el conjunto de las opciones realizadas por un usuario anónimo no se puede vincular.

Para conseguir una o varias de las propiedades mencionadas, las soluciones suelen basarse en el uso de alguna de las siguientes técnicas:

- **Esquemas avanzados de firma digital.**
- **Protocolos criptográficos y de enrutado**: evitan que la dirección de red o el camino que siguen los paquetes puedan identificar a las partes comunicantes.
- **Técnicas de ofuscación**: mecanismos basados en la generalización o supresión de información, para limitar la precisión de la información que se revela.

Protocolos basados en esquemas avanzados de firma digital

A partir del concepto básico de firma digital estudiado en temas anteriores, surgen esquemas más avanzados de firma, con objetivos más ambiciosos:

- **Firma ciega:** el firmante firma mensajes para otros usuarios, desconociendo el contenido de los mensajes que firma.
- **Firma de grupo**: cualquier miembro del grupo firma mensajes de forma anónima en nombre del grupo. En caso de disputa una entidad determinada puede revelar la identidad del firmante (**anonimato rastreable**).

- **Firma de anillo:** similar a la anterior pero el anonimato es total, no es posible saber la identidad del firmante bajo ninguna circunstancia (**anonimato no rastreable ni vinculante**).
- **Firma umbral:** la firma la producen un mínimo de *t* usuarios, en nombre del grupo de *n* que forman parte (*t* < *n*).

Firma ciega

Existen aplicaciones donde una de las propiedades a preservar es el anonimato del usuario. Concretamente con la firma ciega, lo que se consigue es que el mensaje *M* generado por *Alice* sea firmado por *Bob* **sin que conozca el contenido del mensaje**. La firma ciega resultante **puede ser verificada públicamente con posterioridad,** como una firma digital normal.

Este esquema se puede implementar con diferentes algoritmos de clave pública, por ejemplo RSA:

1. Alice crea *r* (número aleatorio *mod n*, llamado "binding factor").
2. Alice crea: $M' = M \cdot r^e \ mod \ n$.
3. Alice → Bob: M'.
4. Bob → Alice: $(M')^d \ mod \ n = M^d \cdot r^{e \cdot d} \ mod \ n = M^d \cdot r \ mod \ n$
5. Alice: $M^d \cdot r \cdot r^{-1} \ mod \ n = M^d \ mod \ n$

Donde.

- *e, d* son las claves pública y privada de Bob.
- *n* es el módulo RSA.
- *r* es un número aleatorio módulo *n*.

Firma de grupo

Al contrario que con los esquemas tradicionales de firma, en los que solo hay un firmante los esquemas de firma en grupo permiten que cualquier miembro de un grupo firme un documento (en nombre del grupo). La figura del **administrador del grupo**, quien controla quién pertenece al mismo y también emite la clave de firma en grupo (la clave con que cualquier miembro firma en nombre del grupo).

Por tanto es estos esquemas hay tres tipos de participantes:

- Administrador del grupo.
- Miembro del grupo.
- Verificador (receptor del documento firmado).

Un esquema de firma de grupo debe satisfacer las siguientes propiedades o condiciones iniciales:

- Solo los **miembros del grupo** pueden **firmar mensajes de forma correcta** (infalsificable).
- A excepción del administrador del grupo nadie puede descubrir:
  - qué miembro del grupo firmó el mensaje (anonimato).
  - si dos firmas han sido emitidas por el mismo miembro del grupo (no-vinculación).
- Los miembros **no pueden evitar** la **apertura de la firma** por parte del administrador, ni firmar por otro.

Un esquema de firma de grupo consta de 4 procedimientos:

1. **Establecimiento**: es un protocolo entre el administrador y los miembros del grupo. Su ejecución origina:
   a. Clave pública *Y* del grupo.

   b. Las claves privadas individuales *x* de cada miembro
   c. Una clave secreta de administración para el administrador
2. **Firma**: es un algoritmo que:
   a. Tiene como entrada un mensaje *M* y la clave privada de uno de los miembros del grupo.
   b. Devuelve una firma *S* sobre el mensaje *M*.
3. **Verificación**: un algoritmo que:
   a. Recibe como entrada un mensaje *M*, una firma *S* y la clave pública *Y* del grupo.
   b. Devuelve información de *M* si la firma es correcta o no.
4. **Apertura**: algoritmo que:
   a. Recibe como entrada una firma *S* y la clave secreta de administración.
   b. Devuelve la identidad del miembro del grupo que realizó la firma *S*, además de una prueba.

(Se asume que todas las comunicaciones entre el administrador y los miembros se llevan a cabo de forma segura).

Ejemplo 1. El administrador proporciona a cada miembro del grupo una lista de claves privadas (disjuntas). El administrador divulga, en un directorio público, y en orden aleatorio, la lista completa de claves públicas correspondientes (esta lista hace de clave pública del grupo). Cada miembro puede firmar un mensaje con una de las claves privadas de su lista. El receptor puede verificar esa firma con la correspondiente clave pública (obtenida del directorio). El administrador conoce todas las claves privadas, por lo que en caso de ser necesario, puede saber fácilmente qué miembro realizó la firma.

Firma de anillo

Las firmas de grupo son útiles cuando los miembros quieren cooperar, mientras que las de anillo son útiles **cuando no quieren o no pueden**. Al contrario que en las de grupo:

- No tienen administradores de grupo.
- No tienen procedimiento de establecimiento.
- No tienen procedimiento de revocación del anonimato del firmante.
- No tienen coordinación.
- No tienen procedimiento para distribuir claves.

Cualquier usuario puede elegir un conjunto de posibles firmantes, incluyéndose él mismo, sin la aprobación ni ayuda de esos otros miembros del conjunto. Computa la **firma** por sí solo, **usando solo su clave privada y las claves públicas de los demás**.

Es decir, los otros miembros del conjunto pueden tener desconocimiento total de que sus claves públicas se están utilizando para ese proceso de firma, con el que quizás ni siquiera estén de acuerdo. El verificador es incapaz de determinar la identidad del firmante dentro de un anillo de tamaño *r*.

Privacidad basada en protocolos criptográficos y de enrutado

Tratan de **proteger el tráfico** frente a entidades que se dedican a observar las comunicaciones. Existen soluciones basadas en: uso de proxy, uso de mezcladores, conocimiento parcial de la ruta, creación de grupos y soluciones híbridas.

Uso de proxy

Un **servidor proxy** hace de intermediario en la comunicación, aceptando conexiones de los clientes y reenviándolas. Destacan:

- **Anonymizer:** el cliente solicita una página web al proxy y este origina una nueva petición, almacenando internamente el cliente y la petición que realizó para reenviarle la respuesta.
    - *Anonymizer* es un software comercial que mientras está activado permite la navegación a través de un proxy que oculta la verdadera dirección IP del usuario.
    - Actualmente es posible **el cifrado entre cliente y proxy** para proporcionar confidencialidad al enlace.
- **LPWA** (*Lucent Personalized Web Assistant*): elimina cualquier información de identidad de los datos del usuario y crea una conexión SSL entre el usuario y el proxy.
- **Proxy caché**: cuando un usuario solicita una página web, el proxy la recupera de una caché donde guarda las peticiones de otros usuarios. Esto mejora el tiempo de respuesta, además de evitar ataques en los que se pueda relacionar el tráfico generado por un cliente con la recibida por un servidor.

Uso de mezcladores

Mixnets: se basan en la utilización de *mixers*, que son dispositivos de almacenamiento y envío:

- El usuario envía el mensaje a través de *mixer*.
- Estos lo almacenan durante cierto tiempo con el fin de que puedan ser mezclados con otros mensajes recibidos saliendo del *mixer* en un orden diferente.
- El esquema solo funciona si el **número de mensajes almacenados temporalmente por el *mixer* es suficientemente grande.**

Conocimiento parcial de la ruta

- **Onion routing**: el *onion proxy* crea un paquete cifrado en capas, que se irán "pelando" a medida que atraviese el camino de onion routers.
- **TOR**: es la segunda generación de Onion routing.
    - Evita algunas deficiencias originales, añadiendo control de congestión, comprobación de integridad…
    - Permite navegar a través de una ruta privada de la red TOR, creada de manera aleatoria entre los distintos repetidores (onion routers) de la red.
    - Funcionamiento:
        - Enrutamiento: los paquetes se envían a través de varios routers onions, los cuales son elegidos de **forma aleatoria** y previamente por un **servidor central**.
        - Todos los nodos Tor son elegidos al azar y ningún nodo puede ser utilizado dos veces.
        - Se conecta a un nodo aleatorio a través de una conexión cifrada. Una vez que el camino ya es conocido desde el origen, cada conexión y salto en la red deberá ser cifrada, excepto con **el penúltimo nodo** de la comunicación, el cual hará una conexión no cifrada.
        - El enrutamiento es asimétrico. Alice lo primero que hace es cifrar el mensaje con la clave pública del último router onion de la lista, para que este último lo pueda descifrar, y así con todos los demás.
        - Para evitar el **análisis de tráfico pasivo**, cada 10 minutos se cambian los nodos de la conexión.
    - Es lento por su arquitectura: basada en routers y saltos de router en router usando criptografía asimétrica.
    - Garantiza el anonimato, pero no la privacidad de los datos.

Creación de grupos

- **Crowds**: los emisores se agrupan creando una "multitud" en la que todos enrutan de manera aleatoria los paquetes recibidos de sus compañeros.
    - Cada nodo solo conoce el destino y el nodo del que recibe el paquete.
    - Los nodos comparten claves con sus vecinos para cifrar
    - El camino seguido por el mensaje es almacenado temporalmente en cada nodo en el que se transita para saber enrutar de vuelta (con la respuesta).
- **Hordes**: muy similar a Crowds con algunas diferencias (respuestas):
    - En este caso se hace un broadcast de la respuesta desde el router a todos los miembros del grupo donde se encuentra el originador del mensaje.
    - Esta diferencia supone una mejora significativa en términos de rendimiento (tiempo) frente a Crowds.
    - Sin embargo es menos robusto, ya que el mensaje broadcast da ciertos indicios sobre la localización del usuario.

Soluciones híbridas

- **Tarzan**: es una solución que combina la **creación de grupos** con el uso de **onion routing**.
    - Cuando un nodo quiere enviar un paquete elige un camino dentro del grupo y le aplica un cifrado sucesivo con las claves simétricas de cada nodo del camino.
    - Estos al descifrar serán capaces de determinar el siguiente salto.
    - Al final, un nodo especial se encarga de sustituir la dirección de cada nodo por un identificador del grupo y viceversa.

	Main goal	**Architecture**	**Techniques**						
			SK	PK	LE	PD	PR	FT	MB
Single-proxy	Sender Anonymity	Centralized	√						
Mix-nets	Unlinkability			√	√	√			
Onion routing			√	√	√			√	
Tor			√	√	√				
Crowds	Sender Anonymity	Decentralized	√						
Hordes			√	√					√
GAP	Unobservability		√	√		√	√	√	
DC-nets			√						√
Herbivore			√	√					√