Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

Modelo de Transacciones Comerciales Seguras Utilizando un Protocolo Zero Knowledge para la Autentificación

Tesis que para optar el grado de
Maestría en Ciencias Computacionales con Especialidad en Redes
Presenta

Lic. Gabriel Barrera Delgadillo

Asesor: **Dr. Carlos Rodríguez Lucatero**

Comité : **Dr. José de Jesús Vázquez Gómez**
Dr. Gustavo Santana

Jurado: **Dr. Gustavo Santana**
Dr. José de Jesús Vázquez Gómez
Dr. Carlos Rodríguez Lucatero

Atizapán de Zaragoza, Edo. de México, Junio 1999.

RESUMEN

En esta investigación se plantea una aplicación de autentificación para el comercio electrónico seguro utilizando un protocolo Zero Knowledge para la autentificación de las partes y el protocolo SSL para enviar los datos de la petición. Dentro de los capítulos se plantean todas las bases para poder tener un comercio electrónico seguro.

Como parte de esta investigación se analizarán varias herramientas que actualmente se utilizan en el comercio electrónico, como Digicash, E-Comm, CyberCash, Microsoft Wallet, entre otros, además de brindar esquemáticamente sus funcionalidades.

En el comercio electrónico se utilizan principalmente dos protocolos uno es el SSL y el otro es un protocolo que podrá ser uno de los más fuertes en los próximos meses, me refiero al protocolo SET. Ambos protocolos son explicados en este trabajo además de hacer ciertas críticas sobre su funcionamiento.

Como protocolo sugerido para esta tesis el Zero Knowledge tiene un apartado especial en el cual se explica su funcionalidad y propiedades. Para poder aplicar el protocolo Zero Knowledge es necesario conocer algunos aspectos de la Teoría matemática, los cuales son explicados en el capítulo 4.

Al final de esta investigación se da una amplia visión de cómo se puede constituir un protocolo para las transacciones comerciales seguras.

CONTENIDO

RESUMEN 1

CONTENIDO 2

LISTA DE FIGURAS 6

ABREVIATURAS Y SÍMBOLOS 7

CAPÍTULO 1 8

1.1 ANTECEDENTES 8
1.2 PLANTEAMIENTO DEL PROBLEMA 9

CAPÍTULO 2 11

2.1 INTRODUCCIÓN AL COMERCIO ELECTRÓNICO 11
2.1.1 SITUACIÓN ACTUAL DEL COMERCIO ELECTRÓNICO 12
2.2 TIPOS DE RELACIONES 13
2.2.1 RELACIONES BANCO-BANCO Y BANCO-CLIENTES 13
2.2.1.1 RELACIONES EMPRESA-EMPRESA 13
2.2.1.2 RELACIONES EMPRESA-CLIENTE 14
2.3 MODELOS PARA COMERCIO ELECTRONICO EN REDES DE INTERCONEXIÓN. 15
2.3.1 MODELO ABIERTO 15
2.3.2 MODELO CERRADO 16
2.3.3 MODELO ABIERTO PERO ACOTADO 16
2.3.4 MODELO CERRADO Y ACOTADO 17
2.4 TIPOS DE PAGO 18
2.5 SISTEMAS PARA EL PAGO EN INTERNET 19
2.5.1 FSTC (FINANCIAL SERVICES TECHNOLOGY CONSORTIUM) 19
2.5.2 CHECKFREE 20

2.5.3 FIRST VIRTUAL (FV) 21
2.5.4 NETMARKET 22
2.5.5 NETBILL 22
2.5.6 NETCASH Y NETCHEQUE 23
2.5.7 CYBERCASH 23
2.5.8 DIGICASH 25
2.5.9 MONDEX INTERNATIONAL 26
2.5.10 OPEN MARKET 26
2.5.11 ¿QUÉ ES E-BUSINESS DE IBM? 27
2.5.12 MICROSOFT WALLET 27
2.5.13 E-COMM 28
2.6 CONCLUSIONES **28**

CAPÍTULO 3 **30**

3.1 INTRODUCCIÓN A LA CRIPTOLOGÍA **30**
3.2 PROCEDIMIENTOS CLÁSICOS DE CIFRADO **32**
3.2.1 PRINCIPIO DE SUSTITUCIÓN Y TRASPOSICIÓN 32
3.2.1.1 EJEMPLOS HISTÓRICOS DE CIFRADO POR SUSTITUCIÓN. 33
3.2.2 CONDICIONES DEL SECRETO PERFECTO 35
3.2.3 APLICACIÓN PRÁCTICA DEL CIFRADO EN FLUJO 37
3.3 FIRMAS DIGITALES **38**
3.3.1 INTRODUCCIÓN A LAS FIRMAS DIGITALES 38
3.3.2 METODOLOGIA DE LA CRIPTOGRAFÍA DE LLAVES 40
3.3.2.1 UTILIZACIÓN DEL MODELO DE LLAVE PÚBLICA PARA LA AUTENTIFICACIÓN 40
3.4 PROTOCOLOS DE SEGURIDAD UTILIZADOS ACTUALMENTE **43**
3.4.1 SECURE SOCKET LAYER (SSL) 44
3.4.1.1 DESCRIPCIÓN 44
3.4.1.2 IMPLEMENTACIÓN DEL PROTOCOLO SSL 46
3.4.2 PROTOCOLO SET 49
3.4.2.1 CARACTERÍSTICAS PRINCIPALES DE SET 50
3.4.2.2 ENTORNO 50
3.4.2.3 JERARQUÍA DE CERTIFICACIÓN 50
3.4.2.3 AUTORIDADES DE REGISTRO 51
3.4.2.4 PAGO ELECTRÓNICO 51
3.4.2.5 VENTAJAS SOBRE SSL 52
3.4.2.6 DEFECTOS DE SET 1.0 53
3.4.2.7 SET 2.0 54
3.4.3 IMPLEMENTACIÓN DE SET EN ESPAÑA 54
3.4.3.1 OBTENCIÓN DE CERTIFICADOS 56
3.4.3.2 ASOCIACIONES 56
3.5 EXPLICACIÓN DEL FUNCIONAMIENTO DE UN PROTOCOLO DE AUTENTIFICACIÓN BASADO EN ZERO KNOWLEDGE **57**
3.5.1 TERMINOLOGÍA ZERO KNOWLEDGE 58

3.5.2 PROPIEDADES GENERALES DE LOS PROTOCOLOS ZERO KNOWLEDGE 58
3.5.3 MODOS DE OPERACIÓN 59
3.6 PRUEBAS ZERO KNOWLEDGE 60
3.6.1 PRUEBAS INTERACTIVAS 60
3.6.1.1 PRUEBA DE LA CAVERNA DE ALI BABA 61
3.6.1.2 PRUEBA CON MÓDULO N 62
3.7 RESISTENCIA CRIPTOGRÁFICA DE LOS PROTOCOLOS ZERO KNOWLEDGE 63

CAPÍTULO 4. 64

4.1 OBJETIVOS DEL MODELO. 64
4.2 TEORÍA NUMÉRICA FUNDAMENTAL PARA CRIPTOGRAFÍA 65
4.2.1 ARITMÉTICA MÓDULO N 65
4.2.2 NÚMEROS PRIMOS 66
4.2.3 MÁXIMO COMÚN DIVISOR 66
4.2.4 INVERSO DEL MÓDULO DE UN NÚMERO 67
4.2.5 TEOREMA PEQUEÑO DE FERMAT 68
4.2.6 TEOREMA DEL RESTO CHINO 68
4.2.7 RESIDUOS CUADRÁTICOS 69

CAPÍTULO 5 71

5.1 INTRODUCCIÓN 71
5.2 PERSONAJES DEL PROTOCOLO 71
5.3 OBJETIVOS DEL PROTOCOLO 72
5.4 PROTOCOLO GENERAL DE LA TRANSFERENCIA DE FONDOS 72
5.4.1 PROTOCOLO GENERAL 72
5.4.2 PROTOCOLO ZERO KNOWLEDGE 74
5.5 RESULTADOS OBTENIDOS 80
5.6 ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS 80
5.6.1. CRITERIOS DE SEGURIDAD 80
5.7 CONCLUSIONES Y RECOMENDACIONES 82

BIBLIOGRAFÍA 83

ANEXO A: EJEMPLO DE MENSAJES 86

ANEXO B. ESTRUCTURA DEL PROTOCOLO SSL 3.0 103

B. Constantes del Protocolo SSL 103
B.1 Asignación de un puerto reservado 103

B.1.1 Nivel de Registros 103
B.2 Cambio de cifrado del mensaje esperado 104
B.3 Mensajes de Alerta 104
B.4 Protocolo Handshake 105
B.4.1 Mensajes de Inicialización (Hello) 105
B.4.2 Autentificación en el servidor e intercambio de mensajes cifrados 106
B.5 Autentificación en el cliente e intercambio de mensajes cifrados 108
B.5.1 Mensaje de Finalización Handshake 109
B.6 La Suite de Cifrado 109
B.7 La especificación CipherSpec 110

ANEXO C 112

C.1 CÓDIGO DE UN EJEMPLO DEL PROTOCOLO EN JAVA 112

LISTA DE FIGURAS

		Página
Figura [1]	Modelo Abierto	15
Figura [2]	Modelo Cerrado	16
Figura [3]	Modelo Abierto pero Acotado	17
Figura [4]	Modelo Cerrado y Acotado	17
Figura [5]	Transacción del FSTC	20
Figura [6]	Funcionamiento de CheckFree	20
Figura [7]	Funcionamiento de First Virtual.	21
Figura [8]	Funcionamiento de Netmarket.	22
Figura [9]	Funcionamiento de NetBill	23
Figura [10]	Funcionamiento de Cybercash	25
Figura [11]	Funcionamiento de DigiCash	26
Figura [12]	Funcionamiento del Proceso Criptográfico	30
Figura [13]	Funcionamiento del Handshake	46
Figura [14]	Funcionamiento del SSL	47
Figura [15]	Jerarquía de Certificación	51
Figura [16]	Funcionamiento del SET	52
Figura [17]	Obtención de Certificados	56
Figura [18]	Caverna de Alibaba	61

ABREVIATURAS Y SÍMBOLOS

ACH	(Automatic Clearing House).
SSL	(Secure Socket Layer).
RPS	(Remote Pay System).
PGP	(Pretty Good Privacy)
Kp	Llave Pública
Ks	Llave Privada
RSA	Rivest, Shamir y Adleman
DES	Data Encryption Standard
CA	Autoridad Certificadora
PKCS	(Public Key Cryptography Standard)
EMV	Europay-MasterCard-Visa
FFS	Feige- Fiat-Shamir

CAPÍTULO 1

1.1 ANTECEDENTES

En la actualidad el comercio mundial se ha visto intensificado, ya que desde 1992 a la fecha se habla de la globalización en todos los sentidos. Para los fines de la globalización en Internet existen muchos caminos para poder comerciar, uno de ellos, que ha tomado mucho auge en nuestros días, es el comercio electrónico. El Comercio Electrónico es una nueva forma de entender la economía que está llamada a ocupar un importante espacio dentro de nuestra sociedad moderna. Entender en que consiste el e-commerce o Comercio Electrónico es una tarea bastante complicada ya que se trata de un concepto que agrupa una gran cantidad de factores.

La enorme expansión que ha tenido Internet en los últimos años ha significado un cambio importante en la forma de relacionarse de muchas personas y empresas. Internet significa un acercamiento entre compradores y fabricantes independientemente de la distancia. Esto supone un gran atractivo para las empresas que con una inversión mínima pueden acercarse a un enorme mercado potencial en crecimiento constante. La primera aplicación al mundo de los negocios de esta tecnología fue la promoción y el mercadeo o publicidad por la red. Actualmente Internet ya se ha consolidado como el cuarto medio de información junto a la Prensa, la Radio y la Televisión. Tras el éxito del mercadeo o publicidad en la red, el siguiente paso es evidente e inevitable: vender directamente por Internet, ahorrándose a los intermediarios y pudiendo así rebajar los precios de los productos o servicios. Esto no es más que una extrapolación de la venta a distancia, por catálogo o por la venta o mercadeo televisivo, donde el pago de estos bienes o productos se realiza a la hora de ser entregados al cliente.

El negocio por Internet está estrechamente ligado al número de usuarios de la propia red, que a su vez está condicionado por el precio de las conexiones a Internet. Como se ha visto actualmente estos precios han ido a la baja, por lo que ahora es más sencillo conseguir una conexión a Internet.

Con la utilización de tarjetas de crédito en Internet, se pueden efectuar pagos electrónicos de modo que la transacción se simplifica bastante. Es más, si el bien adquirido es susceptible de ser enviado digitalmente (como audio, vídeo, documentación digital o cualquier otro tipo de información digitalizada), la compra puede realizarse enteramente por Internet.

Precisamente cuando se habla de Comercio Electrónico nos referimos concretamente al pago electrónico a través de redes de telecomunicaciones digitales ya sean públicas o privadas; aunque, como se ha explicado anteriormente, las implicaciones económicas del Comercio Electrónico van mucho más allá.

Internet es el medio ideal para este tipo de negocio pero no el único. Se está pensando seriamente en la comercialización de sistemas que permitan la navegación por Internet a través de la televisión, estos sistemas reciben el nombre de WebTV. Funcionen o no, estos sistemas, lo que es seguro es que la sociedad actual estará cada vez más interconectada por lo que las perspectivas a mediano plazo del comercio electrónico son extraordinarias. Como ya se ha demostrado con la conocida Amazon.com, una empresa que ha conseguido convertirse en una de las mayores librerías del mundo vendiendo únicamente por Internet.

El inconveniente principal para que el comercio electrónico prospere en Internet es que no se tiene la completa seguridad de que los datos viajarán a través de la red en una forma segura; por tal motivo el enviar un número de tarjeta de crédito por la red no es muy confiable.

Pero no ha sido un obstáculo ya que desde hace algunos años se han estado desarrollando varios tipos de servicios de seguridad los cuales tratan de brindar la confianza necesaria para el comercio electrónico. Dichos servicios utilizan protocolos como los que menciono a continuación.

El protocolo SSL es un estándar de seguridad el cual provee de la privacidad para enviar datos a través de Internet. En una transmisión normal de datos, el cliente se conecta a un puerto (socket) en el servidor. Los datos son escritos y leídos en dicho puerto en forma de texto plano, permitiendo que la información pueda ser interceptada por una persona con un software de escucha o rastreo (snooping). El protocolo SSL está basado en la capa de los protocolos de sockets normales; éste provee de un cifrado de los datos antes de ser escritos en el socket, y los datos son descifrados por el receptor. Para este fin, el cliente y el servidor deben coordinar las actividades para que se asegure la compatibilidad de los algoritmos de cifrado para que los datos en ambos sean compatibles. Dentro del desarrollo de esta tesis se dedica un capítulo a la explicación del funcionamiento de este protocolo.

Los protocolos Zero Knowledge se han utilizado para brindar la confianza en la autentificación de las partes involucradas en la transferencia de información, por lo que considero importante que se incluya en este modelo uno de los protocolos Zero Knowledge, el cual permita identificar a las partes sin compartir el secreto, considerando un modo de operación interactivo. Este modelo en la actualidad se está utilizando para garantizar la privacidad en Internet, y es utilizado por la compañía Zero Knowledge Systems en Estados Unidos.

1.2 PLANTEAMIENTO DEL PROBLEMA

El problema fundamental y que trata de resolver esta tesis es el cómo mejorar de una manera sencilla y óptima el comercio electrónico en Internet, haciendo confiable la transferencia electrónica de pedidos y de números de tarjetas de crédito, además de cumplir con los estándares de seguridad en la transmisión, es necesario garantizar la privacidad de las personas o empresas que se ven involucradas en el proceso de comercio electrónico.

Por lo que he decidido investigar un modelo de transferencia de información por el cual el cliente y el proveedor de servicios tengan la certeza de que la información del pedido y el número de tarjeta de crédito del cliente se haga de una manera segura.

Esto se podrá llevar a cabo mediante la utilización del protocolo SSL en una aplicación desarrollada en JAVA la cual utilizará el un protocolo basado en Zero Knowledge para la autentificación de las partes involucradas.

CAPÍTULO 2

2.1 INTRODUCCIÓN AL COMERCIO ELECTRÓNICO

Se puede definir al Comercio Electrónico como todo aquel proceso que implique por lo menos el envío de una transacción electrónica a través de una red pública o privada. Por lo que el Comercio Electrónico en Internet es uno de los puntos que se consideran como relevantes en la constante evolución que experimenta día a día la red de redes (INTERNET).

Debido a que los usos académicos, científicos e informativos en Internet son ciertamente desaprovechados y poco conocidos, se crea una oportunidad para los particulares y los empresarios, ya que por este medio se podrán obtener y ofertar productos sin tener que hacer desplazamientos innecesarios. Para los consumidores implica un servicio a domicilio y probablemente un ahorro en el precio. Para los empresarios, implica alcanzar un mercado global a un costo muy pequeño, la apertura de mercados inimaginables y la posibilidad de manejar con mayor eficiencia sus recursos e inventarios. Imaginemos que para una empresa, cada computadora personal o estación de trabajo se convierte en una sucursal virtual perteneciente a una cadena de más de 40 millones de establecimientos.

El Comercio Electrónico en Internet y en otras redes públicas necesita de ciertos criterios para satisfacer los siguientes puntos críticos:

1. La demanda de los consumidores para acceder de forma segura al comercio y otros servicios es muy alta.
2. Los comerciantes quieren métodos simples y de costo bajo para manejar las transacciones electrónicas.
3. Las instituciones financieras requieren de los desarrolladores de software soluciones competitivas en precio, manteniendo altos niveles de calidad.
4. Las sociedades de inversión y de medios de pago, administradoras de tarjetas y propietarios de marca necesitan diferenciar el comercio electrónico sin afectar significativamente sus infraestructuras.

Por lo cual varias compañías han dedicado sus esfuerzos a crear aplicaciones o software que cumpla con estos criterios.

2.1.1 SITUACIÓN ACTUAL DEL COMERCIO ELECTRÓNICO

Actualmente existe un gran interés hacia este tema animado por las perspectivas de negocio previstas para los próximos años. El pistoletazo de salida se produjo a raíz de un estudio publicado por la consultora Forrester Research, empresa especializada en estudios sobre Internet en el que vaticinan un crecimiento del comercio electrónico (e-commerce) de los 121 millones de dólares de 1997 a los 3800 en el 2002, solo en EE.UU. Este estudio ha sido confirmado por otros similares realizados por las más prestigiosas consultoras del mundo, entre ellas podemos citar a Andersen Consulting y Price Waterhouse.

La mayoría de las iniciativas actuales de venta por Internet se pueden calificar como fracaso relativo ya que los beneficios obtenidos por esta actividad económica está muy lejos de ser rentable y muy por debajo de otras actividades similares como la venta por catálogo. A pesar de todo, estas iniciativas son muy importantes desde el punto de vista estratégico sobre todo para las grandes empresas que deseen posicionarse y aprovecharse de las ventajas futuras de este tipo de negocios.

Algunas empresas informáticas como Dell o Cisco han conseguido resultados notables en la venta directa por Internet pudiendo abaratar los precios aprovechando la supresión de intermediarios, y creando un referente obligado al resto de empresas del sector.

Mención aparte merece la librería **Amazon.com**, que ha roto todas las previsiones y en unos pocos años ha pasado a convertirse en la mayor librería del mundo, vendiendo únicamente por Internet. Su fama ha crecido hasta límites increíbles y actualmente es el ejemplo utilizado por todos los que apuestan en esta tecnología como paradigma del éxito en el comercio electrónico por Internet. Como resultado de las expectativas puestas en Amazon.com, la empresa ha experimentado un incremento del 3000 % en su cotización en bolsa en el último año.

Merece también una mención aparte pero por motivos totalmente opuestos el proyecto E-Chritsmas'97-98, un ambicioso proyecto que agrupaba a varios países europeos (entre ellos España) y contaba con un equipo excepcional: Microsoft, HP, Banco Santander, etc.
El objetivo era ofrecer diferentes productos orientados a las ventas navideñas del 97-98, cuyo resultado fue un estrepitoso fracaso lo que puso de manifiesto que el mercado europeo no estaba suficientemente maduro por el momento para este tipo de actividades al contrario de lo que pronosticaban los más optimistas.

En México se empiezan a tener iniciativas para que el Banco de México se convierta en una Autoridad Certificadora, por la cual podrán conseguirse los certificados que se utilizarán para poder hacer las transacciones comerciales en México. Esta iniciativa dio origen al NOM-EM-157-1998, pero actualmente se encuentran todavía en etapa de desarrollo.

2.2 TIPOS DE RELACIONES

En función del tipo de entidades entre las que se realiza el negocio se pueden diferenciar varios tipos de comercios, que se diferencian tanto en los sistemas empleados como en los objetivos que persiguen.

2.2.1 RELACIONES BANCO-BANCO Y BANCO-CLIENTES

En términos generales podríamos incluir en la definición de Comercio Electrónico a las transacciones financieras entre los propios bancos. Actualmente se encuentran en un estado muy avanzado y se conocen con el nombre genérico de Transferencia Electrónica de Fondos (EFT). Todos estos sistemas tienen como objetivo la automatización de las operaciones financieras bancarias.

Otro objetivo importante es automatizar la relación con el cliente y los bancos. Para ello se avanza por dos vías:

1. La automatización de las oficinas mediante Cajeros Automáticos y en un futuro cercano mediante la Banca Electrónica por Internet, la cual ya va muy avanzada.

2. La automatización de los pagos corrientes mediante el uso de las Tarjetas de Crédito y los Terminales Punto de Venta (TPV). En ese punto también se está intentando automatizar los pagos de pequeño importe mediante la introducción de la Tarjeta Monedero, también llamada Tarjeta Chip, que no ha tenido gran demanda en México.

Tanto la Banca Electrónica como la Tarjeta Monedero son dos tecnologías estrechamente ligadas al Comercio Electrónico por Internet ya que utilizan las mismas técnicas de cifrado, y en algunos casos, pueden llegar a ser complementarias.

2.2.1.1 RELACIONES EMPRESA-EMPRESA

El objetivo primordial a este nivel es la automatización de la gestión de las facturas y la eliminación de sus costos asociados. Según algunos estudios publicados, la eliminación de estos costos permitiría duplicar o triplicar los beneficios de la mayoría de las grandes empresas, por ello supone un gran atractivo para cualquier gran organización. La principal dificultad que conlleva la aplicación de estas tecnologías es que tanto los proveedores como los clientes de la empresa deben utilizarla, y no siempre resulta posible debido a la gran inversión que ello requiere.

EDI (Intercambio Electrónico de Datos) es un sistema ideado para automatizar la gestión de cobros, ventas y facturas entre empresas que ha tenido una fuerte base en el mundo de los

negocios, en los últimos años. Con la llegada de Internet esta tecnología ha quedado obsoleta ya que no funciona bajo el estándar TCP/IP y por lo tanto debe utilizar su propio sistema de comunicaciones. A pesar de todo, los empresarios son reacios a modificar sus sistemas ya que en la mayoría de los casos EDI ha supuesto una inversión que todavía no ha podido ser rentable. Como respuesta a esta demanda han aparecido una serie de soluciones que encapsulan EDI en TCP/IP, lo que permite su utilización por Internet. Estas soluciones reciben el nombre genérico de EDIWeb.

2.2.1.2 RELACIONES EMPRESA-CLIENTE

Este es el tipo de relaciones a la que generalmente se refiere cuando se habla de Comercio Electrónico. La utilización de las nuevas tecnologías admite, en teoría, un contacto directo entre fabricantes y consumidores, lo que permitiría la eliminación de intermediarios en el proceso de compra. Esto repercutiría enormemente en el precio final del producto pudiendo ofrecer precios muchos más bajos.

La venta directa a través de Internet es una actividad que espera mover un volumen de negocio muy importante en los próximos años. Internet es sólo el primer paso hacia un nuevo concepto de economía en el que los consumidores podrán adquirir bienes desde sus casas sin necesidad de desplazarse a una tienda concreta. La Televisión por cable permitirá generalizar este tipo de negocio al llegar a un número mayor de consumidores potenciales. Uno de los puntos críticos que deben resolverse para garantizar las compras por medios electrónicos como Internet es la creación de un sistema de pago electrónico que permita realizar pagos seguros utilizando sistemas de comunicación inseguros. Este es el punto en el que se centra este proyecto: La seguridad en los pagos electrónicos por Internet.

En las ventas electrónicas podemos distinguir dos grandes tipos en función del tipo de producto que se comercialice:

Información digital y productos físicos

En el primer caso tendríamos los productos susceptibles de ser digitalizados y enviados por una red de comunicación de datos, como la música, videos, software, documentación, etc. En este caso la compra se podrá realizar en su totalidad por una red como Internet sin necesidad de utilizar otro tipo de medio físico.

Si el producto requiere un traslado físico al domicilio del cliente, es obvio que la compra no se podrá realizar enteramente por Internet, teniendo que recurrir a empresas de transporte o mensajería para realizar el envío. En este caso la utilidad de la transacción electrónica se reduce a simplificar el proceso de compra ya que existen otros medios de pago como el pago contraentrega o el giro postal.

2.3 MODELOS PARA COMERCIO ELECTRONICO EN REDES DE INTERCONEXIÓN.

Para poder identificar el tipo modelo de comercio electrónico que se utiliza es necesario conocer las relaciones que se establecen entre las partes en la transacción electrónica. Es decir, el tipo de interacción que existe entre ellas cuando se establece la relación dentro del comercio electrónico. Para poder explicar esto más a detalle a continuación describiré los principales modelos de comercio electrónico que existen.

2.3.1 MODELO ABIERTO

Un modelo abierto involucra el uso de una autentificación electrónica entre los usuarios sin que ellos tengan de antemano un convenio, una relación comercial previamente arreglada o formalizada por un autentificador en particular. Típicamente los modelos abiertos se dan cuando un usuario establece un contrato basado en una tercera parte para el intercambio de las autentificaciones electrónicas validadas donde necesariamente se debe referenciar a un servicio ofrecido por un proveedor de servicio de autentificación. En este caso las partes son entidades legales independientes a pesar de que puede existir una relación legal entre una de las partes y el proveedor de autentificación. La ventaja de este modelo es que permite tener comercio con un ilimitado número de clientes. Si se empiezan a tomar consideraciones como la viabilidad financiera o la consideración de una entrega segura, esto a su vez limita y reduce la libertad del modelo abierto.

Figura [1] Modelo Abierto

2.3.2 MODELO CERRADO

Un modelo cerrado es donde las autentificaciones son intercambiadas entre los usuarios que han tenido un convenio contractual previo o una relación organizacional. Este tipo de modelos se reducen a ser utilizados dentro de las grandes organizaciones y/o arreglos hechos previamente con clientes y proveedores fijos. El ejemplo, más descriptivo de este modelo es el EDI, el cual tiene redes privadas para poder hacer transacciones comerciales entre proveedores y clientes de una manera segura y directa. La principal ventaja de este modelo es que el proveedor conoce plenamente al cliente y no duda de los convenios establecidos.

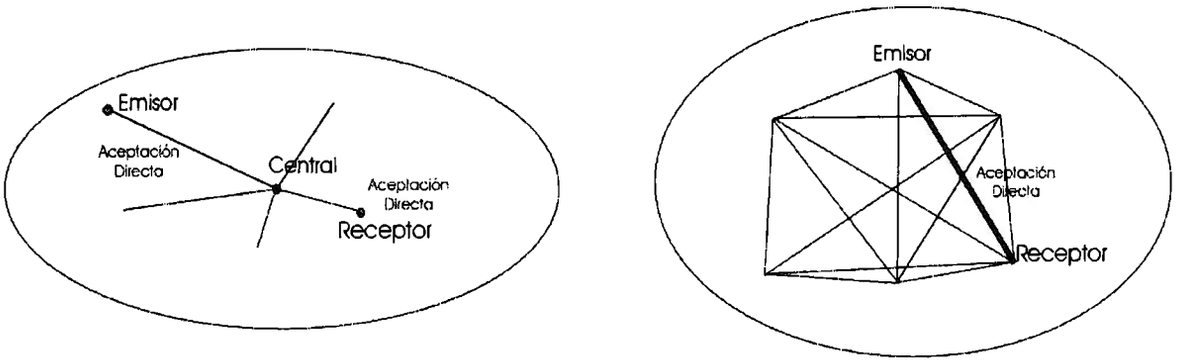

Figura [2] Modelo Cerrado

2.3.3 MODELO ABIERTO PERO ACOTADO

En este modelo múltiples partes pueden relacionarse pero necesitan de un autentificador para limitar el número de posibles relaciones, y de esta manera conocer a las partes con las cuales se establecen las relaciones comerciales.

Con este modelo es necesario que las partes estén coordinadas para aceptar uno o más proveedores de servicio de autentificación.

Un ejemplo de esto puede ser el utilizado por el Gobierno Australiano con la utilización del Project GateKeeper, en el cual la oficina gubernamental decide cuáles de sus clientes pueden utilizar un autentificador simple de cualquier proveedor de servicios certificado.

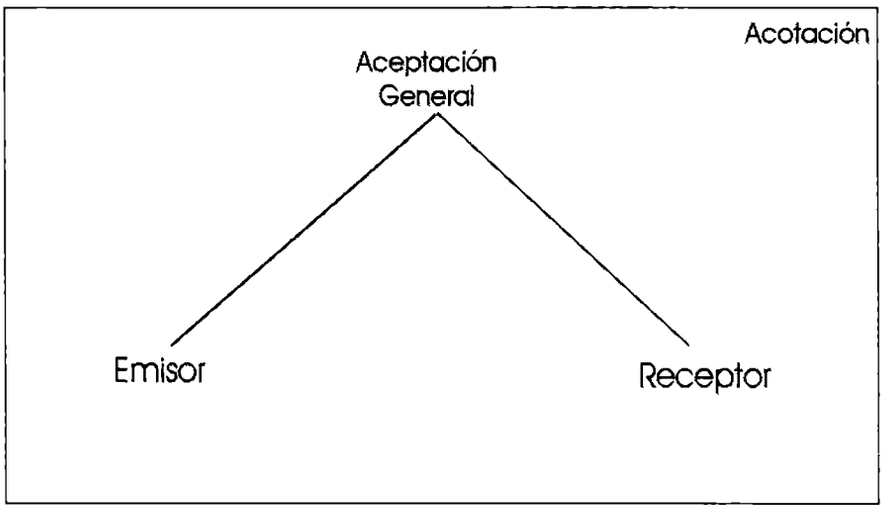

Figura [3] Modelo Abierto pero Acotado

2.3.4 MODELO CERRADO Y ACOTADO

En este modelo se permiten solo tener relaciones entre los integrantes de cierto dominio, a pesar de que ya se encuentren dentro en un modelo cerrado. Este modelo se puede utilizar cuando dentro de una red privada se requiere que algunos de los clientes pueden establecer una relación de negocio con algunos proveedores de la red privada, dependiendo de la clasificación del cliente. La principal ventaja de este modelo es que se puede establecer niveles de comercio en el cual exista una clasificación de las relaciones posibles dentro de la red privada.

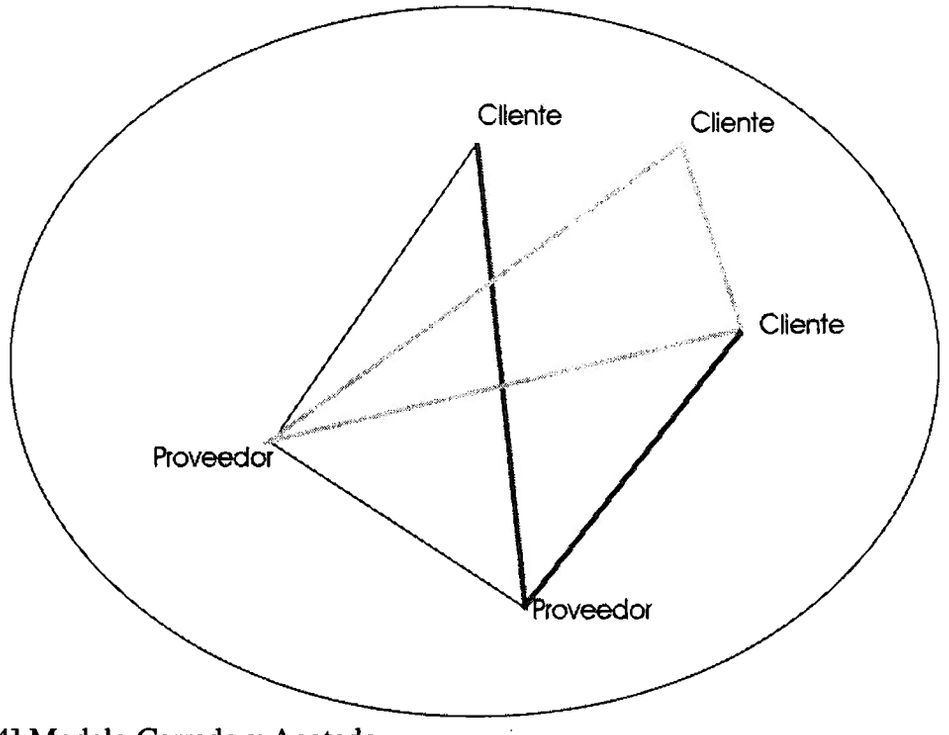

Figura [4] Modelo Cerrado y Acotado

2.4 TIPOS DE PAGO

Los dos sistemas clásicos de pago no efectivo por excelencia son los cheques convencionales y las tarjetas de crédito. En el sistema financiero, el dinero toma forma de entrada en los libros de los bancos y de otras instituciones financieras. Esto para el usuario se refleja en la existencia de una cuenta depósito en la que se graban los depósitos del cliente y se realizan pagos en forma de cheques o transferencias.

Un cheque es un documento escrito por el usuario de la cuenta y autentificado por el mismo que se entrega a un comerciante, quien a su vez lo acepta antes de presentarlo en su banco. Si el banco cobrador y el del vendedor son los mismos, todo se reduce a una compensación interna de cuentas. Si por el contrario, ambos disponen de cuentas en bancos distintos, el banco cobrador deberá presentar el cheque y recoger los fondos a través de un sistema de ajuste o cámara de compensación ACH (Automatic Clearing House). Esta cámara puede pertenecer al estado, por ejemplo, al Banco Nacional de México o tratarse de un sistema intermediario privado, como PROSA. En el momento de la entrega del cheque, el cobrador no sabe si éste se encuentra respaldado por fondos por lo que corre cierto riesgo. Pero los pagadores también asumen determinados riesgos frente a la posible existencia de cheques falsos ya que reciben los extractos de las cuentas después de pagar sin confirmación previa al pago.

Otra alternativa es la utilización de un sistema de crédito, como una tarjeta o un servicio bancario de adelantos.. El comerciante se asegura el pago y el emisor asume la responsabilidad del cobro. El ajuste se realiza entonces cuando el comerciante envía un lote de autorizaciones a su banco entre este y el banco emisor de la tarjeta.

Existen dos tipos de operaciones con tarjeta de crédito que suponen distinto balance de riesgo en función de sí la tarjeta se encuentra presente o no.

En el primer caso, la tarjeta presente, todo el riesgo del crédito pasa al comerciante. En estos casos el comerciante puede verificar la firma del titular comparando la que éste imprime en el recibo con la que aparece en la tarjeta. Por ello, será responsabilidad suya si se hace uso fraudulento de la tarjeta de un tercero. La integridad de la transacción queda garantizada para el cliente mediante una copia del recibo. El nombre de la cuenta se autentifica a través del número de la tarjeta y la transacción puede ser confirmada enviando los datos por medio de una red privada de la asociación.

En el segundo caso, la tarjeta no presente, el consumidor asume cierta parte (pequeña) del riesgo ante un posible fraude ya que es responsabilidad suya proteger el número de tarjeta, debido a que este es el único medio de identificación del pagador en la transacción. Este es el sistema

empleado en los pedidos por teléfono o Internet (en pedidos como correo o fax, aún se cuenta con la firma del titular como medio de aceptación del pedido, pero no de autentificación). Por ello, dado que nada impide que se pueda colocar un sniffer (programa que monitorea todos los datos que pasan por una determinada computadora) en la red, los números de tarjeta deben viajar cifrados por la red.

Hay diferencia entre el uso de cheques y de tarjetas. Un pago mediante cheques supone dinero pagado para el consumidor, que asumirá todos los inconvenientes derivados de una falla del comerciante. Sin embargo, si se paga con tarjeta se puede pedir la restitución del importe. El problema pasa a ser del banco emisor de la tarjeta, que deberá pedir cuentas al del comerciante.

2.5 SISTEMAS PARA EL PAGO EN INTERNET

El crecimiento de Internet ha permitido crear una nueva vía de comunicación entre comerciantes y compradores potenciales, lo que ha propiciado la aparición de diferentes sistemas de pago electrónico. El evidente riesgo de fraude condiciona totalmente estos sistemas por ello ha sido necesario aplicar las más modernas técnicas de cifrado para garantizar la seguridad.

Hoy en día el protocolo utilizado para el pago con tarjeta de crédito por Internet es el SSL (Secure Socket Layer) aunque no es el único ni el primero que se ha utilizado. Su éxito se debe a la gran seguridad que aporta y a la facilidad de utilización ya que es el único de los protocolos estandarizado. A continuación vamos a explicar los sistemas más conocidos creados para el pago por Internet.

2.5.1 FSTC (Financial Services Technology Consortium)

FSTC [37] es un consorcio americano de bancos, organizaciones gubernamentales y empresas tecnológicas creado en 1995. Uno de los proyectos promovidos por este consorcio es la creación de un sistema de cobro de Cheques Electrónicos.

El pagador debe contar con un procesador seguro, que se implementa en forma de tarjeta inteligente. Este procesador es el encargado de generar los cheques que consisten, simplemente en ordenes de pago firmadas digitalmente. Los cheques se trasmiten al comerciante que los acepta firmándolos digitalmente y los envía al banco, que los hará efectivos a través de la una red ACH clásica.

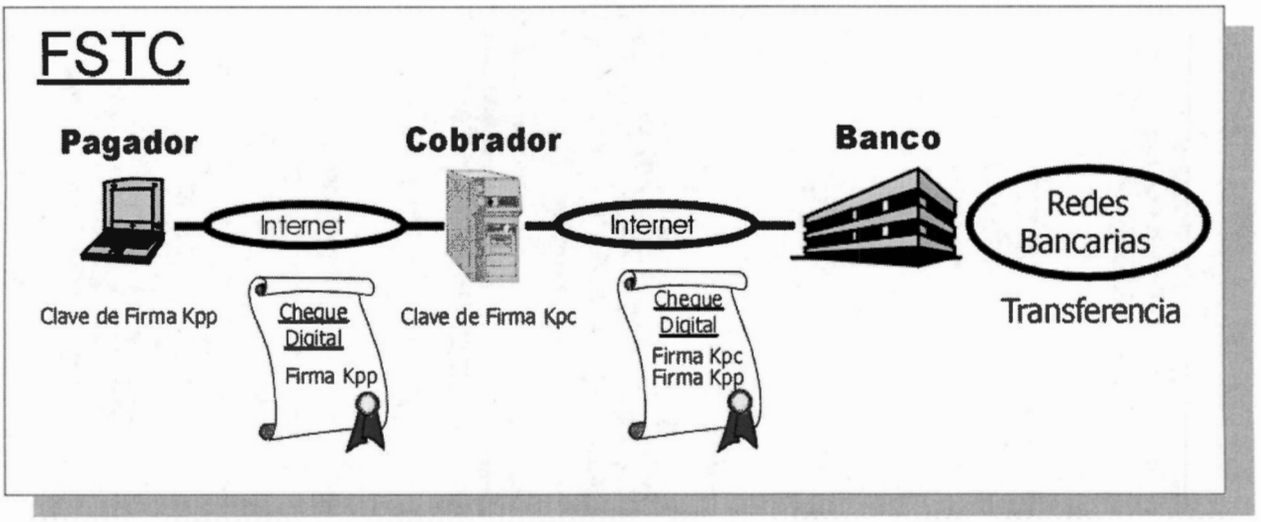

Figura [5] Transacción del FSTC

2.5.2 CHECKFREE

Checkfree Corporation [33] es una entidad financiera americana que lleva realizando transacciones electrónicas y cobros con tarjetas de crédito a distancia desde 1983. Sus clientes son tanto empresas como particulares que deben poseer una cuenta en la entidad. Para realizar un pago electrónico, el usuario debe conectarse vía módem (sin pasar por Internet) a Checkfree y enviar una orden de pago. Dependiendo del tipo de pago, Checkfree utilizará la Reserva Federal de EUA o el sistema RPS de MasterCard para realizar la transferencia electrónica de fondos.
Para el pago por Internet, Checkfree ha decidido utilizar la tecnología de Cybercash de cifrado y autorización.

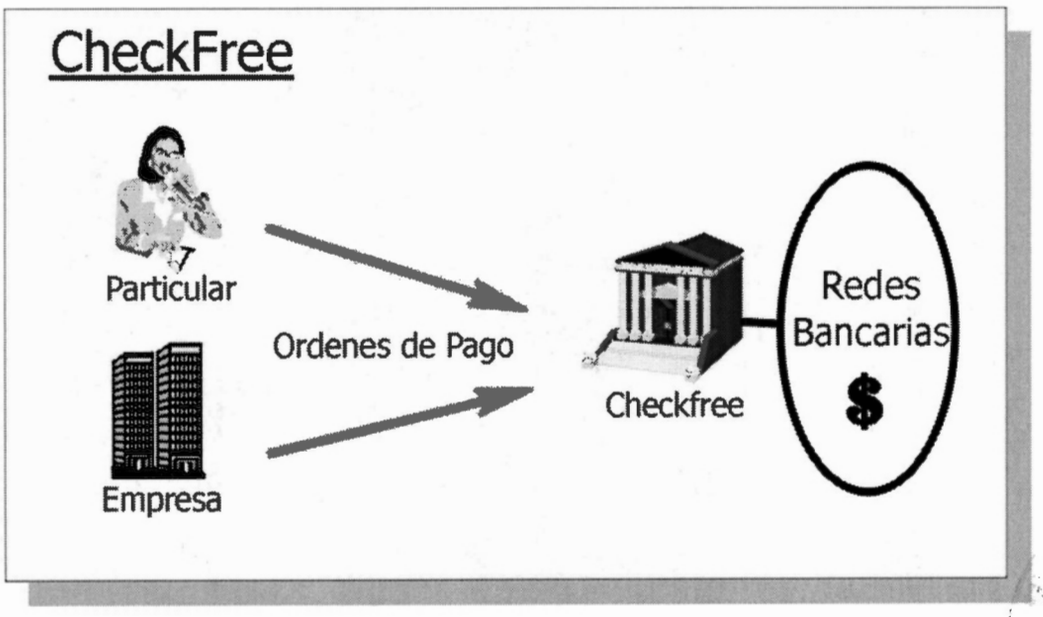

Figura [6] Funcionamiento de CheckFree

2.5.3 FIRST VIRTUAL (FV)

First Virtual [28] fue uno de los primeros en aparecer (1994) con la peculiaridad de no hacer uso de la criptografía. Un consumidor que desee hacer uso del sistema primero debe registrarse en el mismo, tras lo que obtiene un Virtual PIN, ID o identificador de First Virtual. Para ello debe de enviar un número de tarjeta de crédito, VISA o MasterCard, por un medio fuera de línea, como teléfono o fax. Una vez registrado podrá realizar compras por Internet o acceder a información restringida en un servidor que emplee el sistema FV.

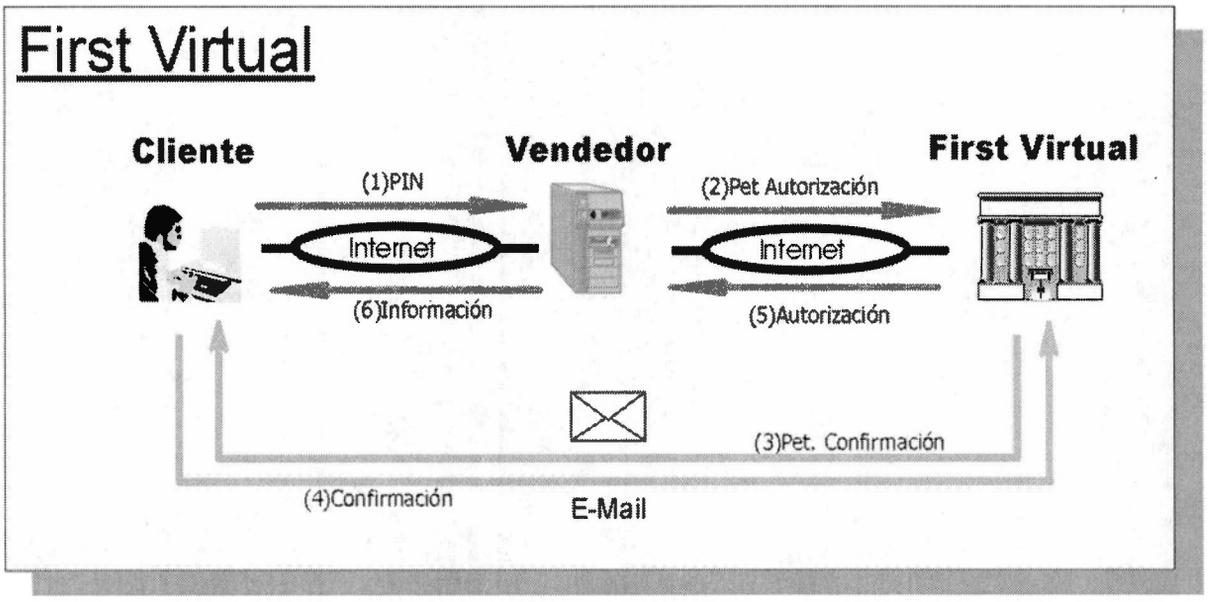

Figura [7] Funcionamiento de First Virtual.

Tras registrarse, se recibe un correo electrónico con una clave de doce dígitos, y un número de teléfono donde confirmarlo. Mantener un PIN cuesta 5 dólares al año aproximadamente.

Para realizar un pago bastará con presentar el Virtual PIN al comerciante. Éste se conectará a un servidor FV para comprobar si el pago es correcto, y de ser así, enviará la mercancía o dará paso libre a la información.

Para realizar un pago al comerciante, First Virtual, envía un correo electrónico al consumidor en el que le indica la operación y le pregunta si desea pagar o no. El consumidor deberá confirmar el pago respondiendo "yes", "no" o "fraud". Mediante este sistema el comerciante no llega nunca a saber el número de tarjeta del consumidor, ni ésta llega nunca a viajar por la red.

El protocolo de validación de FV es él más lento de todos los medios de pagos electrónicos al requerir una confirmación del usuario fuera de línea. No hace uso de la criptografía y permite el anonimato del usuario frente al comerciante.

2.5.4 NETMARKET

Netmarket [34] se trata de un sencillo sistema que permite el pago electrónico por Internet utilizando como seguridad el sistema Pretty Good Privacy (PGP) para cifrar los números de tarjetas de crédito manteniendo la confidencialidad de los datos.

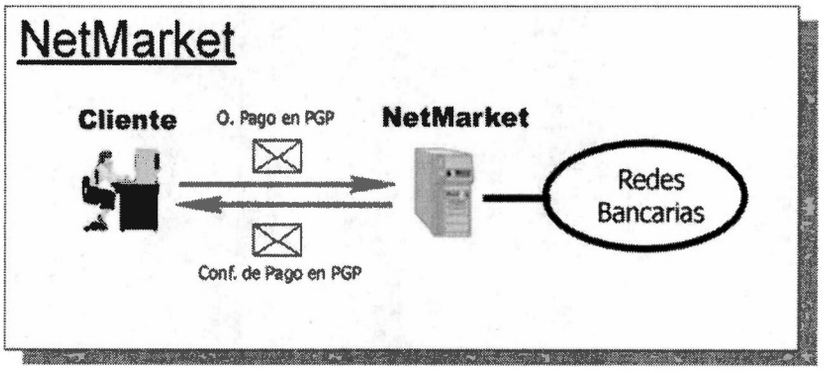

Figura [8] Funcionamiento de Netmarket.

2.5.5 NETBILL

NetBill [31] es una alianza entre la Universidad Carnegie Mellon y Visa Internacional con el objetivo de ofrecer un sistema de pago seguro a través de Internet. NetBill se centra exclusivamente en la venta de información digital que pueda ser enviada por medios electrónicos como artículos de prensa, software, música, videos o cualquier tipo de documentación electrónica.

NetBill es un sistema prepago, es decir, todos los usuarios deben crear una cuenta antes de realizar cualquier compra. El usuario selecciona la información que desea y la recoge del servidor a cambio envía una orden de pago al Vendedor. Éste remite la orden de pago al Servidor NetBill que deberá autorizar la compra realizando una transferencia de la cuenta del comprador a la del comerciante. Tanto la orden de pago como la información adquirida se encuentran cifradas. El Vendedor desconoce los datos de la orden de pago y el Comprador no puede utilizar la información sin conocer la Clave de cifrado. Tras aceptar la compra, el Servidor NetBill realiza la transferencia y envía la Clave de cifrado al usuario que le permitirá descifrar los datos adquiridos.

NetBill funcionó en periodo de pruebas el verano de 1995 y permitió el acceso a diferentes servicios universitarios de la Universidad.

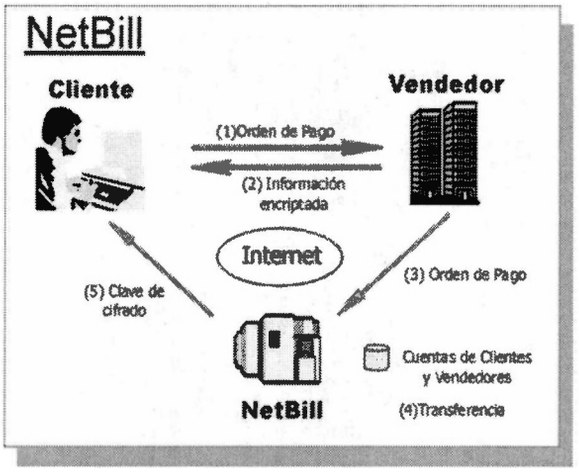

Figura [9] Funcionamiento de NetBill

2.5.6 NETCASH Y NETCHEQUE

Sistema creado en la Universidad de Sur California (USC) en 1997. Se trata de un sistema de pago seguro por Internet que permite varias formas como E-cash o como cheque digital bajo el mismo protocolo. Netcash [32] es similar a Digicash y permite el pago con dinero electrónico de forma que mantiene el anonimato del comprador. Por su parte Netcheque sigue un esquema típico de pago con tarjeta de crédito como Cybercash. Ambos sistemas utilizan el mismo servidor para validar los pagos.

La seguridad del protocolo se basa en Kerberos, y la firma digital empleada es un ticket especial llamado proxy. Si se desea se puede cambiar la criptografía a un modelo de clave pública, pero disminuirá el rendimiento.

Para escribir un cheque, el usuario especifica los datos, y el software cliente obtiene un ticket Kerberos para esta operación, genera un autentificador para una suma de control sobre la información del cheque, y coloca el ticket en el campo firma del cheque. El cheque se codifica entonces en Base-64 y se envía al destinatario. Al recibirlo, el software del vendedor lee la parte en claro del mismo, extrae el ticket Kerberos y lo envía a través de una conexión cifrada al servidor Netcheque que valida la operación.

2.5.7 CYBERCASH

Es uno de los medios de pago electrónicos pioneros. La empresa fue creada en 1994 y el sistema CyberCash [29] se encuentra operativo desde abril de 1995. Se trata de un sistema integrado con el WWW que utiliza un protocolo propio manejado por un software que debe ser distribuido tanto a comerciantes como a consumidores. Constituye un puente entre Internet y las redes de autorización de emisores de tarjetas, contando para ello con la experiencia ganada con el sistema Verifone de autorización de tarjetas por teléfono, del cual deriva. El consumidor cuenta con un software "wallet" o monedero que puede ligar a varias cuentas bancarias o a las tarjetas de

crédito. El software cifra todos los datos, realiza un registro de todas las transacciones y se encuentra protegido mediante contraseña. En el lado del comerciante se sitúa un software similar.

Cuando un usuario baja el software de CyberCash, genera un par de claves una pública, Kp, y privada, Ks para él. El sistema empleado es RSA de 1024 bits. A continuación envía Kp al servidor de CyberCash que almacena en una base de datos. De esta forma sólo CyberCash sabe las claves públicas de todos los interlocutores posibles. La comunicación entre consumidor y comerciante se lleva a cabo en claro, mientras que la comunicación de estos con CyberCash se realiza de forma protegida. Para ello se emplea una clave de sesión DES aleatoria de 56 bits que se distribuye encriptada con la llave pública del interlocutor. CyberCash tiene incrustada su propia llave pública en el software por lo que cualquiera puede comunicarse con él. Un dato importante es que la clave DES es de 56 bits cuando por las leyes de la legislación americana antes fijaban un máximo de 40 bits, y es que CyberCash consiguió un permiso especial.

La transacción se lleva a cabo de la siguiente manera:

El cliente selecciona un ítem a adquirir mediante www y elabora un pedido.

El software del comerciante envía "la factura proforma" al software "wallet". La factura proforma es texto en claro, firmado digitalmente en el que figura una descripción de la compra así como las tarjetas de crédito aceptadas. Para la realización de firmas digitales se emplean funciones hash MD5 y claves secretas RSA. A su recepción, el software "wallet" da a elegir al usuario entre aquellas tarjetas que tiene registradas y le pide autorización para realizar el pago.

Previa confirmación del usuario, el software "wallet" del cliente genera un mensaje de pago y lo envía la comerciante. Este mensaje consiste en un hashing de la factura junto con las instrucciones de pago, todo ello firmado digitalmente y cifrado para CyberCash.

A la recepción del mensaje, el comerciante, que no puede descifrarlo simplemente añade la información al pedido (también firmada digitalmente) y lo remite a CyberCash.

CyberCash descifra y compara los dos mensajes. Si coinciden los datos, solicita confirmación, a través de la red financiera y trasmite la respuesta al comerciante para que éste pueda cerrar la transacción con el cliente.

Todo este proceso se lleva a cabo en un periodo de tiempo inferior a un minuto. El proceso descrito es el caso más habitual, no obstante, el protocolo CyberCash es mucho más complejo y contempla la posibilidad de reintegros, anulaciones y solicitud de estado.

Respecto a la protección del consumidor cabe destacar que, al cifrarse la orden de pago para CyberCash y no para el comerciante este no ve el número de la tarjeta de crédito lo que unido al

hecho de que se comprueba la descripción de la compra, impide el abuso por parte de los comerciantes. En lo referente a la privacidad, aunque CyberCash tiene acceso a la descripción de la compra, ésta no tiene porque incluir los detalles de la compra, puede constar como dato la referencia y el tipo solamente. De este modo el comerciante no ve la tarjeta y CyberCash no ve el producto

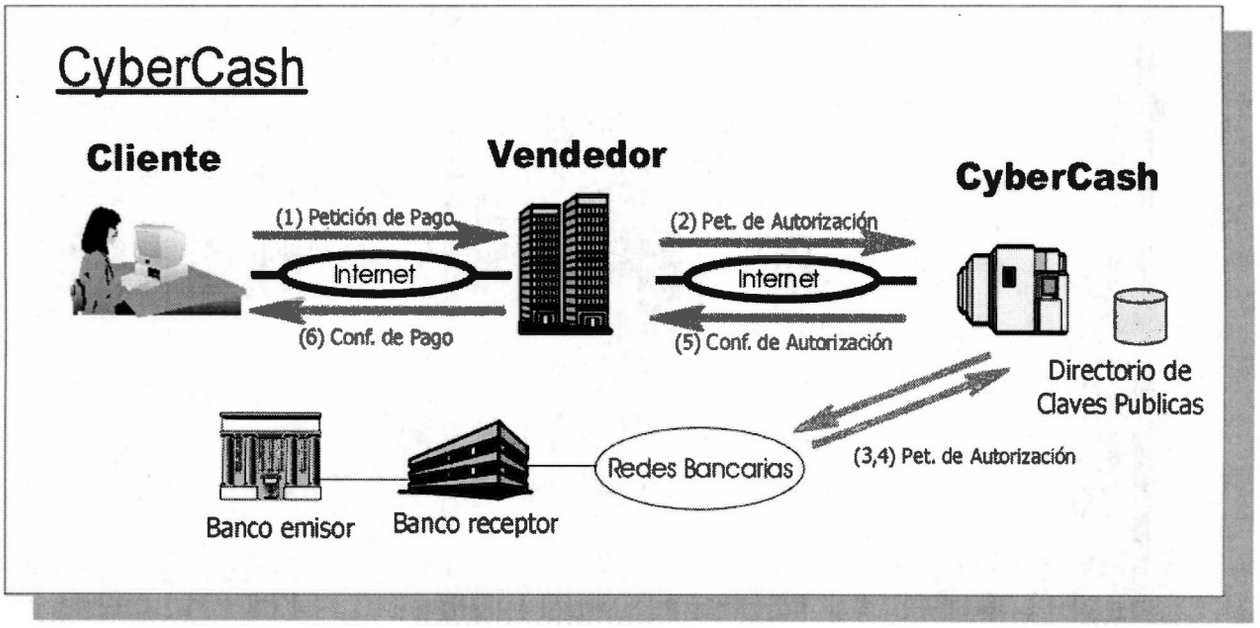

Figura [10] Funcionamiento de Cybercash

¿CUÁL ES LA DIFERENCIA ENTRE LA BILLETERA DE CYBERCASH Y LA DE MICROSOFT?

A pesar de que ambas son similares en muchos aspectos, la billetera de CyberCash y la de Microsoft son entidades diferentes. Ambas son programas clientes, y ambas soportan transacciones de tarjetas de crédito en línea. Próximamente, ambas podrán soportar el Pago por medio de Cheques Electrónicos.

Ambas billeteras son seguras. La billetera de Microsoft utiliza SSL para el pago por tarjeta de crédito, mientras que todos los módulos de CyberCash están hechos utilizando el método de cifrado y autentificación de CyberCash, él cual es 768-bit RSA/56-bit DES.

2.5.8 DIGICASH

Fundada en 1990 en Amsterdam por el prestigioso Criptógrafo David Chaum, DigiCash [30] ha sido una de las empresas que más han aportado al concepto de dinero digital. La diferencia con CyberCash, radica en que es un sistema de pago anticipado, donde se adquiere previamente el dinero del banco y se almacena digitalmente en el software del comprador. Este sistema permite la compra anónima ya que no requiere autentificación.

El usuario abre una cuenta en DigiCash y a cambio recibe una lista de números de 64 bits que equivalen a diferentes cantidades de dinero electrónico (E-Cash). Para pagar únicamente debe enviar uno de estos números al Vendedor. Una vez recibidos, se remitirán al Servidor DigiCash que realizará la transferencia manteniendo el anonimato del comprador. DigiCash es el equivalente electrónico a los Cheques de Viaje.

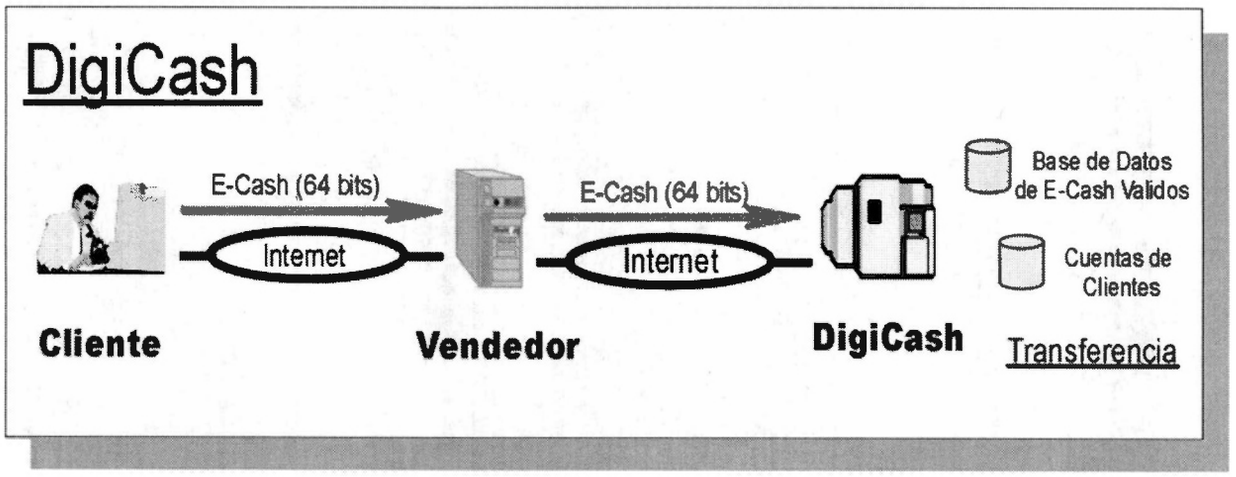

Figura [11] Funcionamiento de DigiCash

2.5.9 MONDEX INTERNATIONAL

Mondex [26] es una asociación de bancos creada en 1995 en Gran Bretaña y destinada a promover el uso de dinero electrónico usando como soporte básico las Tarjetas Inteligentes o Tarjetas Chip. El E-Cash puede ser intercambiado por cliente y comerciantes siempre que posean los medios tecnológicos necesarios compatibles con la tecnología Mondex. Las Tarjetas pueden ser utilizadas a su vez en cajeros automáticos (ATM) o incluso en pagos a distancia por Internet.

2.5.10 OPEN MARKET

Open Market [35] fue fundada en 1994 y es una de las empresas pioneras en el comercio electrónico por Internet. Su contribución más importante no se debe a la creación de nuevos protocolos de seguridad sino a StoreBuilder, un software destinadas a la creación de Tiendas Virtuales y CyberMalls (Grandes almacenes virtuales) en Internet bajo un entorno Web. Los medios de pago han ido adaptándose a medida que han ido apareciendo nuevos medios de pago electrónicos. Primeramente Open Market funcionó permitiendo el pago con tarjeta de crédito con comunicaciones no cifradas. Más tarde se utilizó la tecnología de CyberCash y finalmente adoptó SSL.

2.5.11 ¿QUÉ ES E-BUSINESS DE IBM?

E-Business [25] es un concepto de comercio electrónico creado por IBM, para dar impulso a las empresas que deseen incursionar en la aventura de vender sus productos en Internet. IBM define a E-business como:

"Lo que sucede cuando usted combina el amplio alcance de Internet con los vastos recursos de los sistemas de computación tradicionales"

Es dinámico e interactivo. Su alcance es muy amplio desde intranets privados, hasta el Internet público, pasando por extranets compartidos. Utiliza a Internet para unir a clientes, vendedores, proveedores y empleados de una manera nunca antes posible. En resumen, e-business conecta eficientemente la información valiosa con la gente que la necesita"

Como comentario al respecto su definición no es otra cosa que las bases para crear un mercado para el comercio electrónico.

IBM considera que E-business es una mejor manera de hacer negocios, basando esto en controlar los procesos esenciales de cada negocio a Internet, mejorando el servicio al cliente y reduciendo los ciclos de producción, para obtener más resultados con recursos limitados, e incluso vender sus productos.

Ejemplo de esto son algunas compañías financieras las cuales han creado sitios Web seguros e interactivos donde los clientes pueden comprar, hacer investigación de mercado y obtener respuestas a sus preguntas. Otro ejemplo es un mayorista europeo Supervox Groupe que cuenta con un catálogo de 8000 piezas, creándole una cadena de suministro automatizada que atiende a todo un nuevo segmento del mercado.

Y como resultado, la compañía Supervox Groupe espera un aumento de 8 millones de dólares en sus ganancias anuales.

2.5.12 MICROSOFT WALLET

Es una aplicación desarrollada por Microsoft [26] para incorporarla a sus sitios de comercio electrónico, esta aplicación corre localmente.

Microsoft wallet [49] está incluida con el navegador Internet Explorer, en este lugar se puede guardar información privada, como los números de tarjetas de crédito, información de su tarjeta para cajeros automáticos, certificados digitales, y más. Esta información es almacenada localmente en su computadora y sólo la podrá utilizar a través de su contraseña.

Las ventajas que ofrece esta Wallet o billetera electrónica son:

- Seguridad. Su información se guarda con seguridad en su computadora, y nadie tiene acceso si no conoce la contraseña con la cual fue almacenada.
- Completo control. Usted decide qué información debe estar en su Billetera Electrónica y puede decidir quién tiene acceso.
- Universal, ya que varios sitios soportan Microsoft Wallet [49].
- Fácil de usar. Usar Microsoft Wallet significa que no se va ha introducir la información confidencial cada vez que se haga un pedido.
- Extensible. La billetera está diseñada para soportar varios métodos de pago, por lo que si un nuevo método de pago surge es muy posible que se integre fácilmente a la Billetera.

2.5.13 E-COMM

El Consorcio e-COMM [14] fue fundado en Julio de 1996. Este está formado por 6 socios: tres bancos –BNP, Société Générale y Crédit Lyonnais, una organización de pago internacional – Visa Internacional, un banco de manufactura de tarjetas – Gemplus y un operador de comunicaciones – France Télécom.

El objetivo de e-COMM es el desarrollar y probar un método seguro para el pago por tarjeta de crédito utilizando la red Internet. El consorcio se ha enfocado a desarrollar un sistema combinando tarjetas inteligentes y el estándar SET internacional desarrollado por VISA [16] y MasterCard [17]. Durante la primera fase del proyecto, el sistema es piloteado utilizando tarjetas francesas.

Durante la segunda fase se incluirán tarjetas inteligentes de formato internacional, basándose en el estándar EMV, el cual es capaz de manejar funciones criptográficas.

A este sistema se le ha dado un impulso muy importante en Francia, por lo que en la actualidad se pueden hacer compras en ese país por este medio. Lo importante de este sistema es la utilización del SET, que es el estándar del futuro de las transacciones comerciales.

2.6 CONCLUSIONES

De todos estos sistemas de pago se pueden extraer varias conclusiones de cómo debería ser el sistema de pago ideal. Las dos condiciones más importantes son:

El sistema de pago debe ser un estándar único ya que en caso contrario se limita enormemente el número de posibles usuarios del sistema. Debe permitir conexiones seguras a través de redes inseguras como Internet. Esto puede conseguirse fácilmente con la utilización de la criptografía moderna.

Partiendo de estas dos premisas, las organizaciones y empresas responsables de Internet se esforzaron por crear un protocolo único que permitiese confidencialidad en las comunicaciones por Internet. La solución fue adoptar el protocolo SSL de Netscape (que se explicará en detalle más adelante) como uno de los protocolos oficiales incluyéndolo en los estándares TCP/IP.

Otra conclusión importante es la comprobación de que el método ideal para el pago electrónico, por distintas razones, es la utilización de tarjetas de crédito. Y para ello es necesario la creación de Pasarelas de Pago que permitan la interconexión entre Internet y las Redes Bancarias. Bajo esta perspectiva, el protocolo más perfeccionado de los mencionados es el de CyberCash, protocolo que sirvió de base para el diseño del actual sistema SET.

Por su parte DigiCash y Mondex presentan un concepto nuevo, el Dinero electrónico (E-Cash). En estos casos no se trata de simples ajustes bancarios, sino que el dinero se encuentra "físicamente" en formato electrónico. Serían la versión electrónica de las monedas y billetes. Las posibilidades de fraude en estos caso se disparan dada facilidad para reproducir una información digitalizada. Por ello el avance del E-Cash será más lento que el pago electrónico y requerirá de sistemas todavía más perfeccionados de seguridad y validación.

CAPÍTULO 3

3.1 INTRODUCCIÓN A LA CRIPTOLOGÍA

La Criptología [6] (del griego criptos = oculto y logos = tratado, ciencia) es el nombre genérico con el que se designan dos disciplinas opuestas y a la vez complementarias: Criptografía y Criptoanálisis. La Criptografía se ocupa del diseño de procedimientos para cifrar, es decir, para enmascarar una determinada información de carácter confidencial. El Criptoanálisis, por su parte, se ocupa de romper esos procedimientos de cifrado para así recuperar la información original. Ambas disciplinas siempre se han desarrollado de forma paralela, pues cualquier método de cifrado lleva siempre emparejado su Criptoanálisis correspondiente.

La Criptografía como medio de proteger la información personal es un arte tan antiguo como la propia escritura. Como tal, permaneció durante siglos relacionada a los grupos militares y diplomáticos, ya que la protección de la información que se utiliza en estos grupos la hacen necesaria.

En la actualidad esto ha cambiado, debido al desarrollo de las comunicaciones y telecomunicaciones, hoy en día cualquier persona puede hacer uso de ella debido a que necesita transmitir información a través de computadoras. Esto es necesario para poder garantizar la privacidad de las personas.

El esquema fundamental de un proceso criptográfico (cifrado/descifrado) puede resumirse del modo en que se muestra en la siguiente figura:

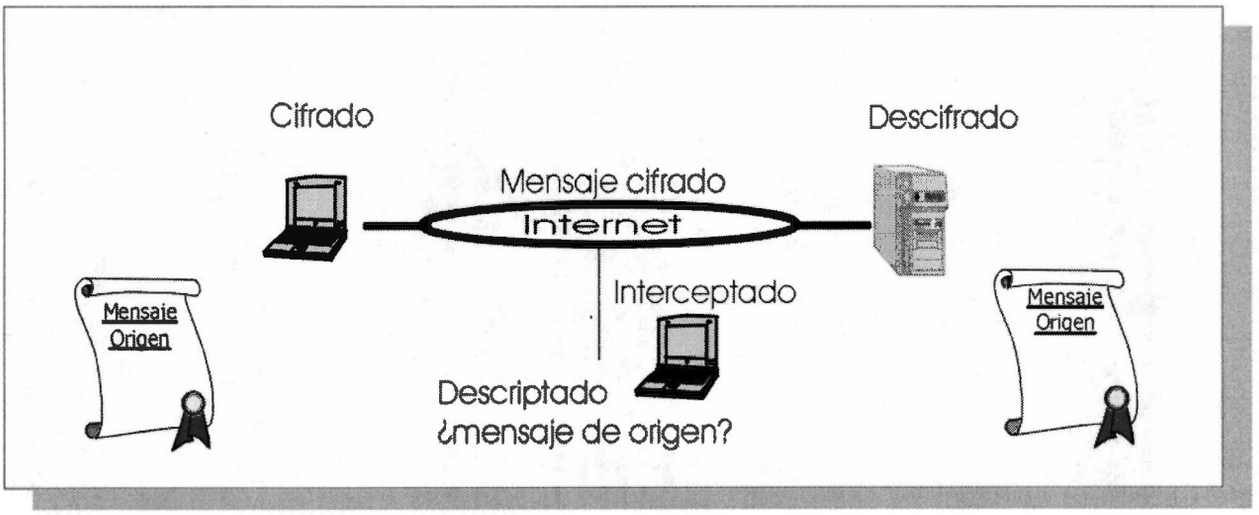

Figura [12] Funcionamiento del Proceso Criptográfico

Juan y Pedro son, respectivamente, el emisor y el receptor de un determinado mensaje. Juan transforma el mensaje original (texto claro o texto fuente), mediante un determinado procedimiento de cifrado controlado por una clave, en un mensaje cifrado (criptograma) que se envía por un canal público (por ejemplo: Internet). En la recepción, Pedro con conocimiento de la clave transforma ese criptograma en el texto fuente, recuperando así la información original.

En el proceso de transmisión, el criptograma puede ser interceptado por un enemigo criptoanalista que lleva a cabo una labor de desencriptado; es decir, intenta a partir del criptograma y sin conocimiento de la clave, recuperar el mensaje original. Un buen sistema criptográfico será, por tanto, aquel que ofrezca un descifrado sencillo pero un desencriptado imposible o, en su defecto, muy difícil.

La finalidad de la Criptografía es doble: por un lado, la de mantener la confidencialidad del mensaje; es decir, que la información allí contenida permanezca secreta; por otro, garantizar la autenticidad tanto del mensaje como de los actores de la comunicación. En efecto, el mensaje recibido ha de ser realmente el enviado, a la vez que el remitente y el destinatario han de ser realmente quienes dicen ser y no remitentes y/o destinatarios fraudulentos. La Criptografía clásica se ocupaba únicamente del primer aspecto, mientras que la Criptografía moderna, basada en el concepto de comunicaciones seguras, ha de garantizar tanto una como la otra.

El tipo particular de transformación aplicada al texto claro o las características de las claves utilizadas marcan la diferencia entre los diversos métodos criptográficos. Una primera clasificación en base a las claves utilizadas puede desglosarse tal y como sigue:

- Métodos simétricos: Son aquellos en los que la clave de cifrado coincide con la de descifrado. Lógicamente, dicha clave tiene que permanecer secreta, lo que presupone que emisor y receptor se han puesto de acuerdo previamente en la determinación de la misma, o bien que existe un centro de distribución de claves que se la ha hecho llegar a ambos por un canal seguro.
- Métodos asimétricos: son aquellos en los que la clave de cifrado es diferente a la de descifrado. En general, la clave de cifrado es conocida libremente por el público, mientras que la de descifrado es conocida únicamente por el usuario.

Los métodos simétricos son propios de la Criptografía clásica o Criptografía de clave secreta, mientras que los métodos asimétricos corresponden a la Criptografía de clave pública, introducida por Diffie y Hellman en 1976.

Una de las diferencias fundamentales entre Criptografía clásica y moderna radica en el concepto de seguridad. Antes, los procedimientos de cifrado tenían una seguridad probable; hoy, los procedimientos de cifrado han de tener una seguridad matemáticamente demostrable. Esto lleva a una primera clasificación de seguridad criptográfica:

- Seguridad incondicional (teórica): el sistema es seguro frente a un atacante con tiempo y recursos computacionales ilimitados.

- Seguridad computacional (práctica): el sistema es seguro frente a un atacante con tiempo y recursos computacionales limitados.
- Seguridad probable: no se puede demostrar su integridad, pero el sistema no ha sido violado.
- Seguridad condicional: todos los demás sistemas, seguros en tanto que el enemigo carece de medios para atacarlos.

Con los antiguos procedimientos manuales y lentos de Criptoanálisis era suficiente la seguridad condicional, pues en la mayoría de los casos se obtenía el descriptado del mensaje cuando la información del documento había perdido toda validez. Si el Criptoanálisis tuvo éxitos de importancia fue sólo porque, al igual que era lento el proceso de análisis, lo era también el de cambio de claves. En la actualidad, con el uso de las computadoras para el Criptoanálisis, los operadores criptográficos tienen que tener propiedades matemáticas que los hagan invulnerables no sólo en el presente, con nuestros conocimientos actuales de matemáticas y el estado actual del desarrollo de las computadoras, sino también en un futuro a corto y mediano plazo.

Ya que las comunicaciones electrónicas se usan hoy en día para casi todas las actividades de interés social, están expuestas a todos los trucos y manipulaciones, consecuencia de las flaquezas humanas. Si en el mundo existiera honradez y confianza mutua, no habría necesidad de la criptografía; pero, a falta de aquéllas, la Criptografía trata de suplirlas con protocolos y algoritmos matemáticos de seguridad demostrable.

3.2 PROCEDIMIENTOS CLÁSICOS DE CIFRADO

3.2.1 PRINCIPIO DE SUSTITUCIÓN Y TRASPOSICIÓN

Dentro de la criptografía clásica aparecen dos procedimientos de cifrado básicos que se han ido repitiendo en épocas posteriores hasta llegar a nuestros días. Estos procedimientos son los de sustitución y transposición. [8]

Sustitución: consiste en establecer una relación corresponsal entre las letras del alfabeto en el que está escrito el mensaje original y los elementos de otro conjunto, que pueden ser el mismo o distinto alfabeto. De esta manera, cada una de las letras del texto original se sustituye por un símbolo o letra previamente asignado, lo que da paso a la creación del criptograma. El receptor, que conoce el alfabeto correspondiente, sustituye cada uno de los símbolos del criptograma por su letra original, lo que da como resultado la recuperación del mensaje original.

Transposición: este método consiste en revolver los símbolos del mensaje original colocándolos en un orden distinto, de manera que el criptograma contenga los mismos elementos del texto original, pero colocados de forma que resulten incomprensibles. El receptor, con conocimiento de la transposición, vuelve a colocar los símbolos desordenados en su posición original.

3.2.1.1 EJEMPLOS HISTÓRICOS DE CIFRADO POR SUSTITUCIÓN.

- **Cifrado de Cesar**. [8] Sustituye la primera letra del alfabeto, A, por la cuarta, D; la segunda, B, por la quinta, E, y así sucesivamente con todas las demás.

En términos matemáticos,

$$Y_i = X_i \oplus Z_i$$

Con (A=0, B=1,..., etc.), donde Y_i corresponde a la i-ésima letra del criptograma; Xi, a la i-ésima letra del texto original, y Zi, a la i-ésima letra de la clave, que en este caso es fija e igual a D.

El símbolo $\oplus$ representa la suma módulo 21, siendo 21 el número de letras en el alfabeto latino. Ejemplo:

```
Mensaje:        VENI VIDI VICI
Clave:          DDDD DDDD DDDD
Criptograma:    BHQM BMGM BMFM
```

Para recuperar el mensaje original se suma nuevamente símbolo a símbolo el criptograma con la inversa de la clave módulo 21:

$$X_i = Y_i \oplus (-Z_i) \qquad (\text{mod } 21)$$

La debilidad de este método radica en que la frecuencia de apariciones de cada letra en el texto original se refleja exactamente en el criptograma. Conociendo la letra de mayor frecuencia en el alfabeto utilizado, queda automáticamente establecida la correspondencia.

Cifrado de Vigenere (1586). Es una generalización del cifrado anterior, con la particularidad de que la clave toma sucesivamente diferentes valores:

```
Mensaje         Paris vaut bien une messe
Clave           loupl oupl oupl oup loupl
Criptograma     aolxd juje pcty iht xsmhp
```

En términos matemáticos puede expresarse como:

$$Y_i = X_i \oplus Z_i \qquad (\text{mod } 26)$$

Con Zi = L, O, U, P, alternativamente, siendo 26 el número de letras del alfabeto. Se observa que a una misma letra en el texto en claro le pueden corresponder diferentes letras en el texto cifrado.

La recuperación del mensaje original es análoga al procedimiento de Cesar, sólo que en ésta se trata de una suma módulo 26.

Aunque el cifrado de Vigenerè fue considerado seguro durante siglos, el método Kasiski (incidencia de las coincidencias), publicado en 1863, consiguió romperlo. Para observar esto se analiza el siguiente ejemplo:

```
PETER LEGRAND IS A GOOD FRIEND OF NAPOLEON LEGRAND
EDGAR EDGARED GA R EDGA REDGAR ED GAREDGAR EDGARED
THZEI PHMRRRG OS R KRUD WVLKNU SI TAGSOKOE PHMRRRG
```

La repetición de un determinado grupo de letras en el criptograma, proveniente de un mismo grupo de letras en el texto claro, tiene lugar a una distancia múltiplo de la longitud de la palabra clave. En efecto, el segundo LEGRAND aparece desplazado 30 posiciones con respecto al primero, siendo 5 la longitud de la clave EDGAR. Estudiando estas repeticiones puede determinarse la longitud K de la palabra clave. Una vez conocida K, el criptograma se descompone en K criptogramas sencillos, correspondientes a otros tantos cifrados de Cesar.

Cifrado de Beaufort (1710). [8] Es una modificación del cifrado anterior, con la particularidad de que se suma la clave con la inversa de cada símbolo del texto claro:

```
MENSAJE:        THIS IS THE SAME OLD STUFF
CLAVE:          WIND WI NDW INDW IND WINDW
CRIPTOGRAMA:    DBFL OQ UWS QNRS UCA EPTYR
```

En términos matemáticos puede expresar se como

$$Y_i = Z_i \oplus (-X_i) \pmod{26}$$

Con Zi = W, I, N, D, alternativamente, siendo 26 el número de letras del alfabeto. El receptor, repite el procedimiento de igual forma.

$$X_i = Z_i \oplus (-Y_i) = Z_i \oplus (-Z_i) \oplus X_i \pmod{26}$$

Observe que el cifrado y descifrado se reducen a la misma operación; luego el dispositivo que lleva a efecto este proceso es el mismo tanto en emisión como en recepción. Este tipo de cifrado se conoce como cifrado recíproco o involutivo, en el sentido de que la transformación aplicada es una involución. En efecto,

F[F(texto claro)]= texto claro

Aplicando dos veces la misma transformación F, obtenemos el texto original.

Por otra parte, las debilidades del método de Beaufort son las mismas que las del método de Vigenerè.

Cifrado Vernam (1917). [8] Representa el caso límite del cifrado de Vigenère. Emplea un alfabeto binario, pues inicialmente se utilizó para comunicaciones telegráficas haciendo uso del código Baudot (cinco dígitos binarios por carácter alfabético). La operación aritmética es una suma módulo 2, y la clave una secuencia binaria aleatoria de la misma longitud que el texto claro. Para el mensaje original "come soon" en código ASCII tenemos:

```
Mensaje:       00011 01111 01101 00101 10011 01111 01111 01110
Clave:         11011 00101 01011 00110 10110 10101 01100 10010
Criptograma:   11000 01010 00110 00011 00101 11010 00011 11100
```

Para recuperar el mensaje original se suma nuevamente el criptograma la secuencia aleatoria, ya que adición y sustracción coinciden en la aritmética módulo 2. La originalidad del procedimiento Vernam radica en que la clave se utiliza solamente una vez, pues, es caso contrario, sucesivos criptogramas concatenados darían lugar a un cifrado tipo Vigenère.

El método Vernam fue utilizado durante la segunda guerra mundial por espías de diversas nacionalidades, a los que se les daba una secuencia binaria aleatoria con la recomendación de utilizarla para un único proceso de cifrado.

En círculos criptográficos se creyó durante mucho tiempo en la seguridad total de este método, pero fue Shannon, en 1949, el primero en dar una prueba teórica de la misma.

3.2.2 CONDICIONES DEL SECRETO PERFECTO

Shannon definió sus condiciones de secreto perfecto partiendo de dos hipótesis básicas:[8][6]

1. La clave secreta se utilizará solamente una vez, a diferencia de lo que sucedía en los métodos clásicos, en los que la clave era fija.
2. El enemigo criptoanalista tiene acceso sólo al criptograma; luego está limitado a un ataque sobre texto cifrado únicamente.

Basadas estas dos hipótesis, Shannon enunció sus condiciones de secreto perfecto, que pueden resumirse a lo siguiente:

Un sistema criptográfico verifica las condiciones de secreto perfecto si el texto claro X es estadísticamente independiente del criptograma Y, lo que en lenguaje probabilístico pueden expresarse como:

$$P(X=x \mid Y=y) = P(X=x)$$

Para todos los posibles textos fuente $x = (x_1, x_2, ..., x_M)$ y todos los posibles criptogramas y=(y1,y2,...,yN); es decir, la probabilidad de que la variable aleatoria X tome el valor de x es la misma con o sin conocimiento del valor tomado por la variable aleatoria Y. En términos más sencillos, esto equivale a decir que la información sobre el texto claro aportada por el criptograma es nula. Por tanto, el enemigo criptoanalista no puede hacer una mejor estimación de X con conocimiento de Y que la que haría sin su conocimiento, independientemente del tiempo y recursos computacionales de los que disponga para el procesado del criptograma.

Asimismo, y basado en el concepto de entropía, Shannon determinó la menor cantidad de clave necesaria para que pudieran verificarse las condiciones de secreto perfecto. En efecto, la longitud de la clave K tiene que ser, al menos, tan larga como la longitud del texto claro M:

$$K >= M$$

La desigualdad se convierte en igualdad para el caso del cifrado Vernam. Una vez establecidas las condiciones de secreto perfecto, cabe preguntarse se existen cifradores perfectos. La respuesta es afirmativa, tal y como se verá a continuación.

Se considera un método de cifrado en el que el texto claro, criptograma y clave tomen valores en un alfabeto L-ario {0,1,... L-1} y en el que la longitud de la clave K, criptograma N y texto claro M coincidan entre sí K=N=M. En este caso, el número de posibles textos claros, criptogramas y claves son iguales entre sí e iguales a L^M.

Se supone que

1. La clave se elige de forma completa aleatoria, es decir,

   $P(Z=z)=L^{-M}$

   Para todos los L^M posibles valores z de la clave secreta.

2. La transformación de cifrado es

$$Y_i = X_i \oplus Z_i, (i = 1,..., M)$$

   Donde + denota la adición módulo L, elemento a elemento.

Fijado un texto fuente X=x, a cada posible valor de la clave $Z=z_j$, $(j=1,...,L^M)$, le corresponde unívocamente un criptograma $Y=y_j$, $(j=1,...,L^M)$. Entonces, de acuerdo con la condición 1, es fácil ver que a un mismo texto claro X=x le puede corresponder con igual probabilidad cualquiera de los L^M posibles criptogramas, luego

$$P(Y=y)=P(Y=y|X=x)=L^{-M}$$

Por tanto, la información aportada por el criptograma sobre el texto claro es nula, X e Y son estadísticamente independientes y la transformación módulo L verifica las condiciones de secreto perfecto. Cuando L=2, tenemos simplemente el cifrado Vernam.

Hay que resaltar que este tipo de cifrado módulo L ofrece una total seguridad respecto a la estadística del texto claro, lo cual es una cualidad muy deseable, puesto que sería extraordinariamente peligroso que la seguridad de un método de cifrado dependiera de la naturaleza estadística del lenguaje utilizado en el mensaje a cifrar.

A la luz de las condiciones de secreto perfecto de Sahnnon, podemos evaluar los métodos criptográficos referenciados anteriormente:

Cifrado de Cesar: Utiliza una clave de longitud menor que el texto claro, la clave es fija y reutiliza continuamente para cada nueva letra del mensaje a cifrar. Claramente, este procedimiento no verifica las condiciones de Shannon, por lo que la operación módulo 21 deja al descubierto en el criptograma la frecuencia de aparición de las letras del texto fuente.

Cifrado de Vigenère: Utiliza una clave más larga que el método anterior, pero en cualquier caso más corta que la longitud del mensaje. La clave no es una secuencia aleatoria, sino una palabra de lenguaje, sometida a sus reglas y características, que se reutiliza sucesivas veces. De acuerdo con las condiciones de Shannon, no se trata de un método de cifrado perfecto, por lo que, aunque sea con mayor dificultad que en el cifrado previo, el criptoanalista termina por encontrar algún método que le permite determinar la estadística del texto claro a partir del criptograma y, posteriormente, romper el criptosistema.

Cifrado Vernam: Utiliza una clave de longitud igual a la del texto claro, siendo ésta una secuencia perfectamente aleatoria que además se utiliza solamente una vez. Verifica, pues, las condiciones de secreto perfecto de Shannon. En este caso, la suma módulo 2 con una secuencia aleatoria ofrece un perfecto enmascaramiento del contenido y estadística del texto claro. Dentro del panorama criptográfico actual, el cifrado Vernam es el único procedimiento incondicionalmente seguro o procedimiento con seguridad probada matemáticamente.

3.2.3 APLICACIÓN PRÁCTICA DEL CIFRADO EN FLUJO

Aunque el método Vernam [6] ofrece las máximas garantías de seguridad, presenta sin embargo un inconveniente evidente, y es que requiere un dígito de clave secreta por cada dígito de texto claro. Teniendo en cuenta las exigencias actuales de información a cifrar, el método resulta inviable para su aplicación habitual en Criptografía. Queda más bien reservado para aquellas circunstancias en las que se requiere unas condiciones máximas de seguridad pero un mínimo de información a proteger. En la práctica, lo que se hace es utilizar generadores pseudoaleatorios de secuencia binaria; es decir, algoritmos determinísticos que, a partir de una clave corta (del orden de 128 bits) conocida únicamente por A y B, generan simultáneamente en emisión y recepción una determinada secuencia de longitud deseada. Ésta será la secuencia cifrante que se suma

módulo 2 con el texto claro (emisión) o con el criptograma (recepción) y que hace las veces de clave en el cifrado Vernam. Claramente, estas secuencias generadas por una máquina de estado finita y un algoritmo determinístico nunca podrán ser auténticas secuencias aleatorias; en última instancia serán secuencias periódicas que intentaremos que se asemejen lo más posible a una secuencia aleatoria y las que denominaremos simplemente secuencias pseudoaleatorias.

La idea fundamental del cifrado de flujo consiste en generar una secuencia larga e imprevisible de dígitos binarios a partir de una clave corta elegida de forma aleatoria. El cifrado en flujo es sencillo, rápido y seguro, aun cuando no está exento de cierto inconvenientes. En conclusión, el cifrado en flujo, tal y como se aplica normalmente, no es más que una aproximación al cifrado Vernam, tanto más segura cuanto más se aproxime la secuencia binaria generada a una auténtica secuencia aleatoria.

3.3 FIRMAS DIGITALES

3.3.1 INTRODUCCIÓN A LAS FIRMAS DIGITALES

¿Qué son las firmas digitales?

Una firma digital es un bloque de caracteres que acompaña a un documento (o archivo), acreditando al autor del mismo ("autentificación") y asegurando que no ha existido manipulación posterior de los datos ("integridad").

Para firmar un documento digital, su autor utiliza su propia clave secreta ("llave privada"), a la que sólo él tiene acceso, lo que impide que pueda posteriormente negar su responsabilidad y autoría ("no revocación"). De esta forma, el autor queda vinculado al documento que firma.

Cualquier persona puede verificar la validez de una firma si dispone de la llave pública del autor.

¿Cómo se realiza una firma digital?

El software del firmante aplica (de forma transparente al usuario) un algoritmo hash sobre el texto a firmar, obteniendo un extracto de la longitud fija, y absolutamente específico para ese mensaje (un mínimo cambio en el mensaje produce un extracto completamente diferente).

Los algoritmos hash más utilizados son MD5 ó SHA-1.

Este extracto, cuya longitud oscila entre 128 y 160 bits (en función del algoritmo utilizado), se somete a continuación a cifrar el mensaje mediante la llave privada del autor, previa petición de contraseña.

El algoritmo utilizado para cifrar el extracto puede ser el mismo RSA o una clave específica para firmar tipo DSS.

El extracto cifrado constituye la firma y se añade al final del mensaje (o en un archivo adherido a él).

¿Cómo se comprueba la validez de una firma digital?

Se necesita disponer de la clave pública del firmante para poder verificar su firma.

El software del receptor descifra el extracto cifrado que constituye la firma digital (de forma transparente al usuario), utilizando para ello la llave pública del remitente. Como resultado obtiene un bloque de caracteres.

A continuación, calcula el extracto hash que corresponde al texto del mensaje. Si el resultado coincide exactamente con el bloque de caracteres obtenido en la operación anterior, la firma se considera válida. Si existe la menor diferencia, la firma se considera no válida.

¿Cuál es el punto débil?

El sistema tiene un punto débil, que es consustancial a los criptosistemas de llave pública.

Efectivamente; la firma digital nos permite comprobar la relación entre un mensaje y la llave utilizada ¿Cómo podemos estar seguros de que esa llave corresponde realmente a la persona o entidad que dice poseerla?

Este problema requiere la intervención de una tercera parte fiable, en la que confíen las dos parte implicadas. Es lo que se le denomina Autoridad de Certificación (CA).

Algunos programas, como PGP, no utilizan autoridades de certificación externas, sino que delegan en el propio usuario la responsabilidad de certificar llaves conforme a su criterio, estableciendo lo que se denomina una red de confianza (Web of Trust) totalmente descentralizada, pero con el apoyo de una red de servidores de llaves.

The Global Trust Register es un directorio que contiene las principales llaves públicas del mundo que permite verificar la validez de certificador X.509 y llaves públicas PGP.

¿Qué es una Autoridad de Certificación?

Es una tercera parte que acredita, actuando como una especie de notario que extiende un certificado de llaves (firmado con su propia llave) para la relación entre una determinada llave y su propietario real.

Verisign es la autoridad de certificación más reconocida en el mundo.

¿Qué son los servidores de certificados?

Son aplicaciones destinadas a crear, firmar y administrar certificados de llaves, que permiten a una empresa u organización constituirse como una autoridad de certificación para subvenir y solventar sus propias necesidades.

Los productos utilizados son Netscape Certificate Server y OpenSoft.

3.3.2 METODOLOGIA DE LA CRIPTOGRAFÍA DE LLAVES

Este capítulo explica como se utiliza la criptografía y el protocolo SSL (Secure Sockets Layer) [4] para implementar la seguridad en Internet. Para esto es necesario conocer la familia de tecnologías que la forman, las cuales se explican a continuación.

- **Cifrado** es la transformación de los datos en una forma no legible para asegurar la privacidad. La comunicación en Internet es como enviar tarjetas postales y cualquiera que este interesado puede leer un mensaje en particular; el cifrado ofrece el equivalente a un sobre sellado.
- **Descifrado** es lo opuesto al Cifrado; éste transforma los datos cifrados en el mensaje original.
- **Autentificación** identifica de manera única a una entidad como un individuo, como una computadora en la red o como una organización.
- **Firma Digital** permite identificar a un poseedor de una llave en particular, lo cual es el equivalente digital de firmar un documento de papel.
- **Verificación de la Firma** permite verificar si la firma digital con la que está relacionada un documento es válida.

El modelo de utilización de llaves públicas RSA es ampliamente utilizado para la autentificación y el cifrado de documentos en la industria de la computación. Para poder utilizar la criptografía RSA es necesario pagar los derechos a la compañía RSA Data Security Inc.

La técnica de cifrado por llaves públicas se basa en usar un par de llaves asimétricas para el cifrado y el descifrado. Cada par de llaves consiste en una llave pública y en una llave privada. La llave pública es distribuida mundialmente y la llave privada es guardada siempre en secreto.

Los datos son cifrados con la llave pública y solo pueden ser descifrados con la llave privada, a su vez, los datos cifrados con la llave privada solamente pueden ser descifrados con la llave pública. Esta asimetría es una de las propiedades que han hecho de este modelo algo muy utilizado.

3.3.2.1 UTILIZACIÓN DEL MODELO DE LLAVE PÚBLICA PARA LA AUTENTIFICACIÓN

La Autentificación [4] [3] es un proceso para verificar la identidad así que una entidad puede verificar que otra entidad es quien dice ser. En el siguiente ejemplo se explica como se utiliza el

método de la llave pública para verificar la identidad. La notación *{mensaje} llave* significa que el mensaje ha sido cifrado o descifrado utilizando la *llave*.

Supongamos que José quiere autentificar a Pedro. Pedro tiene un par de llaves, una pública y una privada. Pedro envía a José su llave pública. José genera un mensaje cualquiera y se lo envía a Pedro:

J -> P *mensaje.*

Pedro usa su llave privada para cifrar el mensaje y regresa la versión cifrada a José:

P -> J *{mensaje} LlavePrivadaDePedro*

José recibe el mensaje y lo descifra utilizando la llave pública que previamente le había enviado Pedro. José compara el mensaje descifrado con el que había enviado en un principio; si los dos concuerdan, entonces está seguro de que está hablando con Pedro. Un impostor presumiblemente no puede conocer la llave privada de Pedro y por consecuencia no puede cifrar el mensaje para que José lo verifique.

Una vez explicado como se cifra un mensaje, es necesario saber que no es una buena idea el cifrar algo con la llave privada y enviarlo por la red. Ya que este valor cifrado puede ser utilizado posteriormente.

Así, en lugar de cifrar el mensaje original enviado por Pedro, José construye un compendio del mensaje y a este le aplica el cifrado con la llave privada. Un compendio del mensaje (Message digest) es derivado de un mensaje aleatorio el cual cumple con las siguientes propiedades:

- El complemento es difícil de ser revertido. Alguien tratando de suplantar a José no puede obtener el mensaje original del compendio.
- Un imitador puede pasar mucho tiempo encontrando mensajes diferentes que son producidos con el mismo valor de compendio.

Utilizando el compendio José se protege así mismo. Él produce el compendio del mensaje aleatorio enviado por Pedro y después cifra el resultado. Él envía de regreso a Pedro el compendio cifrado. Pedro puede producir el mismo compendio y autentificar a José a través del descifrado del mensaje enviado y comparar los resultados.

Esta técnica se le conoce como firma digital. Pedro ha firmado un mensaje generado por José.

J -> P *Hola, eres Pedro?*
P -> J *José, Soy Pedro*
{ compendio[José, Soy Pedro] } LlavePrivadaDePedro

Para poder distribuir las llaves públicas por un medio confiable, es necesario que el protocolo de autentificación tenga los siguientes pasos:

J -> P	*Hola*
P -> J	*Hola, Yo soy Pedro, LlavePúblicaDePedro*
J -> P	*Demuéstralo*
P -> J	*José, Soy Pedro*
	{ compendio[José, Soy Pedro] } LlavePrivadaDePedro

Con este protocolo, cualquiera puede ser Pedro. Todo lo que se necesita es una llave pública y una llave privada. Cualquiera pudiera mentirle a José diciendo que es Pedro, y después proveer de una llave pública en lugar de la de Pedro. Entonces si José le envía la información al supuesto Pedro, éste puede descifrarla con su llave privada, y José no podría estar seguro de que no fuera Pedro.

Para resolver este problema, la comunidad de estándares ha inventado un objeto llamado certificado. Un certificado debe contener las siguientes características:

- El nombre de la entidad que otorgo el Certificado (Certificador) (CA)
- La entidad a la cual se le otorgo el certificado (alias el sujeto)
- La llave pública del sujeto
- Algunos periodos de vigencia

El certificado es firmado utilizando la llave privada del Certificador. Todos conocen la llave pública del Certificador. Certificar es un estándar para la distribución de las llaves públicas.

A través la tecnología de certificación, todos pueden examinar el certificado de Pedro para ver si este es autentico o falsificado. Asumiendo que Pedro guarda responsablemente el control de su llave privada y que en realidad es Pedro quien tiene el certificado, entonces todo debe ser correcto. Derivado de esto la corrección al protocolo sería la siguiente:

J -> P	*Hola*
P -> J	*Hola, Yo soy Pedro, Mi_Certificado*
J -> P	*pruébalo*
P -> J	*José, Soy Pedro*
	{ compendio[José, Soy Pedro] } LlavePrivadaDePedro

Ahora cuando José recibe el primer mensaje de Pedro, él puede examinar el certificado, verificar la firma a través de compendio y la llave pública, y además de verificar al sujeto y ver que realmente es Pedro. José puede confiar en la llave pública de Pedro y solicitar que pruebe su identidad.

Pero esto no lo es todo, ya que para asegurar que nadie pueda interferir y añadir información maliciosa a la comunicación es necesario agregar al protocolo un código de autentificación del mensaje (MAC). Un MAC es una parte de datos que es procesada utilizando un secreto y algunos datos transmitidos. El algoritmo del compendio incluye la función para construir el código de autentificación.

MAC := Compendio[ParteDelMensaje, Secreto]

Uno de los algoritmos utilizados es el MD5 (un algoritmo de compendio inventado por la RSA), el cual permite intercambiar información utilizando un MAC de 128 bits, y la probabilidad de que alguien adivine el MAC es aproximadamente de 1 entre 18,446,744,073,551,616.

A continuación se presenta el protocolo final:

J -> P	*Hola*
P -> J	*Hola, Yo soy Pedro, Mi_Certificado*
J -> P	*pruébalo*
P -> J	*José, Soy Pedro*
	{ compendio[José, Soy Pedro] } LlavePrivadaDePedro
J -> P	*Confirmado, aquí está el secreto {secreto} LlavePúblicaDePedro*
P -> J	*{Algún mensaje, MAC} LlaveSecreta*

3.4 PROTOCOLOS DE SEGURIDAD UTILIZADOS ACTUALMENTE

El principal problema para la explosión definitiva del comercio electrónico es que no existe un protocolo estándar y definitivo para realizar pagos seguros por Internet. SSL ofrece una gran seguridad pero si lo comparamos con otros protocolos existentes como CyberCash, NetBill o Mondex nos damos cuenta que SSL presenta una serie de limitaciones intrínsecas que pueden comprometer la utilidad del protocolo para un uso generalizado del mismo. Analizando los criterios de seguridad y las deficiencias de SSL respeto a estos criterios podremos hacernos una idea de cómo debería ser el protocolo ideal.

Criterios de Seguridad

Todo sistema de pago por medios no seguros, como es el caso de Internet, debe satisfacer una serie de criterios de seguridad que impidan cualquier tipo de fraude. Estos criterios son:

- Confidencialidad

Ninguna persona ajena a la transacción puede tener acceso a los datos. Más aún, las entidades implicadas en la compra no deberían conocer más datos que los imprescindibles para realizar su función. De este modo, el Vendedor no tendría porque tener acceso a los datos financieros del cliente y el banco tampoco debería conocer la lista de los artículos adquiridos.

- Integridad

Ningún dato puede ser modificado ni durante ni después de la conexión.

- Autentificación

Todas las entidades participantes en la transacción deben estar debidamente autentificadas antes de comenzar la compra. Por razones de privacidad, el cliente solo debería garantizar que es el legitimo propietario de la Tarjeta de Crédito, sin necesidad de hacer publica su identidad.

- No Repudio

Debe garantizarse que una vez finalizada la compra ninguna de las partes pueda negar haber participado en ella. Es decir, al finalizar la transacción debe quedar algo equivalente a un Recibo de compra firmado.
Independientemente de este criterio, la ley española establece que todo cliente que participe en una compra a distancia tiene el derecho a negar su participación en ella.

3.4.1 SECURE SOCKET LAYER (SSL)

3.4.1.1 DESCRIPCIÓN

El protocolo SSL [3] fue desarrollado por Netscape para permitir confidencialidad y autentificación en Internet. SSL opera como una capa adicional entre Internet y las aplicaciones, esto permite que el protocolo sea independiente de la aplicación, siendo posible utilizar FTP, Telnet y otras aplicaciones además de HTTP.

Para establecer una comunicación segura utilizando SSL se tienen que seguir una serie de pasos. Primero se debe hacer una solicitud de seguridad. Después de haberla hecho, se deben establecer los parámetros que se utilizarán para SSL. Esta parte se conoce como *SSL Handshake*. Una vez se haya establecido una comunicación segura, se deben hacer verificaciones periódicas para garantizar que la comunicación sigue siendo segura a medida que se transmiten datos. Tras finalizar la transacción se termina SSL.

Solicitud de SSL:
Antes de que se establezca SSL, se debe hacer una solicitud. Típicamente esto implica un cliente haciendo una solicitud de un URL a un servidor que soporte SSL. SSL acepta solicitudes por un puerto diferente al utilizado normalmente para ese servicio. Una vez se ha hecho la solicitud, el cliente y el servidor empiezan a negociar la conexión SSL, es decir, hacen el *SSL Handshake*.

SSL Handshake:
Durante el *handshake* se cumplen varios propósitos. Se hace autentificación del servidor y opcionalmente del cliente, se determina que algoritmos de criptografía serán utilizados y se genera una llave secreta para ser utilizada durante el intercambio de mensajes subsiguientes durante la comunicación SSL.

Los pasos que se siguen son los siguientes:

Client Hello: El "saludo de cliente" tiene por objetivo informar al servidor que algoritmos de criptografía pueden utilizar y solicita una verificación de la identidad del servidor. El cliente envía el conjunto de algoritmos de criptografía y compresión que soporta y un número aleatorio. El propósito del número aleatorio es para que en caso de que el servidor no posea un certificado para comprobar su identidad, aún se pueda establecer una comunicación segura utilizando un conjunto distinto de algoritmos. Dentro de los protocolos de criptografía hay un protocolo de intercambio de llave que define como cliente y servidor van a intercambiar la información, los algoritmos de llave secreta que definen que métodos pueden utilizar y un algoritmo de hash de una sola vía. Hasta ahora no se ha intercambiado información secreta, solo una lista de opciones.

Server Hello: El servidor responde enviando su identificador digital el cual incluye su llave pública, el conjunto de algoritmos criptográficos y de compresión y otro número aleatorio. La decisión de que algoritmos serán utilizados está basada en él más fuerte que tanto cliente como servidor soporten. En algunas situaciones el servidor también puede solicitar al cliente que se identifique solicitando un identificador digital.

Aprobación del Cliente: El cliente verifica la validez del identificador digital o certificado enviado por el servidor. Esto se lleva a cabo descifrando el certificado utilizando la llave pública del emisor y determinando si este proviene de una entidad certificadora de confianza. Después se hace una serie de verificaciones sobre el certificado, tales como fecha, URL del servidor, etc. Una vez se ha verificado la autenticidad de la identidad del servidor. El cliente genera una llave aleatoria y la cifra utilizando la llave pública del servidor y el algoritmo criptográfico y de compresión seleccionado anteriormente. Esta llave se le envía al servidor y en caso de que el handshake tenga éxito será utilizada en el envío de futuros mensajes durante la sesión.

Verificación: En este punto ambas partes conocen la llave secreta, el cliente por que la generó y el servidor por que le fue enviada utilizando su llave pública, siendo la única forma posible de descifrarla utilizando la llave privada del servidor. Se hace una última verificación para comprobar si la información transmitida hasta el momento no ha sido alterada. Ambas partes se envían una copia de las anteriores transacciones cifradas con la llave secreta. Si ambas partes confirman la validez de las transacciones, el handshake se completa, de otra forma comienza de nuevo el proceso.

Ahora ambas partes están listas para intercambiar información de manera segura utilizando la llave secreta acordada y los algoritmos criptográficos y de compresión. El handshake se realiza solo una vez y se utiliza una llave secreta por sesión.

En la figura se ilustra el proceso de handshake:

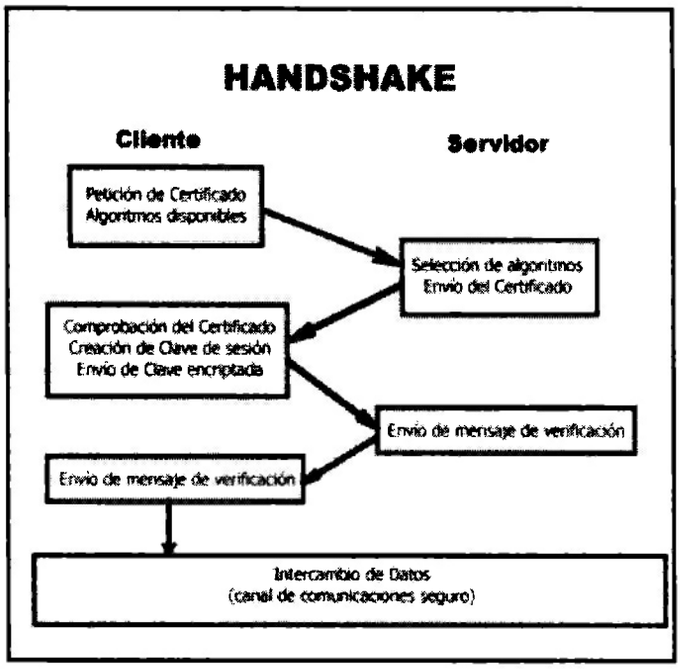

Figura [13] Funcionamiento del Handshake

Intercambio de datos:
Ahora que se ha establecido un canal de transmisión seguro SSL, es posible el intercambio de datos. Cuando el servidor o el cliente desea enviar un mensaje al otro, se genera un digest (utilizando un algoritmo de hash de una vía acordado durante el handshake), cifran el mensaje y el digest y se envía, cada mensaje es verificado utilizando el digest.

Terminación de una sesión SSL:
Cuando el cliente deja una sesión SSL, generalmente la aplicación presenta un mensaje advirtiendo que la comunicación no es segura y confirma que el cliente efectivamente desea abandonar la sesión SSL.

NOTA: Ver el Anexo B para analizar el código del SSL.

3.4.1.2 IMPLEMENTACIÓN DEL PROTOCOLO SSL

Como ya se ha comentado anteriormente, el sistema más utilizado en la actualidad para garantizar pagos seguros por Internet es el SSL (Secure Socket Layer), un protocolo de seguridad que se ha convertido en un estándar de Internet y que viene incluido por defecto en los navegadores Microsoft Explorer y Netscape Navigator. Lo que permite una puesta en marcha muy sencilla del sistema de pago ya que el cliente puede comenzar a comprar sin tener que realizar ningún proceso de autentificación previa.

Para crear un sistema de pago electrónico basado en SSL es necesario conseguir un certificado electrónico para el Vendedor, generalmente se obtiene de la empresa Verisign (filial de RSA Data Security Inc. Y principal Autoridad Certificadora mundial). Verisign está considerada por Microsoft y Netscape como CA de confianza por lo que normalmente viene activada en sus respectivos navegadores.

Una vez realizado el pago, el Vendedor obtiene el PIN de la tarjeta de crédito del cliente, por lo que debe estar provisto de algún método que permita enviar estos datos a una entidad financiera que sea capaz de realizar la transferencia bancaria.

En España, Banesto ha sido pionera en ofrecer un sistema completo de pago electrónico basado en SSL e incluso ha creado su propia tarjeta de crédito para este propósito la Virtual C@sh. La comunicación entre el Vendedor y Banesto se realiza a través de un protocolo privado.

Otra opción que han ofrecido varios bancos es la de utilizar un Terminal Punto de Venta (TPV) para realizar la transferencia. Se conecta la TPV al servidor del Vendedor y mediante un software CGI se realiza la comunicación.

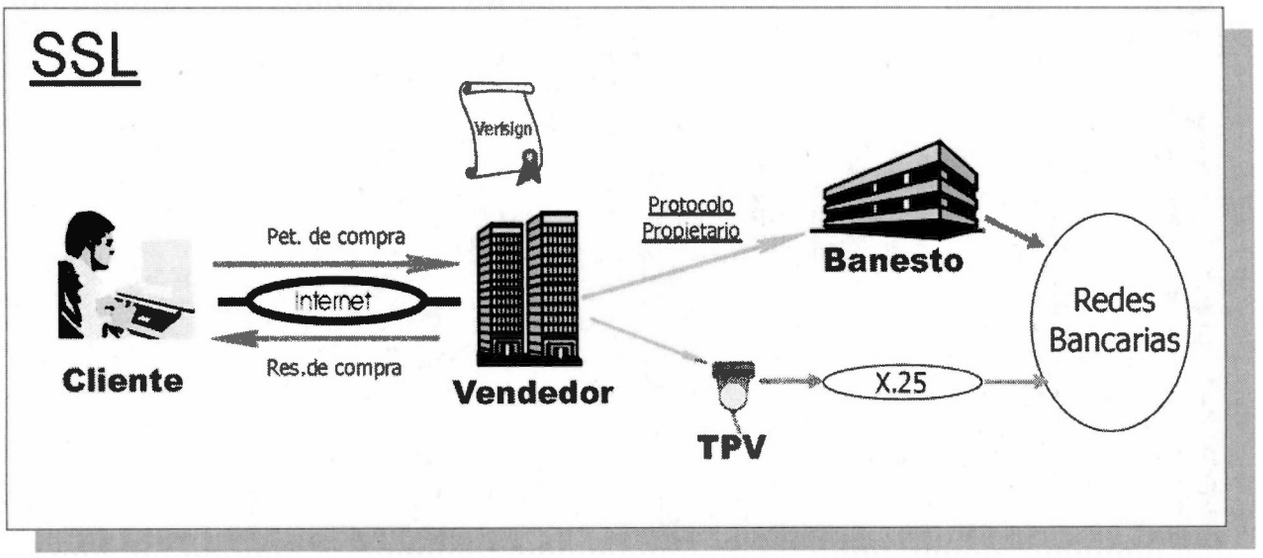

Figura [14] Funcionamiento del SSL

Los servidores seguros SSL los podremos notar porque en al esquina inferior izquierda cambia a una llave cerrada en el caso de Netscape. En la URL, cambia de Http:// a Https:// (Hipertext Transport Protocol Secure).

El gobierno americano permite un cifrado de 128 bits, aunque claro, "La seguridad es solo un estado mental". También el SSL tiene problemas de seguridad, el 26 de junio de 1998 Murray Hill, informático de Lucent Technologies hizo pública la manera de acceder a esa información y obtener el número de tarjeta y datos críticos de la transacción.

El ataque es posible pero poco probable. La vulnerabilidad no se produce sobre el SSL, sino sobre PKCS, (Public Key Cryptography Standard) encargado del intercambio de llaves en SSL.

El ataque consiste en mandar mensajes cuidadosamente construidos contra el servidor seguro que produzcan sendos mensajes de error de vuelta, a partir de cuya información se puede sacar la llave critica para seguridad. Se necesitan en torno a un millón de mensajes, hecho que debe llamar la atención del administrador del sistema.

3.4.1.3 Deficiencias del Protocolo SSL

SSL fue creado como un protocolo de comunicaciones seguro de uso genérico. Como no fue pensado explícitamente para el comercio electrónico presenta usa serie de deficiencias que deben tenerse en cuenta.

- Confidencialidad:

SSL garantiza la confidencialidad extremo a extremo pero una vez finalizada la conexión, el Vendedor posee todos los datos del comprador, así como su número de tarjeta de crédito. El Vendedor podría almacenar esos datos y el Cliente estaría expuesto a cualquier tipo de fraude por parte de toda persona que tuviera acceso a dicha información.

- Integridad:

SSL no garantiza la integridad de la información una vez finalizada la conexión, por lo que el vendedor podría modificar esos datos, por ejemplo, cobrando de más al cliente.

- Autentificación:

El cliente no necesita autentificarse, una persona con acceso a números de tarjeta de crédito robados podría realizar cualquier tipo de compra por Internet. Este es precisamente el tipo de fraude más común y que causa mayores perdidas a las compañías de crédito.

- No repudio:

Una vez finalizada la compra no existe ningún tipo de comprobante de compra por lo que cualquier protesta posterior carecerá de medios para su confirmación. Tampoco existe ningún documento firmado por lo que tanto el Cliente como el Vendedor o el Banco podrían negar su participación en la compra sin que existiera la posibilidad de probar lo contrario.

Con SSL, toda la seguridad recae en la confianza que el Cliente tenga del Vendedor, ya que potencialmente el Vendedor puede realizar cualquier tipo de fraude con total impunidad. Sólo las empresas con muy buena reputación podrían, a priori, contar con esta confianza del consumidor.

El más que posible fraude con números de tarjetas robados hace que las Entidades de Crédito añadan una comisión en las compras bastante elevada (un 5% +/-) para compensar este tipo de fraude. Esto hace que el precio de la compra se incremente considerablemente, lo que anula el atractivo inicial de comprar por Internet: los precios bajos.

Estandarizar la comunicación con el banco es una de los puntos importantes que hay que solucionar para conseguir una mayor transparencia y poder abrirse a la competencia.

SSL utiliza Certificados digitales siguiendo el estándar X.509, es decir, certificados de propósito general. Seria más interesante que existieran Autoridades Certificadoras creadas especialmente para emitir certificados de este tipo y que dichas Autoridades estuvieran avaladas por la banca de tal modo que los certificados digitales expedidos tuvieran conexión con cuentas de bancarias concretas.

3.4.2 PROTOCOLO SET

Las empresas de crédito son las principales interesadas en que el uso de las Tarjetas de Crédito se generalice en todos los ámbitos de la vida, incluyendo evidentemente a Internet. No es de extrañar por tanto que las dos empresas más importantes de este sector, Visa y MasterCard uniesen esfuerzos e impulsarán la creación de un nuevo protocolo que permita los pagos por Internet de un modo totalmente seguro, sin las limitaciones que plantea SSL.

La idea básica consistía en crear un protocolo especialmente diseñado para garantizar la seguridad en el pago mediante Tarjetas de Crédito a través de medios de comunicación inseguros como es el caso de Internet y que este protocolo se convirtiese en un estándar abierto para la industria que sirviese de base a la expansión del Comercio Electrónico por Internet.

Para ello debían contar con el apoyo de las principales compañías informáticas, así que al proyecto se le unieron empresas de la talla de Microsoft, Netscape, IBM, Verifone-HP, GTE y contaron como desarrolladores a RSA Data Security, Verisign, Terisa-Spyrus y SAIC.

El 31 de Mayo de 1997 se hicieron públicas las especificaciones formales del protocolo SET [14] versión 1.0, estas especificaciones se pueden encontrar el web oficial de SET, <www.setco.org>.

Los documentos oficiales son:

1. Book 1: Descripción de negocio
2. Book 2: Guía para Programadores
3. Book 3: Descripción formal del protocolo

A estos documentos hay que añadir un cuarto:

4. Guía de interfase externa

En este documento se dan las normas para la conexión por Internet, ya que SET internamente no especifica el tipo de protocolo de comunicación que debe utilizarse ni las características de las interfases.

3.4.2.1 CARACTERÍSTICAS PRINCIPALES DE SET

Las especificaciones de SET parten de una serie de criterios de diseño que permitan la mayor difusión y seguridad del protocolo. Estas características generales son:

- Estándar abierto
- Objetivo específico: Transferencia de números de tarjetas de créditos
- Utiliza codificación estándar (ASN.1 y DER)
- Independiente del medio de comunicación utilizado
- Utiliza estándares criptográficos ampliamente manejados (PKCS, X.509)
- Utiliza Criptografía de Llave Pública
- Autentificación basada en la certificación digital de todas las entidades participantes en la transferencia

3.4.2.2 ENTORNO

A diferencia de SSL, en SET se definen tres entidades independientes: Cliente(Cardholder), Vendedor(Merchant) y la Pasarela de Pago(Gateway Payment) que se interconectan directamente por Internet, haciendo el Vendedor de puente entre el Cliente y la Pasarela de Pago. Previamente a cualquier comunicación entre ellos, todas las entidades deben haber obtenido un certificado digital valido a través de la Autoridad de Certificación adecuada. La Pasarela de Pago permite la conexión desde Internet con las Redes Bancarias como VisaNet, dentro de estas Redes distinguimos otras dos entidades. El Issuer o entidad emisora de la tarjeta de crédito y el Adquirer o banco receptor de la transacción electrónica.

SET se diseñó pensando en su utilización en Internet pero no de un modo exclusivo como SSL, sino que permite la conexión a través de cualquier tipo de red siempre que se definan las interfases adecuadas.

3.4.2.3 JERARQUÍA DE CERTIFICACIÓN

SET es el primer proyecto de certificación a escala global que se va ha realizar en el mundo. Los certificados SET se estructuran siguiendo una jerarquía piramidal única que culmina en una Autoridad Certificadora Raíz (Root CA) [14] que es la encargada de certificar a todas las demás autoridades certificadoras. Bajo la Root CA se encuentran las Brand CA o CA propiedad de las Entidades emisoras de Tarjetas de Crédito. Obviamente las primeras Brand CA's pertenecen a

Visa Internacional y MasterCard Internacional. Las Brand CA's pueden a su vez certificar a otras CA's para que actúen en un ámbito político determinado, estas CA's reciben el nombre de Brand Geopolitical CA. En España, ACE (Agencia de Certificación Española) asumirá el rol de CA Brand Geopolitical para las tarjetas Visa y MasterCard así como CA emisora de certificados de Cardholder, Merchant y Gateway. En México, no se ha decidido tomar alguna entidad certificadora a nivel Mundial, por lo que el Banco de México está proponiendo ser la Autoridad Certificadora nacional.

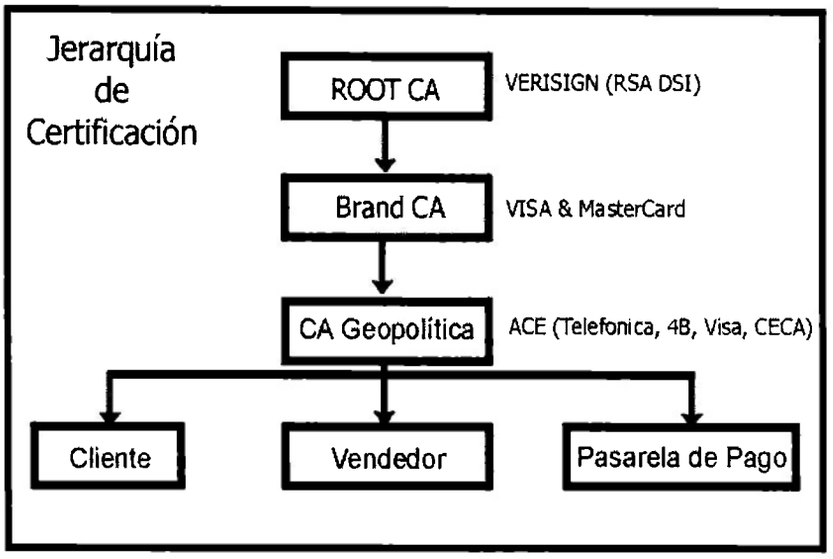

Figura [15] Jerarquía de Certificación

3.4.2.3 AUTORIDADES DE REGISTRO

SET establece un protocolo para la obtención de certificados electrónicos. En la práctica la obtención de un certificado implica que la CA necesita estar segura de que el destinatario del certificado digital es realmente quien dice ser. Esta labor la llevarán a cabo las llamadas Autoridades de Registro, que actuarán de aval ante la CA de los usuarios y se encargarán de tramitar los certificados liberando al usuario final de gran parte de esta labor. Las Autoridades de Registro serán precisamente los bancos, esto permitirá que los certificados estén asociados a números de cuentas bancarias y no a personas físicas; permitiendo las compras anónimas, por lo menos desde el punto de vista del Vendedor.

3.4.2.4 PAGO ELECTRÓNICO

El esquema de pago electrónico SET es muy similar al de CyberCash y, al igual que este, admite una gran variedad de opciones. En un pago normal SET, todo se inicia con una orden de pago que el cliente envía al vendedor. Esta orden de pago está dividida en dos: la descripción de la compra (OD) y los datos financieros del cliente (PIN). Estos datos se firman y se relacionan entre sí por

medio de un algoritmo llamado Firma Dual. Los datos financieros van, a su vez, cifrados con la clave de la pasarela de pago por lo que no pueden ser consultados por el vendedor.

El vendedor envía estos datos cifrados a la Pasarela de Pago que autoriza la transacción. Una vez autorizada, el Vendedor envía una respuesta al comprador firmada que sirve de comprobante de venta. Finalmente el vendedor realiza la captura del importe, es decir envía la orden al banco de que se efectúe la transacción.

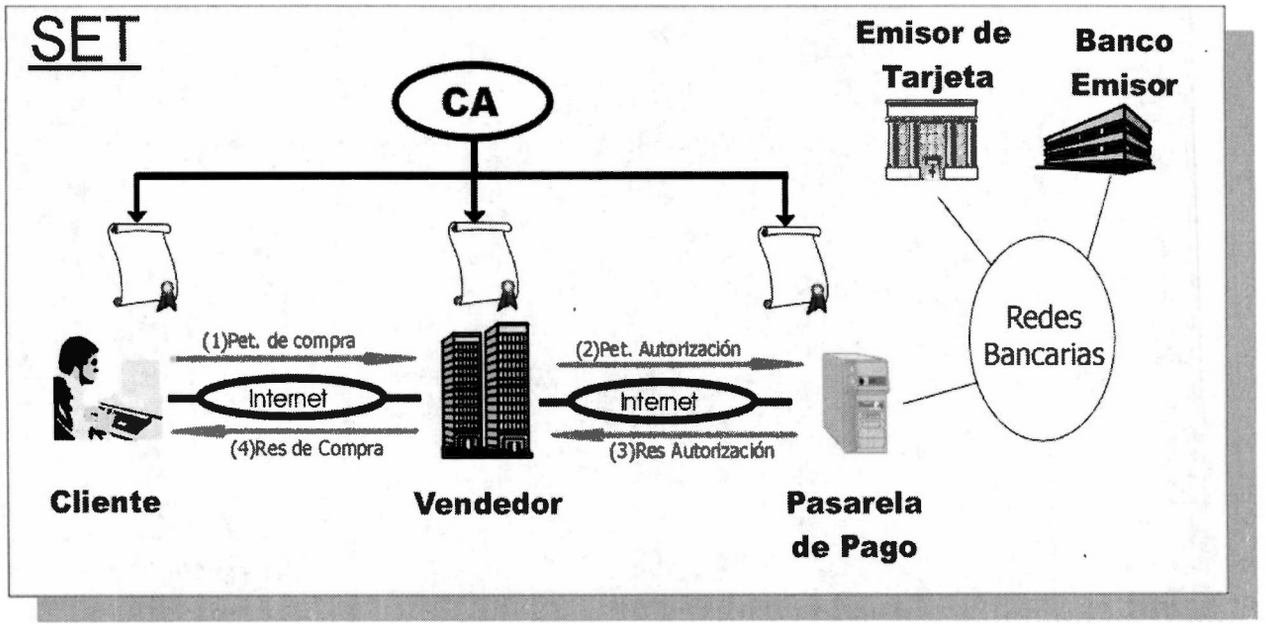

Figura [16] Funcionamiento del SET

3.4.2.5 VENTAJAS SOBRE SSL

SET ofrece una serie de mejoras sobre el sistema basado en SSL. Concretamente en lo referente a los servicios de seguridad podemos comentar lo siguiente:

1. Confidencialidad: Al separar los datos financieros de la descripción de la compra aumentamos la confidencialidad ya que ni el vendedor ni el banco tienen acceso a datos que no le son imprescindibles

2. Integridad: Todos los mensajes van firmados digitalmente de modo que se garantiza la integridad de todos los datos incluso tras finalizar la conexión.

3. Autentificación: Todos los participantes están certificados por una Autoridad Certificadora única, lo que imposibilita cualquier tipo de usurpación de identidad así como la utilización de números de tarjeta de crédito robados

4. No repudio: Los mensajes firmados pueden servir como Recibo de compra, sirviendo de prueba inalterable de que la transacción se produjo de un modo concreto.

3.4.2.6 DEFECTOS DE SET 1.0

La versión 1.0 presenta una serie de defectos y debilidades que intimidan enormemente a la industria y que puede ser uno de los factores que están retrasando la implantación definitiva de este sistema a nivel mundial.

La primera deficiencia importante es la dependencia de algoritmos de cifrado concretos. Esto implica que si se encuentra una vulnerabilidad en alguno de estos protocolos, la seguridad de SET se vería seriamente amenazada. Y precisamente esto es lo que ha sucedido recientemente.

La EFF (Electronic Frontier Foundation) construyó en Noviembre de 1998 una maquina llamada DEScracker capaz de desencriptar el protocolo DES en tan solo 4 días. El objetivo de la EFF no era otro que demostrar lo que se venía diciendo desde hacia tiempo: El algoritmo DES está desfasado y es susceptible de ataques.

Pero el gran defecto de SET 1.0 es su sistema de certificados que presenta numerosas debilidades a la hora de implementarlo en la práctica. Concretamente sus puntos débiles son:

1. Distribución: El modo de conseguir un certificado Digital es tremendamente complicado, especialmente teniendo en cuenta que estos certificados irían destinados al publico en general.

2. Almacenamiento: El talón de Aquiles de SET 1.0, todo el sistema se basa en mantener en secreto la clave privada del cliente. Por lo que esta clave se convertiría en objetivo prioritario de robos, ataques de hackers y virus informáticos. Con la agravante de que lo más probable es que el cliente no fuera consciente de qué es lo que debía proteger.

3. Movilidad: El certificado quedaría almacenado localmente en el ordenador personal del cliente lo que impediría el uso de un terminal diferente para realizar la compra. Este es un defecto estructural de gran importancia por sí solo.

4. Revocación: El método utilizado para controlar los certificados revocados no acaba de convencer a los expertos que están estudiando otras alternativas. Esto, hasta cierto punto es comprensible ya que nunca antes se ha probado una jerarquía de certificación a nivel global de estas características.

3.4.2.7 SET 2.0

Como respuesta a todas estas deficiencias se ha creado un equipo de investigación que está trabajando en la nueva versión de SET que acabara con todos estos problemas. Se permitirá que el protocolo pueda seleccionar entre varios algoritmos de cifrado como SSL y se perfeccionara el sistema de revocación de certificados. Pero el punto básico de la versión 2.0 será la unión de SET con otro tipo de tecnología muy de moda en la actualidad: Las tarjetas Inteligentes, Tarjetas Chip, o smart cards.

Las tarjetas inteligentes son unas tarjetas del tamaño de las tarjetas de crédito convencionales que en lugar de almacenar la información en una cinta magnética lo hacen en un circuito integrado que llevan acoplado a la propia tarjeta. Estas son similares a las que se utilizan para el servicio público telefónico en México. Este chip es un pequeño Microprocesador con capacidad para almacenar y generar claves criptográficas y realizar algunos algoritmos de cifrado.

Con la utilización de estos dispositivos se solucionarían las deficiencias más importantes. Las Tarjetas Chip servirían de sistemas de almacenamiento de claves, evitado el almacenamiento de las claves en el propio ordenador y permitiendo la movilidad a otros ordenadores. Además el proceso de certificación se vería muy simplificado desde el punto de vista del cliente final: el usuario iría al banco y obtendría su tarjeta chip debidamente certificada, siendo el propio banco el encargo de realizar los procedimientos necesarios para la certificación.

El Director General de Visa España, declaró en la conferencia "Comercio Electrónico y Dinero Electrónico" organizada en Abril de 1998 en Barcelona por Fundesco, que Visa España apostaba por la utilización de las Tarjetas Chip como medio para evitar el fraude en el pago por Internet y que por ello iban a mantener la comisión de riesgo (un 5%) a todos los pagos por Internet que no utilizasen este sistema. En términos reales esta comisión es uno de los factores decisivos en la popularización de la compra por Internet dada la enorme dependencia que presenta el comercio electrónico al precio final del producto.

El principal handicap de este sistema es la necesidad de utilizar un Lector de Tarjetas Chip en cada Ordenador Personal. Estos dispositivos se popularizarán en un futuro cercano ya que su precio puede llegar a ser muy baratos. Microsoft ya ha asegurado que la próxima versión de Windows soportara este tipo de dispositivos.

3.4.3 IMPLEMENTACIÓN DE SET EN ESPAÑA

La primera compra electrónica utilizando SET en España [14] la realizó Banesto con la colaboración de IBM en Marzo del 1998. La operación se llevó a cabo en las oficinas de la entidad bancaria, donde a través de la tienda electrónica Dinsa, se adquirió un ordenador personal

IBM Aptiva que se pagó con la tarjeta VirtualCash de Banesto. La oferta de soluciones de IBM se llama Commerce.POINT y está basado en el protocolo SET. La Pasarela de Pago utilizada fue la de Sistemas 4B, que sigue en periodo de pruebas.

A principios de 1999 todavía no se tiene una fecha concreta para el lanzamiento definitivo de SET en España.

Con la llegada de SET sé hacia necesaria la creación de alguna Autoridad Certificadora en España con suficiente credibilidad como para emitir certificados SET en territorio español. Los principales promotores de este proyecto fueron Telefónica, Sistemas 4B y Sermepa-Visa España que crearon ACE (Agencia de Certificación Español) en Agosto de 1997.

Finalmente se dio entrada en el accionariado de ACE a la Confederación Española de Cajas de Ahorro (CECA) con el objetivo de que todos los Bancos y Cajas de Ahorro estuviesen representados en ella, quedando el accionariado definitivo de esta manera:

40% Telefónica
20% Sistemas 4B
20% Sermepa – Visa España
20% CECA

Por su composición, ACE se presenta como una apuesta seria por el Comercio Electrónico por Internet en España ya que los máximos implicados en los pagos son a su vez los propietarios de la empresa.

A partir de Junio de 1998, ACE comenzó a emitir certificados SET válidos de forma experimental, dejando la emisión definitiva de certificados al público en general sin una fecha definitiva.

ACE tiene capacidad para emitir certificados SET para Cliente, Vendedor y Pasarela de Pago para tarjetas Visa y MasterCard. Actualmente está en proceso de convertirse en CA Geo-Political Brand de Visa. El software utilizado es de la empresa Entrust y ha pasado por todos los procesos de validación de SETco.

En SET, el proceso de certificación es una de las partes más delicadas y que están provocando mayores retrasos y conflictos. ACE es una autoridad de certificación corporativa por lo que no atiende directamente las peticiones de particulares. Estas peticiones se tramitan a través de las Autoridades de Registro, entidades financieras encargadas de validar la identidad del usuario final.

3.4.3.1 OBTENCIÓN DE CERTIFICADOS

El usuario se personará físicamente en su banco (asumiendo que previamente el usuario posee una cuenta en dicho banco y una tarjeta de crédito asociada a la cuenta) y pedirá un Certificado Digital SET para su tarjeta de Crédito. El banco le entregará una contraseña que le permitirá realizar el proceso de certificación vía web.

El usuario, desde su casa, iniciará el proceso desde su ordenador utilizando la opción de crear Claves en su Software de Cliente SET. El software creará una Clave Pública y otra Privada. La Clave Privada debe permanecer siempre en el ordenador del usuario ya que es la base de la seguridad del sistema SET. Una vez generadas las claves se iniciará una sesión en la Web de la Autoridad de Registro (AR) y se enviará la contraseña obtenida previamente y la Clave Pública generada. La AR realizará una petición de certificado a ACE mediante el protocolo SET de certificación. Como respuesta, ACE emitirá un certificado digital que la AR remitirá al usuario final.

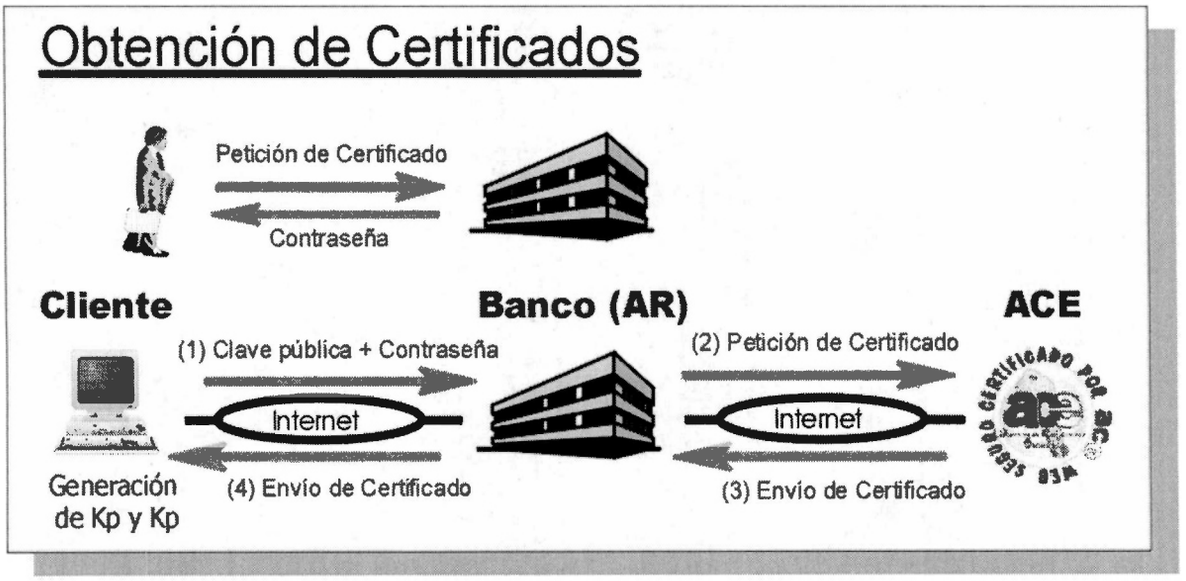

Figura [17] Obtención de Certificados

El Software de Autoridad e Registro para comunicarse con ACE ha sido diseñado por la empresa española Penta3 y está basado en la placa criptográfica SE500+ creada por la misma empresa.

3.4.3.2 ASOCIACIONES

En España existen dos asociaciones de comercio electrónico, AECE y CommerceNet, que se complementan bastante ya que sus objetivos se encuentran bien definidos.

AECE (Asociación Española de Comercio Electrónico), esta asociación forma parte de la Asociación Española de Marketing directo que incluye a varias asociaciones relacionadas con la venta a distancia. Su objetivo es aportar información a sus asociados desde el punto de vista legal. (www.aece.org)

CommerceNet España pertenece al consorcio CommerceNet y tiene como misión impulsar todas aquellas las iniciativas de carácter tecnológico relacionadas con el comercio electrónico, entre sus socios fundadores se encuentra Telefónica. (www.commerce.net)

3.5 EXPLICACIÓN DEL FUNCIONAMIENTO DE UN PROTOCOLO DE AUTENTIFICACIÓN BASADO EN ZERO KNOWLEDGE

Los protocolos Zero Knowledge [1] permiten la identificación, el intercambio de llaves y otras operaciones básicas de cifrado, las cuales no involucran la transmisión de información secreta durante la conversación, además de que requieren menor capacidad de computo que las usadas por los protocolos de llave pública. Por esta razón los protocolos de Zero Knowledge son recomendados para ser utilizados en las tarjetas inteligentes o smart cards y en aplicaciones integradas.

Existen varias propuestas de utilización de los protocolos Zero Knowledge en teoría, pero no existen muchas aplicaciones que los lleven a la práctica tal y como son, a pesar de que muchas aplicaciones tienen ciertas técnicas de Zero Knowledge.

Los protocolos Zero Knowledge [4], se pueden clasificar como protocolos de cifrado, mediante los cuales no revelan información secreta durante la ejecución del protocolo, ni mucho menos a alguien que esté escuchando la red. Estos protocolos tienen propiedades interesantes como son: son un secreto en si mismos, es decir, que no transmiten la identidad a la parte verificadora; con estos protocolos no se pueden hacer pasar por otra persona, es decir, que se evita la usurpación de identidad.

A pesar de que los protocolos Zero Knowledge son poco usuales, muchos de los problemas de cifrado son solucionados utilizándolos, como lo es el modelo de llave pública. Otras posibles aplicaciones son el intercambio de llaves y la identificación interactiva de identidades.

Los siguientes son los participantes de cualquier protocolo de Zero Knowledge:

- El probador que tiene cierta información que desea demostrar al verificador, pero sin que sea necesario que el probador le diga el secreto al verificador.

- El verificador el cual le hará una serie de preguntas, tratando de encontrar si en realidad el probador conoce el secreto o no. El verificador no conoce y no aprende nada del secreto.

- El Espía que es el que escucha la conversación entre el probador y el verificador. Un buen protocolo de Zero Knowledge asegura que ninguna tercera persona aprenda algo del secreto, y se asegura que no se pueda repetir la información posteriormente para evitar que sea utilizada posteriormente.

- El Malicioso que es el que escucha el tráfico de la red y envía maliciosamente mensajes extras o bien modifica y destruye mensajes. El protocolo debe ser lo suficientemente resistente a este tipo de actividad.

3.5.1 TERMINOLOGÍA ZERO KNOWLEDGE

- El *secreto* significa una parte de la información, puede ser una clave de acceso, la llave privada de un sistema de llaves públicas, o una solución de un problema de matemáticas. Con los protocolos Zero Knowledge, el probador puede convencer al verificador que es la parte que tiene el conocimiento, el secreto, sin revelar el secreto mismo, a diferencia de los protocolos tradicionales de nombre de usuario y clave de acceso.

- La *acreditación* significa el aseguramiento de la confidencialidad para cada una de las transacciones del protocolo. Si en un paso del protocolo Zero Knowledge, la posibilidad de que un impostor conteste una pregunta es de 1 en 2, la probabilidad de que pasa en toda una conversación es de 1/2^(número de transacciones).

  Con frecuencia el probador ofrece un problema (valores numéricos particulares que son resultado de problemas de alto grado de dificultad matemática, por ejemplo la factorización de números extremadamente grandes, los cuales dan por resultado número primos muy largos) al verificador, el cual pregunta por una de dos o más posibles soluciones. Si el probador conoce la solución real del problema matemático, es capaz de resolver y dar la solución exacta por la cual está preguntando el verificador. Si desconoce la solución real, no podrá contestar ninguna de las posibles soluciones, por lo que no podrá ser autentificada por el verificador.

- *Cortar y seleccionar* esto se refiere, a la posibilidad de error en un hoyo del protocolo, pero se puede seguir trabajando en el protocolo hasta estar con la seguridad de que el probador es legitimo. Después de que se alcanzó el nivel de confidencialidad que se necesita sin ninguna interrupción, el protocolo se considerará como exitoso.

3.5.2 PROPIEDADES GENERALES DE LOS PROTOCOLOS ZERO KNOWLEDGE

Los protocolos Zero Knowledge [4] pueden ser definidos como protocolos de cifrado, los cuales deben tener las siguientes propiedades:

- El verificador no puede aprender nada del protocolo

El verificador no puede aprender nada del protocolo sin el probador. Este es el concepto principal del Zero Knowledge. Sin esta propiedad, el protocolo no puede ser considerado como un protocolo Zero Knowledge.

- El probador no puede engañar al verificador

Si el probador no conoce el secreto, el no puede tener éxito al tratar de conseguir el secreto. Después de varias interacciones del protocolo, el protocolo debe de ir confirmando que en realidad conoce el secreto.

- El verificador no puede engañar al probador

El verificador no puede obtener información fuera del protocolo, así que siempre se debe de seguir el protocolo. La única tarea que debe hacer el verificador es convencerse a sí mismo que el probador conoce el secreto. El probador siempre revela solamente una solución de muchas existentes a un problema, nunca todas, a menos que trate de encontrar el secreto en el mismo.

- El verificador no puede pretender ser el probador.

Porque la información puede ser escuchada por otras personas es importante que el verificador no pueda tomar la identidad del probador. En algunos de estos protocolos es posible que una tercera persona grabe la conversación y pueda utilizarla posteriormente para autentificarse.

3.5.3 MODOS DE OPERACIÓN

Los protocolos Zero Knowledge pueden ser usados en tres modos:

- Interactivo: donde el verificador y el probador están en constante acción, haciendo transferencias paso a paso.

- Paralelo: donde el probador crea un número no determinado de problemas y el verificador pregunta por algunas de esas soluciones al mismo tiempo. Esto puede ser utilizado para determinar el número de mensajes interactivos con una conexión de respuesta lenta.

- Fuera de Línea: En esta forma el probador crea un número de problemas, y después estos son utilizados con una función de hash en los datos y el conjunto de problemas que utilizará el verificador, es decir que el conjunto de soluciones serán enviados junto con el mensaje. Este modelo lo utilizan las firmas digitales.

3.6 PRUEBAS ZERO KNOWLEDGE

3.6.1 PRUEBAS INTERACTIVAS

Los sistemas de prueba interactivas los define Joe Kilian [1] como sistemas donde aleatoriamente se generan pruebas de conocimiento general, esto da pie a las pruebas Zero Knowledge. Donde se consideran aserciones de forma "Enunciado x está en lenguaje L". Las pruebas interactivas generalizan a las pruebas ordinarias en los siguiente eventos:

1. En los sistemas ordinarios de prueba, el probador envía un mensaje simple al verificador, quien evalúa determinísticamente este mensaje sin ninguna interacción posterior con el probador. En un sistema de prueba interactivo, el probador y el verificador pueden enviar mensajes de uno al otro y viceversa. Ambos, el probador y el verificador se permiten enviar sus mensajes como resultado de seleccionar cara o cruz en un juego aleatório de azar, al echar una moneda al aire. Ambos se consideran entidades probabilísticas. El verificador hace su evaluación en base a la transcripción del protocolo, y envía su conjunto de resultados aleatorios.

2. Suponga que x pertenece L. Con un sistema ordinario de prueba, el verificador garantiza la aceptación de un resultado correcto con probabilidad de 1. Con una prueba interactiva, se puede requerir que el verificador acepte el resultado con una probabilidad cercana a 1.

3. Supóngase que x no pertenece a L. Con una prueba ordinaria, el verificador garantiza el rechazo de la prueba con probabilidad de 1. Con una prueba interactiva, se requiere que el verificador rechace la prueba con probabilidad cercana a 1, a pesar de una estrategia de prueba maliciosa.

En este tipo de pruebas interactivas la probabilidad puede ser que sea mayor a .99 o a .500001, el conjunto de aserciones que posiblemente interactúen deben de dejar el mismo resultado. Se requiere que el número de mensajes y el tamaño de los mensajes permanecen polinomiales en el tamaño de las aserciones probadas. Entonces, los sistemas de prueba interactivos se pueden generalizar como problemas NP. Finalmente, cuando se refiere a las estrategias arbitrarias utilizadas por los probadores maliciosos, estos no obtienen la suficiente información para poder sustituir y engañar al verificador.

Este tipo de pruebas interactivas dan paso a las pruebas Zero Knowledge con un probador y un verificador, donde el probador convence al verificador de un enunciado (con grado alto de probabilidad) sin que de a conocer información acerca de cómo probar el enunciado.

El ejemplo clásico de las pruebas interactivas es el ejemplo de la Caverna de Ali Baba el cual se describe a continuación.

3.6.1.1 PRUEBA DE LA CAVERNA DE ALI BABA

Ésta es la prueba más sencilla con la que se puede representar el funcionamiento de los protocolos Zero Knowledge [7]. Considerando el ejemplo de una variante de la Caverna de Ali Baba, con una puerta secreta que puede ser abierta por una clave. Peggy conoce la clave de la puerta, y ella quiere convencer a Víctor de que la conoce, pero ella no quiere que Víctor conozca la clave.

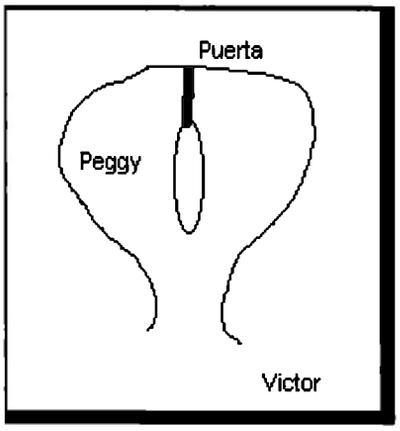

Figura [18] Caverna de Alibaba

Ellos trabajan de la siguiente forma:

- Peggy entra seleccionando aleatoriamente un camino de la caverna, el cual no conoce Víctor estando fuera de la caverna.
- Víctor entra a la caverna e indica aleatoriamente el camino (izquierdo o derecho), donde Peggy debe de salir.
- Si Peggy conoce la clave, ella podrá salir por el camino correcto siempre, abriendo y pasando a través de la puerta secreta, aplicando la clave si es necesario. Si Peggy no conoce la clave, ella tiene un 50% de posibilidad de haber entrado inicialmente en el camino correcto, y si ella no puede pasar a través de la puerta secreta, Víctor la llamará estafadora.

Ellos repiten este proceso tantas veces lo requiera Víctor para estar convencido. Si Víctor desea estar convencido con 1 error en 1024 ocurrencias de que Peggy no conoce la clave, ellos necesitan 10 repeticiones (2^10 = 1024).

Este ejemplo demuestra una de las cualidades de los protocolos Zero Knowledge, en el cual Víctor está convencido de que Peggy conoce el secreto, pero el no lo ha aprendido.

Si Víctor grabará la operación. Pero esa grabación no podrá ser utilizada para convencer a alguien más, solamente que el verificador y el probador estén de acuerdo para poder pasar la prueba.

En este protocolo no existe información generada que aprenda Víctor. Este ejemplo da pauta para conocer parte de las pruebas interactivas que se utilizan en los protocolos Zero Knowledge.

3.6.1.2 PRUEBA CON MÓDULO N

Primero, debe asumir, que se obtendrá un resultado aritmético MÓDULO n, donde $n = p*q$, y p y q son primos. El factorizar n es un problema de difícil solución [1][9].

Rabin muestra en su teorema que el encontrar las raíces cuadradas MÓDULO n es equivalente a factorizar n. Esto es, si se tiene un algoritmo que pueda encontrar las raíz cuadrada de un número MÓDULO n, entonces se puede utilizar ese algoritmo como factor n. La prueba Zero Knowledge consiste en varias interacciones, en las cuales se trata de corroborar que el probador conoce una raíz cuadrada de un número publicado, donde no se ha revelado ninguna nueva información acerca de la raíz cuadrada. Se conoce que existe una raíz cuadrada de ese número (conocimiento público), esto es su residuo cuadrático. Los factores del MÓDULO n puede guardarse en secreto.

El probador, Pedro, publica el residuo cuadrático v para el cual Pedro solicita el conocimiento de la raíz s.

Cuando Pedro trata de probar su conocimiento de s con el verificador, Verónica, Pedro ejecuta varias interacciones. En cada interacción, Pedro selecciona un nuevo número aleatorio r y envía $x = r^2$ *MÓDULO* n a Verónica. Ahora, Verónica selecciona un bit aleatorio b, y lo envía a Pedro. Pedro contesta con $y = r * s ^ b$. Para verificar la hipótesis de Pedro, Verónica calcula y^2 y comparara esto con $x*v ^ b$.

Ahora, viene el análisis.

La primera situación es que sólo Pedro puede completar satisfactoriamente el protocolo obteniendo ambos valores de b. Esto está claro conociendo que $y\_1 = r*s$ cuando $b = 1$ y $y\_2 = r$ cuando $b = 0$, esto significa que se conoce s, mientras que $y\_1/y\_2 = s$.

La segunda situación es que un impostor de Pedro, el cual actualmente desconoce el valor de s puede tener éxito con una probabilidad exactamente de 1/2 en cada interacción: para observar esto, vea que si el impostor de Pedro adivina correctamente que $b = 0$, entonces él puede seguir con el protocolo y tener éxito; de otra manera, si el impostor de Pedro adivina $b = 1$, el impostor de Pedro puede generar x seleccionando un número aleatorio t y ajustando $x = t^2/v$. La respuesta es $y = t$.

La tercera situación es que no existe nueva información en el protocolo. Para ver esto, considere que una persona E escucha la red. En el caso del bit aleatorio $b=0$, E ve un número aleatorio r y que su raíz es x; en el caso de $b = 1$, E observa los números $r*s$ y $x = (rs)^s/v$. Estos números los puede generar el escucha en un acercamiento. Siendo más precisos, un simulador S puede correr

ambos lados del protocolo, y utilizando información avanzada como el valor del bit aleatorio, S puede simular el protocolo sin conocer el valor de *s*.

En cada interacción se muestra que existe 1/2 de posibilidad de que el probador P" pueda no conocer *s*. Interactuando 20 veces da como probabilidad 2^-20 ó 0.0000009536 que P" no conozca *s*. Por lo que la posibilidad es muy pequeña al utilizar este tipo de protocolos.

3.7 RESISTENCIA CRIPTOGRÁFICA DE LOS PROTOCOLOS ZERO KNOWLEDGE

Los libros de texto de matemáticas y de protocolos enseñan las definiciones, pero no siempre muestran una orientación práctica para aplicarlas a la vida real.

Como todos los protocolos criptográficos, Zero Knowledge se basan en la criptografía matemática, es común ver cálculos MÓDULO n, cálculos de matemáticas discretas y cálculos con números extremadamente grandes. Esto es lo que le da fuerza y consistencia a un protocolo de seguridad.

La resistencia criptográfica de los protocolos Zero Knowledge está basada en resolver problemas de difícil solución:

- Tales como problemas para resolver logaritmos discretos de números de cientos de bits.
- Tales como problemas para conocer si un número es cuadrado módulo n o no, siempre que no se conozcan los factores de n.
- Tales como problemas de factorización de número muy grande que son producto de la multiplicación de dos números primos muy grandes (de cientos de bits).

Todo esto hace seguro a un protocolo de este tipo, siempre y cuando el poder de cómputo tenga limitaciones.

CAPÍTULO 4.

4.1 OBJETIVOS DEL MODELO.

Para poder crear un modelo en el cual la autentificación sea a través de un protocolo Zero Knowledge es necesario plantear varios objetivos los cuales se listan a continuación.

El objetivo principal de esta tesis es el de desarrollar una herramienta de autentificación basada en un protocolo Zero Knowledge y la transferencia de la información a través de Internet, utilizando como protocolo de seguridad de transferencia de información el SSL. Dando así la pauta de un modelo para desarrollar una herramienta de Comercio Electrónico en Internet.

Como objetivos específicos en la realización de este trabajo de tesis se tienen:

1. Diseñar un método de autentificación Zero Knowledge por el cual se pueda dar seguridad a los participantes en las transacciones comerciales seguras, sin que sea muy costosa su puesta en marcha.

2. Desarrollar el modelo en un lenguaje que pueda ser transportado a varias plataformas.

3. Obtener los resultados del modelo aplicándolo en un sistema de transacciones comerciales de venta de software en un servidor Windows NT.

Para el cumplimiento de los objetivos anteriormente citados se tienen los siguientes pasos a realizar para cada objetivo.

1. Diseñar un método por el cual se puedan hacer transacciones comerciales seguras, sin que sea muy costosa su puesta en marcha.

- Investigar sobre otros métodos o modelos de transacciones seguras en la red.
- Investigar sobre los protocolos de Zero Knowledge.
- Comparar los diversos protocolos de seguridad
- Comparar las aplicaciones de comercio electrónico en la actualidad.

2. Desarrollar el modelo en un lenguaje que pueda ser transportado a varias plataformas.

- Desarrollar un modelo en Java para poder brindar transacciones seguras en Internet.
- Utilizar la versión de Java 1.1.8 y Java Web Server 1.1.3 para la programación del modelo.

3. Obtener los resultados del modelo aplicándolo a un sistema de transacciones comerciales de venta de software en un servidor Windows NT.

- Desarrollar el modelo para un servidor Windows NT 4.0 y utilizando el Java Web Server 1.1.3
- Probar la seguridad que se brinda con este modelo en un sistema de comercio electrónico de venta de software por medio de Internet.

Estos objetivos fueron los que marcaron la pauta para la investigación y desarrollo de la tesis. Los análisis y desarrollos obtenidos serán explicados en el capítulo siguiente.

4.2 TEORÍA NUMÉRICA FUNDAMENTAL PARA CRIPTOGRAFÍA

4.2.1 ARITMÉTICA MÓDULO N

La aritmética módulo n [5] es de las operaciones fundamentales, esto se puede aplicar al siguiente problema: Si Pedro tenía como hora de llegada las 10:00 y el llegó 13 horas más tarde, ¿a qué hora llegó? Expresando la hora en un sistema de 12 horas.

(10 + 13) MÓDULO 12 = 23 MÓDULO 12 = 11 MÓDULO 12

Otra forma de escribir esta relación es diciendo que 23 y 11 son equivalentes módulo 12.

$10 + 13 \equiv 11$ (MÓDULO 12)

Básicamente, $a \equiv b$ *(MÓDULO n)* si $a = b + kn$ para algún entero k. Si a es positivo y b está entre 0 y n, se puede pensar que b es el residuo de a entre n. En algunas ocasiones, b es llamado el residuo de *a MÓDULO n.* Algunas veces a es designado como congruente a *b MÓDULO n.*

La operación *a MÓDULO n* denota al residuo de a, así que el residuo es algún entero de *0* a *n-1.* Esta operación es una reducción modular. Por ejemplo, *5 MÓDULO 3 = 2.*

Esta definición de MÓDULO puede ser diferente de la definición utilizada por algunos lenguajes de programación. Por ejemplo, la operación MÓDULO de Pascal en algunas ocasiones regresa números negativos. Este da como resultado números entre *–(n-1) y n-1.* Por lo que hay que sumar n cuando salga un número negativo.

La operación MÓDULO como cualquier operación aritmética tiene las siguientes propiedades: conmutativa, asociativa y distributiva. Por lo que se expresan a continuación.

(a+b) MÓDULO n = ((a MÓDULO n) + (b MÓDULO n)) MÓDULO n
(a-b) MÓDULO n = ((a MÓDULO n) -(b MÓDULO n)) MÓDULO n
(a*b) MÓDULO n = ((a MÓDULO n) * (b MÓDULO n)) MÓDULO n
(a * (b + c)) MÓDULO n = (((a * b) MÓDULO n) + ((a *c) MÓDULO n)) MÓDULO n

En criptografía se utilizan mucho las operaciones MÓDULO, porque el calcular logaritmos discretos y raíces cuadradas MÓDULO n pueden ser problemas muy difíciles de resolver. Para k bit MÓDULO n, los resultados intermedios de cualquier suma, resta o multiplicación no debe ser más largo que 2k bits de longitud. Partiendo de esto, se puede calcular la exponenciación en la operación MÓDULO sin generar resultados parciales.

Calcular la elevación de algún número MÓDULO otro número, a^x *MÓDULO n* no es más que una serie de multiplicaciones y divisiones. Para poder calcular rápidamente este tipo de elevación se puede utilizar la propiedad distributiva, la cual permite hacer operaciones más sencillas rápidamente. Por ejemplo: Si se calcula a^8 *MÓDULO n* es igual a *(a*a*a*a*a*a*a*a) MÓDULO n* lo cual se puede reducir a *((a^2 MÓDULO n)^2 MÓDULO n)^2 MÓDULO n.*

4.2.2 NÚMEROS PRIMOS

Un número primo [5] es un entero mayor a 1 el cual tiene como únicos factores al 1 y a sí mismo. Dos es un número primo, así como el 73, 2521, 2365347734339, y 2^{756839}-1. Hay un número infinito de números primos. La criptografía de llave pública utiliza primos largos de 512 bits o mide mayor longitud.

4.2.3 MÁXIMO COMÚN DIVISOR

Dos números son primos relativos [5] cuando ellos comparten sólo como factor el número 1. En otras palabras, si el máximo común divisor de *a* y *n* es igual a 1. Esto se escribe como:

$$mcd(a,n) = 1$$

Los números 15 y 28 son primos relativos, 15 y 27 no lo son, y 13 y 500 son primos relativos. Un número primo es primo relativo a todos los demás números excepto sus múltiplos. El algoritmo de Euclides describe perfectamente como calcular el mcd de dos números. Este algoritmo se puede ejemplificar con el siguiente programa en C:

```
/* Da como resultado el MCD de x y y*/

int mcd (int x, int y)
{
```

```
        int g;

        if (x < 0)
                x = -x;
        if (y < 0)
                y = -y;
        if (x+y == 0)
                ERROR;
        g = y;
        while (x >0) {
                g = x;
                x = y%x;
                y= g;
        }
        return g;
}
```

4.2.4 INVERSO DEL MÓDULO DE UN NÚMERO

Para poder explicar el inverso del MÓDULO n [5], empezaremos por recordar el inverso multiplicativo, esto lo haremos con el siguiente ejemplo: el inverso de 4 es ¼, porque 4 * ¼ = 1. Las operaciones del inverso MÓDULO *n* son más complicadas que el inverso multiplicativo.

El inverso MÓDULO *n* se puede explicar con el siguiente ejemplo:

$$4 * x \equiv 1 \text{ (MÓDULO 7)}$$

Esta ecuación es equivalente a encontrar una *x* y una *k* tales que

$$4x = 7k + 1$$

donde ambas *x* y *k* son enteros.

El problema general es encontrar una *x* tal que

$$1 = (a*x) \text{ MÓDULO n}$$

Esto puede escribirse como:

$$a^{-1} \equiv x \text{ (MÓDULO n)}$$

El inverso MÓDULO n es un problema complicado de resolver. Algunas ocasiones tiene solución, en otras no tiene solución. Por ejemplo: el inverso de 5 MÓDULO 14 es 3. Y el 2 MÓDULO 14 no tiene solución.

En general, $a^{-1} \equiv x$ (MÓDULO n) tiene una única solución si *a* y *n* son primos relativos. Si *a* y *n* no son primos relativos entonces $a^{-1} \equiv x$ (MÓDULO n) no tiene solución. Si *n* es un número primo, entonces para todo número en el rango de 1 a *n*-1 es un primo relativo de *n* y tiene exactamente un inverso MÓDULO *n* en ese rango.

4.2.5 TEOREMA PEQUEÑO DE FERMAT

Si m es un número primo, y a no es un múltiplo de m, entonces el teorema pequeño de Fermat dice que:

$$a^{m-1} \equiv 1 \text{ (MÓDULO m)}$$

4.2.6 TEOREMA DEL RESTO CHINO

Si se conoce la factorización prima de *n* [5], entonces se puede utilizar el teorema del resto chino para resolver algunos sistemas de ecuaciones. En general, si la factorización prima de n es $p_1 * p_2 *...*p_t$, entonces el sistema de ecuaciones

$$(x \text{ MÓDULO } p_i) = a_i, \text{ donde i } 1,2,...,t$$

tiene una única solución, *x*, donde *x* es menor que *n*. En otras palabras, un número es identificado por sus residuos MÓDULO los números primos seleccionados.

Por ejemplo, usando los números primos 3 y 5, y el número 14. 14 MÓDULO 3 = 2, y 14 MÓDULO 5 = 4. Hay solamente un número menor que 3*5 = 15 el cual tiene como residuos 2 y 4, este número es el 14. Estos residuos determinan al número 14.

Así, para un arbitrario a < p y b < q (donde p y q son primos), existe una única x, donde x es menor que p*q, tal que

$$x \equiv a \text{ (MÓDULO p), y } x \equiv (b \text{ MÓDULO q)}$$

Para encontrar esta x, primero se utiliza el algoritmo de Euclides para encontrar u, tal que

$$u*q \equiv 1 \text{ (MÓDULO p)}$$

Entonces se calcula:

$$x = (((a-b)*u) \text{ MÓDULO } p)*q + b$$

El teorema del resto chino puede ser utilizado para encontrar la solución al problema: si p y q son números primos, y p es menor que q, entonces existe una única x menor que p*q, tal que

$$a \equiv x \text{ (MÓDULO } p), \text{ y } b \equiv x \text{ (MÓDULO } q)$$

Si a >= b MÓDULO p entonces

$$x = (((a - (b \text{ MÓDULO } p))*u \text{ MÓDULO } p)*q + b$$

Si a < b MÓDULO p entonces

$$x = (((a + p - (b \text{ MÓDULO } p))*u) \text{ MÓDULO } p)*q + b$$

4.2.7 RESIDUOS CUADRÁTICOS

Si p es un número primo, y a es mayor que 0 y menor que p, entonces a es un residuo cuadrático MÓDULO p sí

$$x^2 \equiv a \text{ (MÓDULO } p), \text{ para alguna } x$$

No todos los valores de *a* satisfacen la propiedad. Para que *a* sea un residuo cuadrático MÓDULO n, debe se un residuo cuadrático MÓDULO todos los factores primos para n. Por ejemplo, si p = 7 sus residuos cuadráticos son 1,2, y 4:

$$1^2 = 1 \equiv 1 \text{ (MÓDULO 7)}$$
$$2^2 = 4 \equiv 4 \text{ (MÓDULO 7)}$$
$$3^2 = 9 \equiv 2 \text{ (MÓDULO 7)}$$
$$4^2 = 16 \equiv 2 \text{ (MÓDULO 7)}$$
$$5^2 = 25 \equiv 4 \text{ (MÓDULO 7)}$$
$$6^2 = 36 \equiv 1 \text{ (MÓDULO 7)}$$

Note que cada uno de los residuos cuadráticos aparece dos veces en la lista.
No existen valores de x que satisfagan alguna de estas ecuaciones:

$$x^2 \equiv 3 \text{ (MÓDULO 7)}$$
$$x^2 \equiv 5 \text{ (MÓDULO 7)}$$
$$x^2 \equiv 6 \text{ (MÓDULO 7)}$$

Los residuos no cuadráticos MÓDULO 7 son: 3,5 y 6.

Cuando p es impar, hay exactamente (p-1)/2 residuos cuadráticos MÓDULO p y el mismo número de residuos no cuadráticos MÓDULO p. También, si a es un residuo cuadrático MÓDULO p, entonces a tiene exactamente dos raíces cuadradas, una de ellas está entre 0 y (p-1)/2, y la otra entre (p-1)/2 y (p-1). Una de estas raíces cuadradas es también un residuo cuadrático MÓDULO p; ésta es llamada la raíz cuadrática principal.

Si n es el producto de dos primos, p y q, existen exactamente (p-1)(q-1)/4 residuos cuadráticos MÓDULO n. Un residuo cuadrático MÓDULO n es una raíz perfecta MÓDULO n. Por ejemplo, hay seis residuos cuadráticos MÓDULO 35: 1, 4, 9, 11, 16, 29. Cada residuo cuadrático tiene exactamente cuatro raíces cuadráticas, como se vera a continuación.

El producto de los factores primos 5, 7 nos da como resultado 35 y siendo los residuos cuadráticos con sus respectivas raíces cuadráticas las siguientes:

Residuo Cuadrático	Raíz Cuadrática	Raíz Cuadrática	Raíz Cuadrática	Raíz Cuadrática
1	1.0	29.0	6.0	34.0
v:4	2.0	23.0	12.0	33.0
v:9	17.0	3.0	32.0	18.0
v:11	16.0	9.0	26.0	19.0
v:16	31.0	24.0	11.0	4.0
v:29	22.0	8.0	27.0	13.0

CAPÍTULO 5

Como se ha observado en el análisis previo es importante considerar varias premisas para poder desarrollar este modelo de transacciones comerciales seguras de venta de software, como se mencionó antes en este modelo se utilizará un protocolo Zero Knowledge para la autentificación de los participantes. Y por otro lado se utilizará el protocolo SSL para la transmisión de los datos y peticiones dentro de la transacción.

5.1 INTRODUCCIÓN

Para poder llevar a cabo una transacción comercial es necesario identificar varios actores, entre los que se mencionan, el comprador, el vendedor y el banco. Estos actores aparecen en cualquier sistema comercial, pero al realizarlo a través de los medios electrónicos es necesario considerar muchos otros aspectos que se salen de control. Por lo que es necesario tomar en cuenta que algunas personas se dedicarán a estar escuchando el medio de transmisión para tratar de hacer dinero fácil o simplemente realizar fraudes electrónicos. De aquí se deriva la importancia de tener plena seguridad cuando se comercia a través de medios electrónicos como en Internet.

El ciclo de este protocolo es muy simple, primero el cliente visita la página electrónica del vendedor y observa la información sobre diversos productos que el vendedor ofrece. Si el cliente decide comprar uno o más artículos de esta tienda virtual entonces entra en un área restringida para el comercio electrónico, es aquí en donde empieza todo el proceso de autentificación y validación, el cual será descrito posteriormente.

Una vez que el vendedor autentificó al cliente y validó que en realidad fuera quien dijo ser, preguntará por el pedido que el cliente desea, este pedido será transmitido a través del canal seguro (SSL). Una vez terminada el envío del pedido el vendedor se dispone a efectuar el cargo electrónico, esto lo hará con el banco, al cual le enviará la información de la cuenta electrónica solicitada al cliente, para a su vez solicitar un depósito o pago de la venta realizada, sin conocer realmente la identidad del cliente. A su vez el banco solicitará confirmación por parte del cliente para que la transacción sea completada.

5.2 PERSONAJES DEL PROTOCOLO

Como se ha descrito anteriormente en cualquier protocolo se deben identificar ciertos actores, por lo que en esta tesis utilizaremos los siguientes para la interpretación del protocolo de transacciones comerciales seguras:

El vendedor que será la parte que tratará de verificar la autenticidad del comprador. Y a su vez tratará de verificar si es un pedido autorizado por el cliente, para poder hacer el cargo correspondiente y así dar paso al envío del producto.

El comprador que es la parte que va ha realizar el pedido y tratará de identificarse sin dar datos personales de él.

El banco, el cual verá si realmente los fondos se transfieren. Y realizará la comparación del cliente para poder surtir el pedido.

El estafador, que es la persona que estará a la expectativa para poder averiguar los datos del comprador y así realizar estafas con la información del comprador.

5.3 OBJETIVOS DEL PROTOCOLO

1. Brindar la confianza para poder enviar datos a través de un canal seguro, utilizando el protocolo SET.

2. Poder identificar plenamente a los actores sin conocer su personalidad o datos comprometedores.

3. Utilizar un protocolo Zero Knowledge para la autentificación de las partes involucradas.

4. Cumplir con los criterios básicos de seguridad.

5.4 PROTOCOLO GENERAL DE LA TRANSFERENCIA DE FONDOS

5.4.1 PROTOCOLO GENERAL

Este modelo de transacciones comerciales deberá de ser iniciativa como servicio de un banco el cual deberá contar con el control de las transacciones, ya que dependerá de él el buen fin del uso de los recursos monetarios que ha depositado su cliente en sus arcas. Este servicio deberá ser ofertado a sus clientes y a posibles empresas que deseen vender sus productos a través de la red Internet.

Es necesario que el cliente cuente con una cuenta bancaria en ese banco o con una tarjeta de crédito de una entidad emisora de tarjetas como PROSA, VISA o MasterCard. Por lo que tendremos dos casos prácticamente, el primero será un pago con cuenta de cheque y el segundo será con un pago con tarjeta de crédito.

El vendedor deberá ser afiliado al banco que ofrece este servicio, como se hace actualmente con las terminales punto de venta (TPV) para cargos automáticos. Por lo que el establecimiento que vende tiene una cuenta en la que se depositan todos los recursos de las ventas por este medio.

El vendedor tendrá que contar con un servidor seguro que utilice el SSL para las transmisiones seguras, esto, como se ha visto en capítulos anteriores, requiere de certificados obtenidos de una entidad certificadora, esta entidad deberá ser el banco que provee la terminal punto de venta.

Una vez obtenidos los certificados serán instalados en el servidor de Web del vendedor para poder establecer una comunicación segura a través del SSL, y así poder brindar un canal seguro a su cliente que requiera adquirir un producto en su tienda virtual.

A continuación se definirán los mensajes dentro del protocolo que se llevará a cabo durante toda la transacción.

COMPRADOR	VENDEDOR	BANCO
1. Observa el catálogo y selecciona los productos que va a adquirir.		
2. Decide comprar un producto y solicita la entrada al área segura de ventas.		
	3. El sitio del vendedor recibe una petición https, lo que significa que la petición tendrá que ser procesada a través del servidor seguro en el puerto 443.	
4. El cliente acepta el certificado y se establece la comunicación segura entre el cliente y el vendedor.		
5. El cliente le envía la solicitud de los productos a comprar, además de cierta información para que el vendedor pueda corroborar con el banco que el cliente es quien dice que es. Todo esto es enviado al vendedor por medio del canal seguro	6. El vendedor espera la petición del cliente.	
	7. El vendedor recibe la petición del cliente, la información de los productos seleccionados se los guarda el para su bitácora. Y envía al banco la cantidad a cobrar y la	

	información que le envío el cliente para su autentificación.	
		8. El banco recibe la información y procede a autentificar al cliente por medio de la información enviada. Una vez identificado procede ha realizar una corroboración del monto con el cliente.
9.El cliente recibe la corroboración del banco y contesta al banco si el monto es el adecuado.		
		10. Una vez corroborada la información se procede ha realizar el cargo o la cancelación dependiendo del cliente. Y envía al vendedor esta información.
	11. El vendedor recibe la aceptación o la cancelación, y de ser satisfactoria procede con la venta normal. De lo contrario cancela la operación.	
12. El cliente es notificado por el vendedor de los resultados de la transacción. Y él puede proseguir con sus actividades normales.		

5.4.2 PROTOCOLO ZERO KNOWLEDGE

Dentro de este protocolo de transacciones comerciales se plantea utilizar el protocolo que modificaron Uriel Feige, Amos Fiat y Adi Shamir para probar la identidad de una persona. Este algoritmo lo publicaron en Julio de 1986 y lo quisieron patentar en Estados Unidos, pero como el protocolo es muy bueno, el gobierno americano trato de mantenerlo en secreto, declarando una orden para que no fuera publicado fuera de los Estados Unidos.

Pero al ver esto y no siendo ciudadanos norteamericanos los autores decidieron presentar su trabajo en diferentes naciones, para que de esa forma lo conocieran y lo utilizaran. De esta premisa nace que utilice este protocolo para el modelo propuesto.

El esquema que se utiliza para este modelo es un modelo de construcción paralela el cual incrementa el número de acreditaciones por interacción y reduce el número total de interacciones entre el cliente y el verificador.

A continuación se explica el protocolo. Primero se genera n, que es el producto de dos primos largos. Esto es con el fin de generar las llaves pública y privada del cliente, se seleccionan k

diferentes números: v_1, v_2, ..., v_k, donde cada vi es un residuo cuadrático MÓDULO n. En otras palabras, se selecciona una

v_i tal que $x^2 = v_i$ MÓDULO n tiene solución y v_i^{-1} MÓDULO n existe.

Esta cadena, v_1, v_2, ..., v_k, es la llave publica.

Entonces se calcula la más pequeña s_i tal que

$s_i = \text{sqrt}(v_i^{-1})$ MÓDULO n.

Esta cadena s_1, s_2, ..., s_k, es la llave privada.

Y el protocolo es:

1) El cliente toma una r aleatoria, donde r es menor que n. El entonces calcula $x = r^2$ MÓDULO *n*, y lo envía al verificador.
2) El verificador envía al cliente una cadena de bits aleatorios de longitud k: b_1, b_2, ..., b_k
3) El cliente calcula $y = r * (s_1^{b_1} * s_2^{b_2} * ... * s_k^{b_k})$ MÓDULO n. Y envía *y* al verificador.
4) El verificador corrobora que $x = y2 * (v_1^{b_1} * v_2^{b_2} * ... * v_k^{b_k})$ MÓDULO n

El cliente y el verificador repiten el protocolo t veces, hasta que el verificador se convenza de que el cliente conoce s_1, s_2, ..., s_k.

La posibilidad de que el cliente pueda engañar al verificador es de $1/2^{kt}$. Se sugiere que k = 5 y t = 4 para que la posibilidad de engaño sea de $1/2^{20}$.

Para demostrar el que el protocolo funciona analicemos el siguiente ejemplo:

Si n = 35 (que es el producto de 5 y 7), entonces para poder calcular las cuatro raíces cuadráticas de cada residuo cuadrático se necesitan hacer los cálculos como siguen:

Seleccionemos la v = 11

a continuación, se determina las congruencias del número v módulo los factores de n:

v MÓDULO 5 = 11 MÓDULO 5 = 1
v MÓDULO 7 = 11 MÓDULO 7 = 4

Después se calculan las raíces cuadradas de 1 MÓDULO 5 y de 4 MÓDULO 7. Lo cual es posible al buscar un número que satisfaga la siguiente ecuación:

$1 = x^2$ MÓDULO 5
$4 = y^2$ MÓDULO 7

Estas raíces son: +- 1 y +- 2, respectivamente.

Como siguiente calculo se buscan los inversos de 5 MÓDULO 7 y de 7 MÓDULO 5, buscando un número que satisfaga lo siguiente:

5 * a = 1 (MÓDULO 7)
7 * b = 1 (MÓDULO 5)

como resultado obtenemos 3 y 3 respectivamente, ya que:

5 * 3 = 15 y 15 MÓDULO 7 = 1
7 * 3 = 21 y 21 MÓDULO 5 = 1

Una vez que tenemos estos resultados calculamos por el teorema del resto chino las cuatro raíces de v = 11, para lo cual hay que resolver las siguientes operaciones

((1*7*3) + (2*3*5)) MÓDULO 35 = 16
(-(1*7*3) + (2*3*5)) MÓDULO 35 = 9
((1*7*3) - (2*3*5)) MÓDULO 35 = -9 pero como es negativo entonces 35 –9 = 26
(-(1*7*3) - (2*3*5)) MÓDULO 35 = -16 pero como es negativo entonces 35 –16 = 19

dando como resultado 16,9,26,19 las raíces cuadráticas 11 MÓDULO 35.

A continuación se listan los posibles residuos cuadráticos de n = 35.

v:1	$x^2 \equiv 1$ (MÓDULO 35)	1.0	29.0	6.0	34.0
v:4	$x^2 \equiv 4$ (MÓDULO 35)	2.0	23.0	12.0	33.0
v:9	$x^2 \equiv 9$ (MÓDULO 35)	17.0	3.0	32.0	18.0
v:11	$x^2 \equiv 11$ (MÓDULO 35)	16.0	9.0	26.0	19.0
v:14	$x^2 \equiv 14$ (MÓDULO 35)	7.0	28.0	7.0	28.0
v:15	$x^2 \equiv 15$ (MÓDULO 35)	15.0	15.0	20.0	20.0
v:16	$x^2 \equiv 16$ (MÓDULO 35)	31.0	24.0	11.0	4.0
v:21	$x^2 \equiv 21$ (MÓDULO 35)	21.0	14.0	21.0	14.0
v:25	$x^2 \equiv 25$ (MÓDULO 35)	30.0	30.0	5.0	5.0
v:29	$x^2 \equiv 29$ (MÓDULO 35)	22.0	8.0	27.0	13.0
v:30	$x^2 \equiv 30$ (MÓDULO 35)	10.0	10.0	25.0	25.0

Para automatizar este proceso se puede ejecutar el siguiente programa desarrollado en JAVA.

```
import java.net.*;
import java.io.*;

public class ResiduosCuadraticos{
```

```
public static void main(String[] args) {
    double primo1 = (new Integer(args[0]).longValue());
    double primo2 = (new Integer(args[1]).longValue());

    System.out.println(args[0]);
    System.out.println(args[1]);
    double v[];
    double v1[];


    double n = primo1 * primo2;


    System.out.println("n:"+n);
    for (long i=1; i < n; i++){
        double vmodp1 = (i) % primo1;
        double vmodp2 = (i) % primo2;
        double r1 = 0;
        double r2 = 0;
        double r3 = 0;
        double r4 = 0;
        double sqrtvMODp1Modp1 = 0;
        double sqrtvMODp2Modp2 = 0;
        double invp1MODp2 = 0;
        double invp2MODp1 = 0;
        double vmodn = (i * i) % n;

        //Calculo de la raiz cuadrada de vmodp1 mod p1
        double x = 1;
        while (x < n)
        {
            sqrtvMODp1Modp1 = (x * x) % primo1;
            if (sqrtvMODp1Modp1 == vmodp1){
                sqrtvMODp1Modp1 = x;
                x=i;
                break;
            }else{ sqrtvMODp1Modp1 = 0; }
            x++;
        }

        //Calculo de la raiz cuadrada de vmodp2 mod p2
        double x2 = 1;
        while (x2 < n)
        {
```

```
    sqrtvMODp2Modp2 = (x2 * x2) % primo2;
    if (sqrtvMODp2Modp2 == vmodp2){
        sqrtvMODp2Modp2 = x2;
        x2=i;
        break;
    }else { sqrtvMODp2Modp2 = 0; }
    x2++;
}

//Calculo del inverso de primo1 mod primo2
double x3 = 1;
while (x3 < n)
{
    invp1MODp2 = (x3 * primo1) % primo2;
    if (invp1MODp2 == 1){
        invp1MODp2 = x3;
        x3=i;
        break;
    }else { invp1MODp2 = 0; }
    x3++;
}

//Calculo del inverso de primo2 mod primo1
double x4 = 1;
while (x4 < n)
{
    invp2MODp1 = (x4 * primo2) % primo1;
    if (invp2MODp1 == 1){
        invp2MODp1 = x4;
        x4=i;
        break;
    }else { invp2MODp1 = 0; }
    x4++;
}

if ((sqrtvMODp1Modp1 != 0) & (sqrtvMODp2Modp2 != 0)){
    System.out.print("v:"+i+"\t");
    r1 = (sqrtvMODp1Modp1*primo2*invp2MODp1)+(sqrtvMODp2Modp2*primo1*invp1MODp2);

    r1 = r1 % n;
    if (r1 < 0) { r1 = n + r1; }
    r2 = (sqrtvMODp2Modp2*primo1*invp1MODp2)-(sqrtvMODp1Modp1*primo2*invp2MODp1);
    r2 = r2 % n;
    if (r2 < 0) { r2 = n + r2; }
    r3 = (sqrtvMODp1Modp1*primo2*invp2MODp1)-(sqrtvMODp2Modp2*primo1*invp1MODp2);
```

```
                r3 = r3 % n;
                if (r3 < 0) { r3 = n + r3; }
                r4= (sqrtvMODp1Modp1*(-1)*primo2*invp2MODp1)-(sqrtvMODp2Modp2*primo1*invp1MODp2);
                r4 = r4 % n;
                if (r4 < 0) { r4 = n + r4; }
                System.out.println(r1+"\t"+r2+"\t"+r3+"\t"+r4+" ");
            }
        }
    }
}
```

Una vez que ya se calcularon los residuos cuadráticos se necesitan calcular para cada uno de ellos el inverso MÓDULO n y su raíz cuadrada más pequeña del inverso MÓDULO n.

Lo que da como resultado la siguiente tabla:

v	v^{-1}	S=sqrt(v^{-1})
1	1	1
4	9	3
9	4	2
11	16	4
16	11	9
29	29	8

Cabe mencionar que 14,15,21,25 y 30 no tienen inverso MÓDULO 35, porque ellos no son primos relativos a 35. Esto es claro, porque debe haber (5-1)*(7-1)/4 residuos cuadráticos MÓDULO 35 que son primos relativos a 35: Esto es mcd(x,35) =1.

Así que el cliente debe seleccionar una llave pública consistente en k=5 valores: {4,9,11,16,29}. Y la correspondiente Llave privada es {3,2,4,9,8}, Ejemplifiquemos solo un ciclo de la interacción del protocolo.

a) El cliente escribe su nip multiplicado por un número aleatorio generando r, y calcula r^2 MÓDULO 35, siendo que su nip es 8 y el número al azar es 2, tenemos que r = 16 y que 16^2 MÓDULO 35 = 11. Este resultado se envía al vendedor para que lo pase al banco y verifique al cliente.
b) El banco le envía una cadena binaria aleatoria {1,0,1,1,0}
c) El cliente calcula 16 * $((3^1)*(2^0)*(4^1)*(9^1)*(8^0))$ MÓDULO 35 = 13 y se lo envía al banco
d) El banco verifica que 13^2 * $((4^1)*(9^0)*(11^1)*(16^1)*(29^0))$ MÓDULO 35 = 11

El cliente y el banco deben repetir este protocolo 3 veces más con un diferente número aleatorio. Pero con esto se confirma que el cliente conoce su nip sin transmitir su nip y no hay un aprendizaje de esto.

5.5 RESULTADOS OBTENIDOS

Se logró desarrollar una tienda virtual para poder hacer transacciones comerciales seguras utilizando el protocolo Zero Knowledge y el SSL. A continuación se da un concentrado de los resultados obtenidos.

Para poder analizar el protocolo fue necesario desarrollar la tienda virtual en un servidor Windows NT, en el cual se instaló una version de prueba del Java Web Server 1.1.3, cabe mencionar que este servidor de web es seguro en su versión comercial, ya que importa certificados de las principales entidades certificadoras a niveles mundiales.

El Java Web Server brinda una completa integridad con el desarrollo, ya que nativamente se puede utilizar el lenguaje Java para crear servlets, estos servlets fueron utilizados para buscar los productos y hacer todo el seguimiento del pedido.

Fue necesario crear un servidor en Java que manejara sockets para garantizar la conexión segura al servidor de verificación, esta conexión se realiza a través de un applet en Java que es cargado por el navegador. En esta primera versión el applet sólo puede ser leido por el navegador Netscape 4.0 o superior, ya que el manejo de los sockets a través de los navegadores implica cuestiones de seguridad que actualmente sólo soporta este Netscape.

El protocolo creado en java para la autentificación es rápido, debido a que sólo calcula las raíces MÓDULO n, de acuerdo con la cadena de números binarios enviada por el verificador. Cabe mencionar que es muy difícil conocer las raíces por medio de un esquema de factorización, por lo que este protocolo basa su premisa de seguridad en este aspecto.

5.6 ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS

A continuación se presentan los análisis pertinentes para considerar que el protocolo sugerido en esta tesis cumple con los criterios de seguridad necesarios, además de cumplir con los puntos principales de cada protocolo empleado.

5.6.1. CRITERIOS DE SEGURIDAD

Todo sistema de pago por medios no seguros, como es el caso de Internet, debe satisfacer una serie de criterios de seguridad que impidan cualquier tipo de fraude. Esos criterios son: Confidencialidad, Integridad, Autentificación y No repudio.

Analicemos las ventajas que tiene el modelo propuesto en comparación con la utilización del simple protocolo SSL.

- Confidencialidad en SSL.

SSL garantiza la confidencialidad extremo a extremo pero una vez finalizada la conexión, el Vendedor posee todos los datos del comprador, así como su número de tarjeta de crédito. El Vendedor podría almacenar esos datos y el Cliente estaría expuesto a cualquier tipo de fraude por parte de toda persona que tuviera acceso a dicha información.

- Confidencialidad

En el modelo propuesto la confidencialidad se cumple ya que el Vendedor no obtiene información para acceder a los datos financieros del cliente y el banco tampoco conoce la lista de los artículos adquiridos. Además en el protocolo propuesto no se pone en compromiso la información del cliente, por lo que se cumple con este criterio.

- Integridad en SSL:

SSL no garantiza la integridad de la información una vez finalizada la conexión, por lo que el vendedor podría modificar esos datos, por ejemplo, cobrando de más al cliente.

- Integridad

En el modelo propuesto ningún dato puede ser modificado ni durante ni después de la conexión. Esto es posible a que el vendedor envía directamente al momento de la transacción el monto de lo comprado y después este monto es verificado por el banco con el cliente.

- Autentificación en SSL:

El cliente no necesita autentificarse, una persona con acceso a números de tarjeta de crédito robados podría realizar cualquier tipo de compra por Internet. Este es precisamente el tipo de fraude más común y que causa mayores perdidas a las compañías de crédito.

- Autentificación

Por medio de este modelo se puede corroborar la autentificación del cliente, ya que es corroborada por el banco que acepta la transacción comercial, y el banco cuenta con información suficiente como para verificar la autenticidad de sus clientes, sin que el vendedor tenga noción de quien es en realidad. Y el banco no hace pública la identidad del cliente.

- No repudio en SSL:

Una vez finalizada la compra no existe ningún tipo de comprobante de compra por lo que cualquier protesta posterior carecerá de medios para su confirmación. Tampoco existe ningún documento firmado por lo que tanto el Cliente como el Vendedor o el Banco podrían negar su participación en la compra sin que existiera la posibilidad de probar lo contrario.

- No Repudio

En este modelo este criterio se satisface con la corroboración del cliente cuando el banco le pregunta sobre el monto de la transacción que está efectuando, quedando así registrado en las transacciones del banco y del vendedor el monto exacto. Además de que el vendedor puede emitir un recibo firmado con su llave privada al cliente, en éste se incluye el monto y la información utilizada para la autentificación en esa transacción.

5.7 CONCLUSIONES Y RECOMENDACIONES

El utilizar el protocolo Zero Knowledge en la autentificación de las transacciones asegura que el cliente se autentifique de una manera plana si brindar información sustancial de su autentificación, por lo que el banco no tendría duda de que es él el que hace la petición o la compra por Internet.

Este modelo puede ser utilizado por cualquier institución bancaria y comercial que necesite una autentificación segura de sus clientes.

Cabe mencionar que este modelo puede ser utilizado en las tarjetas inteligentes, para agilizar el cálculo de la autentificación. La combinación del modelo descrito favorece al comercio electrónico, ya que le brinda confianza al cliente de que tiene una autentificación única y que es difícil de falsificar, lo que hará que más usuarios puedan sentir la confianza para adquirir productos a través de Internet.

BIBLIOGRAFÍA

[1] An ACM Distinguished Dissertation 1989. Uses of Randommess In Algorithms and Protocols, Joe Kilian, Mit Press

[2] "Comercio Electrónico en Internet: el futuro ya está aquí" 1997, Juan Carlos Benítez Campoy

[3] "Securing Communication on the Intranet and Over the Internet" Julio 1996, Taher Elgamal, Netscape Communications Corporation.

[4] "Zero Knowledge Protocol and Small System", Aro95 Aronsson H, Helsinki University of Technologe, 1995

[5] "Acourse in Number Theory and Cryptography", Neal Koblitz, Gradute Texas in Mathematics, Springer

[6] "Técnicas Criptográficas de protección de datos", Amparo Fúster Sabater, Dolores de la Guís Martínez, Ra-ma, 1998

[7] "Zero Knowledge Proofs of identity", Feige, U. Fiat, and Shamir, Proceeding of the 19th annual ACM symposium

[8] "Seguridad informática, Técnicas Criptográficas", Pino Caballero Gil, Ra-ma, 1996, Madrid, España.

[9] "Applied Cryptography ", Schneier, Bruce; 1996, John Wiley & Sons, Inc.

Web Sites:

Especificación del Protocolo SSL la puede encontrar en:
[10]
http://www.netscape.com/newsref/std/SSL_old.html SSLv2
[11]
http://www.netscape.com/newsref/std/SSL.html SSLv3

[12] Lista de discusión en Netscape:
ssl-talk@netscape.com, y se puede suscribir a esta lista enviando un correo electrónico a ssl-talk-request@netscape.com y como título del mensaje "subscribe".

[13] La lista de FAQ la pueden encontrar en:
http://www.consensus.com/security/ssl-talk-faq.html

[14] Sitio de Criptografía y Seguridad de España
http://www.kiptopolis.com

[15] Sitio de Zero Knowledge Systems
http://www.zks.net

[16] Visa Internacional
http://www.visa.com

[17] MasterCard
http://www.mastercard.com

[18] SETco
http://www.setco.org

[19] RSA Data Security Inc.
http://www.rsa.com

[20] Terisa Systems
http://www.terisa.com

[21] Hp-Verifone
http://www.verifone.com

[22] Verisign, Inc
http://www.verisign.com

[23] GTE Cybertrust
http:// www.cybertrust.gte.com

[24] SAIC
http://www.saic.com

[25] IBM
http://www.internet.ibm/commercepoint

[26] Microsoft Corp.
http://www.microsoft.com

[27] Netscape Comunications Inc.
http://www.netscape.com

[28] First Virtual Holdings Incorporated
http://www.fv.com

[29] CyberCash, Inc.
http://www.cybercash.com

[30] DigiCash bv
http://www.digicash.com

[31] NetBill
http://www.ini.cmu.edu/netbill

[32] NetCheque/NetCash
http://nii.isi.edu/info/Netcheque

[33] CheckFree Corporation
http://www.checkfree.com

[34] The NetMarket Company
http://www.netmarket.com/sa/pages/home

[35] Open Market
http://www.openmarket.com

[36] Mondex International
http://www.mondex.com

[37] Financial Services Technology Consortium (FSTC)
http://www.fstc.org

[38] ComerceNet
http://www.commerce.com

[39] Commerce España
http://www.commerce.net

[40] AECE
http://www.aece.org

[41] ACE
http://www.ace.es

[42] Banesto
http://www.banesto.es

[43] Cyberpunks
http://www.cyberpunks.to

[45] SSLeay
http://www.ssleay.org

[46] Lista de correo sobre comercio electrónico
http://www.comercio-ELECTRÓNICO.org

[47] DELL
http://www.dell.com

[48] Telefónica Servicios Avanzados de Información (TSAI)
http://www.tsai.es

[49] Microsoft Wallet
http://www.microsoft.com/wallet/default.asp

[50] E-Wallets en Java
http://java.sun.com/products/commerce/index.html

[51] E-commerce Europa
http://www.ispo.cec.be/ecommerce/

[52] Electronic Commerce Association (ECA)
http://www.eca.org.uk/

[53] Directorio sobre dinero electrónico
http://ganges.cs.tcd.ie/mepeirce/project.html

ANEXO A: EJEMPLO DE MENSAJES

Ejemplo de mensajes

En este anexo se incluyen los formatos de los mensajes siguientes:

- **Certificado de Tarjetahabiente**
- **PInitReq**
- **PInitResData**
- **OIData**
- **PIData**
- **PResData**
- **AuthReqData**
- **AuthResData**
- **CapReqData**
- **CapResData**

Formato

Por cada mensaje se incluye la información siguiente:

- La estructura de datos y los nombre de cada campo.
- El contexto donde son aplicables. (Si una estructura de datos o campo es una construcción, los bytes del contenido no se muestran.)
- Codificación DER

Certificado del Tarjetahabiente

This is a UnsignedCertificate data structure. The total length of the data structure is 693 bytes.

Data Structures/Fields	Content	DER encoding
UnsignedCertificate		30 82 02 B1
.version	ver3(2)	A0 03 02 01 02
.serialNumber	22	02 01 16
.signature		30 0D
..algorithm	id-sha1-with-rsa-signature	06 09 2A 86 48 86 F7 0D 01 01 05
..parameters	null	05 00
.issuer		30 41
..countryName		31 0B 30 09
...type	id-at-countryName	06 03 55 04 06
...value	US	13 02 55 53
..organizationName		31 0E 30 0C
...type	id-at-organizationName	06 03 55 04 0A
...value	Brand	13 05 42 72 61 6E 64
..organizationUnitName		31 22 30 20
...type	id-at-organizationUnitName	06 03 55 04 0B
...value	Tarjetabiente Certificate CA	13 19 43 61 72 64 68 6F 6C 64 65 72 20 43 65 72 74 69 66 69 63 61 74 65 20 43 41
.validity		30 1E
..notBefore	961126222439Z	17 0D 39 36 31 31 32 36 32 32 32 34 33 39 5A
..notAfter	971126235900Z	17 0D 39 37 31 31 32 36 32 33 35 39 30 30 5A
.subject		30 53
..countryName		31 0B 30 09
....type	id-at-countryName	06 03 55 04 06
....value	US	13 02 55 53
..organizationName		31 0E 30 0C
...type	id-at-organizationName	06 03 55 04 0A
...value	Brand	13 05 42 72 61 6E 64
..organizationUnitName		31 0D 30 0B
...type	id-at-organizationUnitName	06 03 55 04 0B
...value	Bank	13 04 42 61 6E 6B
..commonName		31 25 30 23
...type	id-at-commonName	06 03 55 04 06
...value		13 1C 43 61 72 64 68 6F 6C 64 65 72 39 39 39 39 39 39 30 31 32 33 34 35 36 37 38 38 43 65

Tarjetahabiente Certificate, continuación

Data Structures/Fields	Content	DER encoding
.subjectPublicKeyInfo		30 82 01 22
..algorithm		30 0D
...algorithm	id-rsaEncryption	06 09 2A 86 48 86 F7 0D 01 01 01
...parameters	null	05 00

..subjectPublicKey		03 82 01 0F 00 30 82 01 0A 02 82 01 01 00 AC 0B 1D 55 77 4D 23 DE F7 0A 26 C6 BE 64 9E 9C 4F 0E B6 9B D2 19 43 95 3A 86 A0 D1 9A D4 FF 99 63 0D A3 F5 68 7D 5E F5 6C 9E 34 F5 ED 75 5C 47 FB 53 FE 9F 92 F0 E5 CE 95 60 44 EC D0 BA 25 A6 1F D1 65 7A BE B0 4D D6 85 97 AB 7D 2C AE FA 59 71 A1 AE 3C CD E9 DF 33 27 39 02 36 83 8E AE AB 8C 3F A0 C7 61 8D 78 22 24 CD 46 A1 25 84 43 B1 F7 5F B5 78 73 EE 1A 3E 4D D1 BB BA 06 64 D1 A4 FD 67 65 4D 06 F9 CA 28 AD 24 76 E3 99 7B 5F D1 A8 A0 3D 73 45 AB 52 30 53 02 1D 61 12 F1 F5 CA 94 97 FE 5C 15 DA F3 4A B0 5B 1F 9B 65 54 09 4A C1 EB AE D1 B7 6D E2 47 34 B5 C1 A1 49 A2 2D A5 76 F2 BD 02 0D D5 FF 9C 40 0E 34 CB A2 B1 D8 B0 BF 2C 2E 9B 11 C5 DD BB A6 5A 21 37 78 33 32 D3 DB 09 04 21 1F 65 04 25 FC CB A4 91 14 A4 09 E7 81 99 BD CF 4A C3 45 57 7E 59 B9 AE DB F5 74 A5 02 03 01 00 01

Tarjetahabiente Certificate, continuación

Data Structures/Fields	Content	DER encoding
.extensions		A3 81 B9 30 81 B6
..keyUsage		30 0E
...extrnID	id-ce-keyUsage	06 03 55 1D 0F
...critical	TRUE	01 01 FF
...extnValue digitalSignature(0)		04 04 03 02 01 80
..privateKeyUsagePeriod		30 2B
...extrnID	id-ce-privateKeyUsagePeriod	06 03 55 1D 10
...extnValue		04 24 30 22
....notBefore	19961126221453Z	80 0F 31 39 39 36 31 31 32 36 32 32 31 34 35 33 5A
....notAfter	19970826221453Z	81 0F 31 39 39 37 30 38 32 36 32 32 31 34 35 33 5A
..certificatePolicies		30 14
...extrnID	id-ce-certificatePolicies	06 03 55 1D 20
...critical	TRUE	01 01 FF
...extnValue		04 0A 30 08 30 06
....policyIdentifier	id-set-setQualifier	06 04 70 2A 07 06
..certificateType		30 10
...extrnID	id-set-certificateType	06 04 70 2A 07 01
...critical	TRUE	01 01 FF
...extnValue		04 05

	card(0)	03 03 07 80 00
..basicConstraints		30 0A
...extrnID		06 03 55 1D 13
...critical	TRUE	01 01 FF
...extnValue		04 00
..authorityKeyIdentifier		30 43
...extrnID		06 03 55 1D 23
...extnValue		04 3C 30 3A
....authorityCertIssuer		A1 34
.....directoryName		A4 32 30 30
......countryName		31 0B 30 09
......type	id-at-countryName	06 03 55 04 06
......value	US	13 02 55 53
.....organizationName		31 0E 30 0C
......type	id-at-organizationName	06 03 55 04 0A
......value	Brand	13 05 42 72 61 6E 64
.....organizationUnit Name		31 11 30 0F
......type	id-at-organizationUnitName	06 03 55 04 0B
......value	Brand CA	13 08 42 72 61 6E 64 20 43 41
....authorityCertSerial Number	4660	82 02 12 34

PInitReq

This is a **PInitReq** message. The total length of the message is 214 bytes.

Data Structures/Fields	Content	DER encoding
MessageWrapper		30 81 D3
.header		30 5D
..version	setVer1(1)	02 01 01
..revision	0	02 01 00
..date	19970514041853Z	18 0F 31 39 39 37 30 35 31 34 30 34 31 38 35 33 5A
..messageIDs		A0 16
...localID-C		80 14 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D
..rrpid		81 14 87 FB 2B 3D 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
..swIdent	SET Specification v1.0	1A 16 53 45 54 20 53 70 65 63 69 66 69 63 61 74 69 6F 6E 20 76 31 2E 30
.message		A0 72
..purchaseInitRequest		A0 81 6F 30 6D
...rrpid		04 14 87 FB 2B 3D 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
...language	en	1A 03 65 6E 20
...localID-C		04 14 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D
...chall-C		04 14 88 FB 2B 3D 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
...brandID	Brand:Product	1A 0D 42 72 61 6E 64 3A 50 72 6F 64 75 63 74
...bin	999999	12 06 39 39 39 39 39 39
...thumbs		A1 0D 30 0B
....digestAlgorithm		30 09
.....algorithm	id-sha1	06 05 2B 0E 03 02 1A
.....parameters	null	05 00

PInitResData

This is a **PInitResData** data structure to be signed in the **PInitRes** message. The total length of the data structure is 189 bytes.

Data Structures/Fields	Content	DER encoding
PInitResData		30 81 BA
.transIDs		30 42
..localID-C		04 14 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D
..xID		04 14 78 69 64 2D 78 69 64 2D 78 69 64 2D 78 69 64 2D
..pReqDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
..language	en	1A 03 65 6E 20
.rrpid		04 14 C9 36 C4 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
.chall-C		04 14 CA 36 C4 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
. peThumbs		A1 23 30 21
..digestAlgorithm		30 09
...algorithm	id-sha1	06 05 2B 0E 03 02 1A
...parameters	null	05 00
..thumbprint		04 14 A6 A3 30 4C BC 0E BE 1F 85 A9 56 14 77 7D 8D 25 1F EF 06 02
.thumbs		A2 0D 30 0B
..digestAlgorithm		30 09
...algorithm	id-sha1	06 05 2B 0E 03 02 1A
...parameters	null	05 00

OIData

This is an **OIData** data structure to be included in the **PReq** message. The total length of the data structure is 220 bytes.

Data Structures/Fields	Content	DER encoding
OIData		30 81 D9
.transIDs		30 42
..localID-C		04 14 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D
..xID		04 14 78 69 64 2D 78 69 64 2D 78 69 64 2D 78 69 64 2D
..pReqDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
..language	en	1A 03 65 6E 20
.rrpid		04 14 D1 65 CE 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
.chall-C		04 14 CA 36 C4 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
.hod		30 2C
..ddVersion	ddVer0(0)	02 01 00
..digestAlgorithm		30 09
...algorithm	id-sha1	06 05 2B 0E 03 02 1A
...parameters	null	05 00
..contentInfo		30 06
...contentType	id-set-content-OIData	06 04 70 2A 00 04
..digest		04 14 FB 7C C8 2F 80 B3 00 86 D2 60 84 29 36 69 05 70 CD CB 61 03
.odSalt		04 14 D2 65 CE 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
.chall-M		04 14 D3 66 CF 65 62 63 73 54 75 66 57 69 19 6E 60 2C 4D 2E 6F 10
.brandID	Brand:Product	1A 0D 42 72 61 6E 64 3A 50 72 6F 64 75 63 74

PIData

This is a **PIData** data structure to be encrypted in the **PReq** message. The total length of the data structure is 299 bytes.

Data Structures/Fields	Content	DER encoding
PIData		30 82 01 27
.piHead		30 81 DC
..transIDs		30 42
...localID-C		04 14 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D
...xID		04 14 78 69 64 2D 78 69 64 2D 78 69 64 2D 78 69 64 2D
...pReqDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
...language	en	1A 03 65 6E 20
..inputs		30 3B
...hod		30 2C
....ddVersion	ddVer0(0)	02 01 00
....digestAlgorithm		30 09
.....algorithm	id-sha1	06 05 2B 0E 03 02 1A
.....parameters	null	05 00
....contentInfo		30 06
.....contentType	id-set-content-HODInput	06 04 70 2A 00 08
....digest		04 14 FB 7C C8 2F 80 B3 00 86 D2 60 84 29 36 69 05 70 CD CB 61 03
...purchAmt		30 0B
....currency	840(US)	02 02 08 40
....amount	3059	02 02 0B B3
....amtExp10	-2	02 01 FE
..merchantID	MerchantID	13 0A 4D 65 72 63 68 61 6E 74 49 44
..transStain		04 14 18 29 34 4D 58 69 74 2D 38 49 24 D2 86 96 46 D2 88 79 74 3D
..swIdent	SET Specification v1.0	1A 16 53 45 54 20 53 70 65 63 69 66 69 63 61 74 69 6F 6E 20 76 31 2E 30
..acqBackInfo		A1 1F 30 1D
...backAlgID		30 11
....algorithm	id-desCBC	06 05 2B 0E 03 02 07
....parameters		04 08 CE 64 61 62 63 64 65 66

PIData, continuación

DER encoding (continuación)

Data Structures/Fields	Content	DER Encoding
...backKey		04 08 42 52 69 1F 4C A7 9B 0E
.panData		30 46
..pan	9999990123456788	12 10 39 39 39 39 39 39 30 31 32 33 34 35 36 37 38 38

..cardExpiry	199901	12 06 31 39 39 39 30 31
..panSecret		04 14 70 61 6E 73 65 63 72 65 74 70 61 6E 73 65 63 72 65 74 70 61
..exNonce		04 14 D3 65 CE 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70

PResData

This is a **PResData** data structure to be signed the **PRes** message. The total length of the data structure is 178 bytes.

Data Structures/Fields	Content	DER encoding
PResData		30 81 AF
.transIDs		30 42
..localID-C		04 14 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D
..xID		04 14 78 69 64 2D 78 69 64 2D 78 69 64 2D 78 69 64 2D
..pReqDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
..language	en	1A 03 65 6E 20
.rrpid		04 14 D1 65 CE 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
.chall-C		04 14 CA 36 C4 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
.pResPayloadSeq		30 3D 30 3B
..completionCode	capturePerformed(4)	0A 01 04
..results		30 36
...authStatus		A1 19
....authDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
....authCode	approved(0)	0A 01 00
....authRatio	1	09 03 80 00 01
...capStatus		A2 19
....capDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
....capCode	success(0)	0A 01 00
....capRatio	1	09 03 80 00 01

AuthReqData

This is an **AuthReqData** data structure to be encrypted in the **AuthReq** message. The total length of the data structure is 258 bytes.

Data Structures/Fields	Content	DER encoding
AuthReqData		30 81 FF
.authReqItem		30 81 FC
..authTags		30 7B
...authRRTags		30 35
....rrpid		04 14 D1 65 CE 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
....merTermIDs		30 0C
.....merchantID	MerchantID	13 0A 4D 65 72 63 68 61 6E 74 49 44
....currentDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
...transIDs		30 42
....localID-C		04 14 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D
....xID		04 14 78 69 64 2D 78 69 64 2D 78 69 64 2D 78 69 64 2D
....pReqDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
....language	en	1A 03 65 6E 20
..checkDigests		A0 5C
...hOIData		30 2C
....ddVersion	ddVer0(0)	02 01 00
....digestAlgorithm		30 09
.....algorithm	id-sha1	06 05 2B 0E 03 02 1A
.....parameters	null	05 00
....contentInfo		30 06
.....contentType	id-set-content-OIData	06 04 70 2A 00 04
....digest		04 14 8F 34 3E AC 28 EB BF 6C B0 38 CD C0 93 79 E1 23 70 85 3C A2
...hod2		30 2C
....ddVersion	ddVer0(0)	02 01 00
....digestAlgorithm		30 09
.....algorithm	id-sha1	06 05 2B 0E 03 02 1A
.....parameters	null	05 00
....contentInfo		30 06
.....contentType	id-set-content-HODInput	06 04 70 2A 00 08

AuthReqData, continuación

DER encoding (continuación)

Data Structures/Fields	Content	DER encoding
....digest		04 14 FB 7C C8 2F 80 B3 00 86 D2 60 84 29 36 69 05 70 CD CB 61 03
..mThumbs		A1 0B

...digestAlgorithm		30 09
....algorithm	id-sha1	06 05 2B 0E 03 02 1A
....parameters	null	05 00
..authReqPayload		30 12
...subsequentAuthInd	FALSE	01 01 00
...authReqAmt		30 0B
....currency	840(US)	02 02 08 40
....amount	3059	02 02 0B B3
....amtExp10	-2	02 01 FE
...merchData		30 00

AuthResData

This is an **AuthResData** data structure to be encrypted in the **AuthRes** message. The total length of the data structure is 163 bytes.

Data Structures/Fields	Content	DER encoding
AuthResData		30 81 A0
.authTags		30 7B
..authRRTags		30 35
...rrpid		04 14 D1 65 CE 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
...merTermIDs		30 0C
....merchantID	MerchantID	13 0A 4D 65 72 63 68 61 6E 74 49 44
...currentDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
..transIDs		30 42
...localID-C		04 14 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D
...xID		04 14 78 69 64 2D 78 69 64 2D 78 69 64 2D 78 69 64 2D
...pReqDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
...language	en	1A 03 65 6E 20
.authResPayload		30 21
..authHeader		30 1F
...authAmt		30 0B
....currency	840(US)	02 02 08 40
....amount	3059	02 02 0B B3
....amtExp10	-2	02 01 FE
...authCode	approved(0)	0A 01 00
...responseData		30 0D
....authValCodes		A0 08
.....approvalCode	567891	80 06 35 36 37 38 39 31
....respReason	issuer(0)	81 01 00

CapReqData

This is a **CapReqData** data structure to be encrypted in the **CapReq** message. The total length of the data structure is 477 bytes.

Data Structures/Fields	Content	DER encoding
CapReqData		30 82 01 D9
.capRRTags		30 35
..rrpid		04 14 D1 65 FB 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
..merTermIDs		30 0C
...merchantID	MerchantID	13 0A 4d 65 72 63 68 61 6e 74 49 44
..currentDate	19970509175510Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 35 31 30 5a
.mThumbs		A0 0B 30 09
..digestAlgorithm		06 05 2b 0e 03 02 1a
..parameters	null	05 00
.capItemSeq		30 82 01 91 30 82 01 8D
..transIDs		30 42
...localID-C		04 14 6c 69 64 63 2d 6c 69 64 63 2d 6c 69 64 63 2d 6c 69 64 63 2d
...xID		04 14 78 69 64 2d 78 69 64 2d 78 69 64 2d 78 69 64 2d
...pReqDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
...language	19970509175416Z	1A 03 65 6E 20
..capPayload		30 82 01 45
...capDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
...capReqAmt		30 0B
....currency	840(US)	02 02 08 40
....amount	3059	02 02 0B B3
....amtExp10	-2	02 01 FE

CapReqData, continuación

DER encoding (continuación)

Data Structures/Fields	Content	DER encoding
...authReqItem		A0 81 FF 30 81 FC
....authTags		30 7B
.....authRRTags		30 35
......rrpid		04 14 D1 65 CE 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
......merTermIDs		30 0C
.......merchantID	MerchantID	13 0A 4D 65 72 63 68 61 6E 74 49 44
......currentDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
.....transIDs		30 42
......localID-C		04 14 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D 6C 69 64 63 2D
......xID		04 14 78 69 64 2D 78 69 64 2D 78 69 64 2D 78 69 64 2D 78 69 64 2D
......pReqDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
......language	en	1A 03 65 6E 20
....checkDigests		A0 5C
.....hOIData		30 2C
......ddVersion	ddVer0(0)	02 01 00
......digestAlgorithm		30 09
.......algorithm	id-sha1	06 05 2B 0E 03 02 1A
.......parameters	null	05 00
......contentInfo		30 06
.......contentType	id-set-content-OIData	06 04 70 2A 00 04
......digest		04 14 8F 34 3E AC 28 EB BF 6C B0 38 CD C0 93 79 E1 23 70 85 3C A2
.....hod2		30 2C
......ddVersion	ddVer0(0)	02 01 00
......digestAlgorithm		30 09
.......algorithm	id-sha1	06 05 2B 0E 03 02 1A
.......parameters	null	05 00
......contentInfo		30 06
.......contentType	id-set-content-HODInput	06 04 70 2A 00 08

CapReqData, continuación

DER encoding (continuación)

Data Structures/Fields	Content	DER encoding
......digest		04 14 FB 7C C8 2F 80 B3 00 86 D2 60 84 29 36 69 05 70 CD CB 61 03
....mThumbs		A1 0B
.....digestAlgorithm		30 09
......algorithm	id-sha1	06 05 2B 0E 03 02 1A
......parameters	null	05 00
....authReqPayload		30 12
.....subsequentAuthInd	FALSE	01 01 00
.....authReqAmt		30 0B
......currency	840(US)	02 02 08 40
......amount	3059	02 02 0B B3
......amtExp10	-2	02 01 FE
.....merchData		30 00
...authResPayload		A1 23 30 21
....authHeader		30 1F
.....authAmt		30 0B
......currency	840(US)	02 02 08 40
......amount	3059	02 02 0B B3
......amtExp10	-2	02 01 FE
.....authCode	approved(0)	0A 01 00
.....responseData		30 0D
......authValCodes		A0 08
.......approvalCode	567891	80 06 35 36 37 38 39 31
......respReason	issuer(0)	81 01 00

CapResData

This is a **CapResData** data structure to be encrypted in the **CapRes** message. The total length of the data structure is 152 bytes.

Data Structures/Fields	Content	DER encoding
CapResData		30 81 95
.capRRTags		30 35
..rrpid		04 14 D1 65 FB 64 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
..merTermIDs		30 0C
...merchantID	MerchantID	13 0A 4d 65 72 63 68 61 6e 74 49 44
..currentDate	19970509175510Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 35 31 30 5a
.capResItemSeq		30 5C 30 5A
..transIDs		30 42
...localID-C		04 14 6c 69 64 63 2d 6c 69 64 63 2d 6c 69 64 63 2d 6c 69 64 63 2d
...xID		04 14 78 69 64 2d 78 69 64 2d 78 69 64 2d 78 69 64 2d
...pReqDate	19970509175416Z	18 0F 31 39 39 37 30 35 30 39 31 37 35 34 31 36 5A
...language	19970509175416Z	1A 03 65 6E 20
..capResPayload		30 14
...capCode	success(0)	0A 01 00
...capAmt		30 0B
....currency	840(US)	02 02 08 40
....amount	3059	02 02 0B B3
....amtExp10	-2	02 01 FE
...batchID	102	80 02 01 02

ANEXO B. ESTRUCTURA DEL PROTOCOLO SSL 3.0

B. Constantes del Protocolo SSL

En esta sección se describen las estructuras y las constantes del protocolo.

B.1 Asignación de un puerto reservado

La utilización actual del protocolo SSL, utilizando TCP/IP, tiene su origen en la tecnología de redes. La IANA ha reservado los siguientes puertos para usarlos en conjunto con los protocolos IP y el SSL.

443 - Reservado para el uso del Protocolo de Hypertexto con SSL (*https*).
465 - Reservado para el uso del SMTP con SSL (*ssmtp*).
563 - Reservado para el uso del News y SSL (*snntp*).

B.1.1 Nivel de Registros

```
struct {
        uint8 major, minor;
} ProtocolVersion;

ProtocolVersion version = { 3,0 };      /* Define la version SSL 3.0 */

enum {
        change_cipher_spec(20), alert(21), handshake(22),
        application_data(23), (255)
} ContentType;

struct {
        ContentType type;
        ProtocolVersion version;
        uint16 length;
        opaque fragment[SSLPlaintext.length];
} SSLPlaintext;

struct {
        ContentType type;              /* el mismo SSLPlaintext.type */
        ProtocolVersion version;       /* el mismo SSLPlaintext.version */
        uint16 length;
```

```
        opaque fragment[SSLCompressed.length];
} SSLCompressed;

struct {
        ContentType type;
        ProtocolVersion version;
        uint16 length;
        select (CipherSpec.cipher_type) {
            case stream: GenericStreamCipher;
            case block: GenericBlockCipher;
        } fragment;
} SSLCiphertext;

stream-ciphered struct {
          opaque content[SSLCompressed.length];
          opaque MAC[CipherSpec.hash_size];
} GenericStreamCipher;

block-ciphered struct {
        opaque content[SSLCompressed.length];
        opaque MAC[CipherSpec.hash_size];
        uint8 padding[GenericBlockCipher.padding_length];
        uint8 padding_length;
} GenericBlockCipher;
```

B.2 Cambio de cifrado del mensaje esperado

```
struct {
        enum { change_cipher_spec(1), (255) } type;
} ChangeCipherSpec;
```

B.3 Mensajes de Alerta

```
enum { warning(1), fatal(2), (255) } AlertLevel;
enum {
        close_notify(0),
        unexpected_message(10),
        bad_record_mac(20),
        decompression_failure(30),
        handshake_failure(40), no_certificate(41), bad_certificate(42),
            unsupported_certificate(43), certificate_revoked(44),
```

```
            certificate_expired(45), certificate_unknown(46),
            illegal_parameter (47),
        (255)
} AlertDescription;

struct {
        AlertLevel level;
        AlertDescription description;
} Alert;
```

B.4 Protocolo Handshake

```
enum {
        hello_request(0), client_hello(1), server_hello(2),
        certificate(11), server_key_exchange (12), certificate_request(13),
        server_done(14), certificate_verify(15), client_key_exchange(16),
        finished(20), (255)
} HandshakeType;

struct {
        HandshakeType msg_type;          /* type of handshake message */
        uint24 length;  /* # bytes in handshake msg body */
        select (HandshakeType) {
            case hello_request: HelloRequest;
            case client_hello: ClientHello;
            case server_hello: ServerHello;
            case certificate: Certificate;
            case server_key_exchange: ServerKeyExchange;
            case certificate_request: CertificateRequest;
            case server_done: ServerHelloDone;
            case certificate_verify: CertificateVerify;
            case client_key_exchange: ClientKeyExchange;
            case finished: Finished;
        } body;
} Handshake;
```

B.4.1 Mensajes de Inicialización (Hello)

```
struct { } HelloRequest;

struct {
```

```
   uint32 gmt_unix_time;
   opaque random_bytes[28];
} Random;

opaque SessionID<0..32>;
uint8 CipherSuite[2];
enum { null(0), (255) } CompressionMethod;

struct {
        ProtocolVersion client_version;
        Random random;
        SessionID session_id;
        CipherSuite cipher_suites<0..2^16-1>;
        CompressionMethod compression_methods<0..2^8-1>;
} ClientHello;

struct {
        ProtocolVersion server_version;
        Random random;
        SessionID session_id;
        CipherSuite cipher_suite;
        CompressionMethod compression_method;
} ServerHello;
```

B.4.2 Autentificación en el servidor e intercambio de mensajes cifrados

```
opaque ASN.1Cert<2^24-1>;

struct {
        ASN.1Cert certificate_list<1..2^24-1>;
} Certificate;

enum { rsa, diffie_hellman, fortezza_dms } KeyExchangeAlgorithm;

struct {
        opaque RSA_modulus<1..2^16-1>;
        opaque RSA_exponent<1..2^16-1>;
} ServerRSAParams;

struct {
        opaque DH_p<1..2^16-1>;
        opaque DH_g<1..2^16-1>;
        opaque DH_Ys<1..2^16-1>;
```

```
} ServerDHParams;

struct {
        opaque r_s [128]
} ServerFortezzaParams

struct {
        select (KeyExchangeAlgorithm) {
            case diffie_hellman:
                  ServerDHParams params;
                  Signature signed_params;
            case rsa:
                  ServerRSAParams params;
                  Signature signed_params;
            case fortezza_dms:
                  ServerFortezzaParams params;
        };
} ServerKeyExchange;

enum { anonymous, rsa, dsa } SignatureAlgorithm;

digitally-signed struct {
        select(SignatureAlgorithm) {
             case anonymous: struct { };
             case rsa:
                  opaque md5_hash[16];
                  opaque sha_hash[20];
             case dsa:
                  opaque sha_hash[20];
        };
} Signature;

enum {
        RSA_sign(1), DSS_sign(2), RSA_fixed_DH(3), DSS_fixed_DH(4),
        RSA_ephemeral_DH(5), DSS_ephemeral_DH(6), Fortezza_dms(20), (255)
} CertificateType;

opaque DistinguishedName<3..2^16-1>;

struct {
        CertificateType certificate_types<1..2^8-1>;
        DistinguishedName certificate_authorities<3..2^16-1>;
} CertificateRequest;
struct { } ServerHelloDone;
```

B.5 Autentificación en el cliente e intercambio de mensajes cifrados

```
struct {
        select (KeyExchangeAlgorithm) {
            case rsa: EncryptedPreMasterSecret;
            case diffie_hellman: DiffieHellmanClientPublicValue;
            case fortezza_dms: FortezzaKeys;
        } exchange_keys;
} ClientKeyExchange;

struct {
        ProtocolVersion client_version;
        opaque random[46];
} PreMasterSecret;

struct {
        public-key-encrypted PreMasterSecret pre_master_secret;
} EncryptedPreMasterSecret;

struct {
        opaque y_c<0..128>;
        opaque r_c[128];
        opaque y_signature[20];
        opaque wrapped_client_write_key[12];
        opaque wrapped_server_write_key[12];
        opaque client_write_iv[24];
        opaque server_write_iv[24];
        opaque master_secret_iv[24];
        opaque encrypted_preMasterSecret[48];
} FortezzaKeys;

enum { implicit, explicit } PublicValueEncoding;

struct {
        select (PublicValueEncoding) {
            case implicit: struct {};
            case explicit: opaque DH_Yc<1..2^16-1>;
        } dh_public;
} ClientDiffieHellmanPublic;

        struct {
             Signature signature;
        } CertificateVerify;
```

B.5.1 Mensaje de Finalización Handshake

```
struct {
        opaque md5_hash[16];
        opaque sha_hash[20];
} Finished;
```

B.6 La Suite de Cifrado

Los siguientes valores definen la `CipherSuite` de los códigos utilizados por los mensajes de inicialización tanto en el cliente como en el servidor.

La `CipherSuite` define las especificaciones de cifrado soportadas por el protocolo SSL versión 3.0.

```
CipherSuite SSL_NULL_WITH_NULL_NULL                    = { 0x00,0x00 };
```

Las siguientes definiciones de `CipherSuite` requieren que el servidor provea un certificado RSA para que pueda ocurrir el intercambio de llaves. El servidor es capaz de pedir un certificado de RSA o DES para certificar el mensaje.

```
CipherSuite SSL_RSA_WITH_NULL_MD5                      = { 0x00,0x01 };
CipherSuite SSL_RSA_WITH_NULL_SHA                      = { 0x00,0x02 };
CipherSuite SSL_RSA_EXPORT_WITH_RC4_40_MD5             = { 0x00,0x03 };
CipherSuite SSL_RSA_WITH_RC4_128_MD5                   = { 0x00,0x04 };
CipherSuite SSL_RSA_WITH_RC4_128_SHA                   = { 0x00,0x05 };
CipherSuite SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5         = { 0x00,0x06 };
CipherSuite SSL_RSA_WITH_IDEA_CBC_SHA                  = { 0x00,0x07 };
CipherSuite SSL_RSA_EXPORT_WITH_DES40_CBC_SHA          = { 0x00,0x08 };
CipherSuite SSL_RSA_WITH_DES_CBC_SHA                   = { 0x00,0x09 };
CipherSuite SSL_RSA_WITH_3DES_EDE_CBC_SHA              = { 0x00,0x0A };
```

Las siguientes definiciones `CipherSuite` son utilizadas para la autentificación del servidor y opcionalmente pueden ser utilizadas para autentificar al cliente. Éstas reciben el nombre de Diffie-Hellman y son denotadas por un *DH* en el cual el servidor de certificados contiene los parámetros Diffie-Hellman firmados por una autoridad de certificados (CA). *DHE* denota ephemeral Diffie-Hellman, donde los parámetros Diffie-Hellman son firmados por un certificado DSS o RSA, el cual ha sido firmado por un CA. El algoritmo de firma utilizado es especificado después del parámetro DH o DHE. En todos los casos, el cliente debe tener el mismo tipo de certificado, y debe utilizar los parámetros Diffie-Hellman seleccionados por el servidor.

```
CipherSuite SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA       = { 0x00,0x0B };
```

```
CipherSuite SSL_DH_DSS_WITH_DES_CBC_SHA                 = { 0x00,0x0C };
CipherSuite SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA            = { 0x00,0x0D };
CipherSuite SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA        = { 0x00,0x0E };
CipherSuite SSL_DH_RSA_WITH_DES_CBC_SHA                 = { 0x00,0x0F };
CipherSuite SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA            = { 0x00,0x10 };
CipherSuite SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA       = { 0x00,0x11 };
CipherSuite SSL_DHE_DSS_WITH_DES_CBC_SHA                = { 0x00,0x12 };
CipherSuite SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA           = { 0x00,0x13 };
CipherSuite SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA       = { 0x00,0x14 };
CipherSuite SSL_DHE_RSA_WITH_DES_CBC_SHA                = { 0x00,0x15 };
CipherSuite SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA           = { 0x00,0x16 };
```

Las siguientes definiciones son utilizadas para una comunicación Diffie-Hellman anónima en la cual ninguna de las partes se autentifica. Se hace notar que este modelo es sumamente vulnerable a los ataques y no es totalmente aceptado.

```
CipherSuite SSL_DH_anon_EXPORT_WITH_RC4_40_MD5          = { 0x00,0x17 };
CipherSuite SSL_DH_anon_WITH_RC4_128_MD5                = { 0x00,0x18 };
CipherSuite SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA       = { 0x00,0x19 };
CipherSuite SSL_DH_anon_WITH_DES_CBC_SHA                = { 0x00,0x1A };
CipherSuite SSL_DH_anon_WITH_3DES_EDE_CBC_SHA           = { 0x00,0x1B };
```

La siguiente definición pertence al Fortezza token.

```
CipherSuite SSL_FORTEZZA_DMS_WITH_NULL_SHA              = { 0X00,0X1C };
CipherSuite SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA      = { 0x00,0x1D };
```

Nota: Todas las definiciones que tengan como primer byte un 0xFF son consideradas como privadas y pueden ser utilizadas para definir algoritmos experimentales.

B.7 La especificación CipherSpec

Una especificación de cifrado identifica una CipherSpec. Esta estructura es parte del estado de la sesión SSL. La especificación CipherSpec incluye:

```
enum { stream, block } CipherType;
enum { true, false } IsExportable;
enum { null, rc4, rc2, des, 3des, des40, fortezza } BulkCipherAlgorithm;
enum { null, md5, sha } MACAlgorithm;

struct {
        BulkCipherAlgorithm bulk_cipher_algorithm;
        MACAlgorithm mac_algorithm;
        CipherType cipher_type;
```

```
        IsExportable is_exportable
        uint8 hash_size;
        uint8 key_material;
        uint8 IV_size;
} CipherSpec;
```

ANEXO C

C.1 CÓDIGO DE UN EJEMPLO DEL PROTOCOLO EN JAVA

Código en Java para calcular las Raices

```
import java.net.*;
import java.io.*;

public class CalculaRaiz{

    public static void main(String[] args) {
        double primo1 = (new Integer(args[0]).longValue());
        double primo2 = (new Integer(args[1]).longValue());

        System.out.println(args[0]);
        System.out.println(args[1]);
        double v[];
        double v1[];


        double n = primo1 * primo2;


        System.out.println("n:"+n);
        for (long i=1; i < n; i++){
            double vmodp1 = (i) % primo1;
            double vmodp2 = (i) % primo2;
            double r1 = 0;
            double r2 = 0;
            double r3 = 0;
            double r4 = 0;
            double sqrtvMODp1Modp1 = 0;
            double sqrtvMODp2Modp2 = 0;
            double invp1MODp2 = 0;
            double invp2MODp1 = 0;
            double invvMODn = 0;
            double raizv1 = 0;
            double s = 0;
            double vmodn = (i * i) % n;
```

```
//Cálculo de la raiz cuadrada de vmodp1 mod p1
double x = 1;
while (x < n)
{
    sqrtvMODp1Modp1 = (x * x) % primo1;
    if (sqrtvMODp1Modp1 == vmodp1){
        sqrtvMODp1Modp1 = x;
        x=i;
        break;
    }else{ sqrtvMODp1Modp1 = 0; }
    x++;
}

//Cálculo de la raiz cuadrada de vmodp2 mod p2
double x2 = 1;
while (x2 < n)
{
    sqrtvMODp2Modp2 = (x2 * x2) % primo2;
    if (sqrtvMODp2Modp2 == vmodp2){
        sqrtvMODp2Modp2 = x2;
        x2=i;
        break;
    }else { sqrtvMODp2Modp2 = 0; }
    x2++;
}

//Cálculo del inverso de primo1 mod primo2
double x3 = 1;
while (x3 < n)
{
    invp1MODp2 = (x3 * primo1) % primo2;
    if (invp1MODp2 == 1){
        invp1MODp2 = x3;
        x3=i;
        break;
    }else { invp1MODp2 = 0; }
    x3++;
}

//Cálculo del inverso de primo2 mod primo1
double x4 = 1;
while (x4 < n)
{
    invp2MODp1 = (x4 * primo2) % primo1;
```

```
    if (invp2MODp1 == 1){
        invp2MODp1 = x4;
        x4=i;
        break;
    }else { invp2MODp1 = 0; }
    x4++;
}

//Cálculo de la v ^ -1
double x5 = 1;
while (x5 < n)
{
    invvMODn = (x5 * i) % n;
    if (invvMODn == 1){
        invvMODn = x5;
        x5=i;
        break;
    }else { invvMODn = 0; }
    x5++;
}

//Cálculo de la v ^ -1
double x6 = 1;
while (x6 < n)
{
    s = (x6 * x6) % n;
    if (s == invvMODn){
        s = x6;
        x6=i;
        break;
    }else { s = 0; }
    x6++;
}



if ((sqrtvMODp1Modp1 != 0) & (sqrtvMODp2Modp2 != 0) & (invvMODn !=0)){
    System.out.print("v:"+i+" ");
    r1 = (sqrtvMODp1Modp1*primo2*invp2MODp1)+(sqrtvMODp2Modp2*primo1*invp1MODp2);

    r1 = r1 % n;
    if (r1 < 0) { r1 = n + r1; }
    r2 = (sqrtvMODp2Modp2*primo1*invp1MODp2)-(sqrtvMODp1Modp1*primo2*invp2MODp1);
    r2 = r2 % n;
    if (r2 < 0) { r2 = n + r2; }
```

```
                r3 = (sqrtvMODp1Modp1*primo2*invp2MODp1)-(sqrtvMODp2Modp2*primo1*invp1MODp2);
                r3 = r3 % n;
                if (r3 < 0) { r3 = n + r3; }
                r4 = (sqrtvMODp1Modp1*(-1)*primo2*invp2MODp1)-(sqrtvMODp2Modp2*primo1*invp1MODp2);
                r4 = r4 % n;
                if (r4 < 0) { r4 = n + r4; }
                System.out.println(r1+" "+r2+" "+r3+" "+r4+" "+invvMODn+" "+s+" ");
            }
        }
    }
}
```

Código del cliente para la autentificación.

```
// Desarrollado por Gabriel Barrera D. C 1999
// Cliente que prueba el Servidor
// lanza varios threads

import java.net.*;
import java.io.*;
import java.awt.TextArea;

class ClienteThread extends Thread {
 static final int port = 8187;
 private Socket socket;
 BufferedReader in;
 PrintWriter out;
 private TextArea textArea = null;
 private static int counter = 0;
 private int id = counter++;
 static final int maxthreads = 1;
 static int threadcount = 0;

//
// Cliente para correrlo desde DOS
//

public  ClienteThread(InetAddress addr) {
    System.out.println("Transformando Informacion " + id);
    threadcount++;
    try {
     InetAddress addr2 = InetAddress.getByName("148.241.27.43");
     socket = new Socket(addr2, port);
     in =
        new BufferedReader(
```

```
            new InputStreamReader(
                socket.getInputStream()));
        // Enable auto-flush:
        out =
            new PrintWriter(
             new BufferedWriter(
                new OutputStreamWriter(
                 socket.getOutputStream())), true);
        start();
       } catch(Exception e) {
        e.printStackTrace();
       }
     }

//
// Cliente para correrlo desde el Browser
//

public ClienteThread(InetAddress addr, TextArea textAreaParam) {

       textArea=textAreaParam;

       textArea.append("Transformando Informacion" + id+"\n");
       threadcount++;
       try {
        InetAddress addr2 = InetAddress.getByName("148.241.27.43");
        socket = new Socket(addr2, port);
        in =
            new BufferedReader(
             new InputStreamReader(
                socket.getInputStream()));
        // Enable auto-flush:
        out =
            new PrintWriter(
             new BufferedWriter(
                new OutputStreamWriter(
                 socket.getOutputStream())), true);
        start();
       } catch(Exception e) {
        e.printStackTrace();
       }

}
public void run() {
          try {
```

```
            for(int i = 0; i < 5; i++) {

            /*
            if (textArea!=null)
                textArea.append("Making client " + id+"\n");
            else
                System.out.println(str);
            }*/
            long r = 8 * 2;
            long n = 35;
            long x = (r * r) % n;
            out.println(x);

            String binario = in.readLine();
            System.out.println(binario);

            long y = ( r * (3) * (4) * (1) * (8)) % n;
            out.println(y);

            String autentifica = in.readLine();
            System.out.println(autentifica);
            }
            out.println("END");
            in.close();
            out.close();
            socket.close();
            threadcount--; // Ending this thread

        } catch(IOException e) {
            e.printStackTrace();
        }
    }
}


// Desarrollado por Gabriel Barrera D. C 1999
// Clientedos.java
// Cliente que verifica
// a través de varios clientes.
import java.net.*;
import java.io.*;

public class clientedos {
  public static void main(String args[]) {
    try {
```

```
      InetAddress addr =
         InetAddress.getByName(null);
      for(int k=0; k < 4; k++) {
         if(ClienteThread.threadcount <
           ClienteThread.maxthreads)
          new ClienteThread(addr);
         Thread.currentThread().sleep(10);
      }
     } catch(Exception e ) {
      e.printStackTrace();
     }
  }
}
```

Código del Servidor de Autentificación.

```
// AtiendeUnCliente.java
//
// Lic. Gabriel Barrera Delgadillo. 1999
// Código del servicio que ofrece el servidor.
//Para cualquier NÚMERO de clientes.

import java.io.*;
import java.net.*;

class AtiendeUnCliente extends Thread {
  private Socket socket;
  private BufferedReader in;
  private PrintWriter out;
  AtiendeUnCliente(Socket s) {
     socket = s;
     try {
      in =
         new BufferedReader(
          new InputStreamReader(
             socket.getInputStream()));
      // Habilitando auto-flush:
      out =
         new PrintWriter(
          new BufferedWriter(
             new OutputStreamWriter(
              socket.getOutputStream())), true);
     } catch(IOException e) {
      e.printStackTrace();
     }
```

```
    start(); // Llama run()
  }
  public void run() {
    try {
     while (true) {
        String str = in.readLine();
        if (str.equals("END")) break;

        System.out.println("Recibe r^2 mod n= " + str);

        double x = (new Integer(str).doubleValue());

        out.println("1101");

        String str2 = in.readLine();
        System.out.println("Recibe x2= " + str2);

        double y = (new Integer(str2).doubleValue());

        double z = ((y*y) * (4) * (11) * (1) * (29)) % 35;

        if (z == x) {out.println("Aceptado");}
        else {out.println("rechazado");}


     }
     System.out.println("Cerrando conexión ...");
     in.close();
     out.close();
     socket.close();
    } catch (IOException e) {
     e.printStackTrace();
    }
  }
}



// Desarrollado por Lic. Gabriel Barrera D. C 1999
// ServidorMultiAutentificador
// serverdos.java
// requiere de ServeOneJabber
// Es un servidor que puede atender a muchos clientes
//
```

```
import java.io.*;
import java.net.*;

public class serverdos {
 static final int port = 8187;
 public static void main(String[] args ) {
     try {
      ServerSocket s = new ServerSocket(port);
      System.out.println("El servidor está Listo!!!, esperando peticiones....");
      while(true) {
          // Espera hasta que una conexión ocurra:
          Socket socket = s.accept();
          new AtiendeUnCliente(socket);
      }
     } catch(Exception e) {
      e.printStackTrace();
     }
 }
}



//Desarrollado por Gabriel Barrera 1999.
//Applet para Netscape 4.x

import netscape.security.*;
import java.applet.*;
import java.awt.*;
import java.net.*;
import java.io.*;


/**
Para ejecutarlo es necesario incluir una preferencia de usuario en el
archivo prefs.js, esta preferencia es:

user_pref("signed.applets.codebase_principal_support", true);
*/


public class AppletClienteDos extends Applet implements java.awt.event.ActionListener {
     private Button ivjButton1 = null;
     private Panel ivjPanel1 = null;
     private TextArea ivjTextArea1 = null;
     private TextField ivjTextField1 = null;
```

```
/**
 * Metodo que controla los eventos.
 * @param e java.awt.event.ActionEvent
 */

public void actionPerformed(java.awt.event.ActionEvent e) {
    // user code begin {1}
    // user code end
    if ((e.getSource() == getButton1()) ) {
        connEtoC1(e);
    }
    // user code begin {2}
    // user code end
}



/**
 * Envia la peticion de autentificacion
 */
public void button1_ActionPerformed(java.awt.event.ActionEvent actionEvent) {

    try {
     PrivilegeManager.enablePrivilege("UniversalConnect");
     long var1 = (new Integer(ivjTextField1.getText())).longValue();
     System.out.println("var1:"+var1);
     InetAddress addr =
        InetAddress.getByName(null);
     for(int k=0; k < 4; k++) {
        if(ClienteThread.threadcount <
          ClienteThread.maxthreads)
         new ClienteThread(addr,ivjTextArea1);
        Thread.currentThread().sleep(10);
     }
    } catch(Exception e ) {
     e.printStackTrace();
    }

    return;
}
/**
 * connEtoC1:  (Button1.action.actionPerformed(java.awt.event.ActionEvent) -->
 * AppletClienteDos.button1_ActionPerformed(Ljava.awt.event.ActionEvent;)V)
 * @param arg1 java.awt.event.ActionEvent
 */
```

```
private void connEtoC1(java.awt.event.ActionEvent arg1) {
    try {
        // user code begin {1}
        // user code end
        this.button1_ActionPerformed(arg1);
        // user code begin {2}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {3}
        // user code end
        handleException(ivjExc);
    }
}
/**
 * Da la informacion del Applet.
 *
 */

public String getAppletInfo() {
    return "AppletClienteDos applet de autentificacion.";
}

/**
 * Return the Button1 property value.
 * @return java.awt.Button
 */

private Button getButton1() {
    if (ivjButton1 == null) {
        try {
            ivjButton1 = new java.awt.Button();
            ivjButton1.setName("Button1");
            ivjButton1.setBounds(254, 59, 56, 23);
            ivjButton1.setLabel("Verificar");
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    };
    return ivjButton1;
}
```

```
/**
 * regresa las propiedades del Panel1.
 * @return java.awt.Panel
 */
private Panel getPanel1() {
    if (ivjPanel1 == null) {
        try {
            ivjPanel1 = new java.awt.Panel();
            ivjPanel1.setName("Panel1");
            ivjPanel1.setLayout(null);
            getPanel1().add(getTextField1(), getTextField1().getName());
            getPanel1().add(getTextArea1(), getTextArea1().getName());
            getPanel1().add(getButton1(), getButton1().getName());
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    };
    return ivjPanel1;
}
/**
 * regresa las propiedades del TextArea1.
 * @return java.awt.TextArea
 */
private TextArea getTextArea1() {
    if (ivjTextArea1 == null) {
        try {
            ivjTextArea1 = new java.awt.TextArea();
            ivjTextArea1.setName("TextArea1");
            ivjTextArea1.setBounds(68, 107, 248, 137);
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    };
    return ivjTextArea1;
}
/**
 * Return the TextField1 property value.
```

```
 * @return java.awt.TextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private TextField getTextField1() {
    if (ivjTextField1 == null) {
        try {
            ivjTextField1 = new java.awt.TextField();
            ivjTextField1.setName("TextField1");
            ivjTextField1.setBounds(67, 57, 163, 30);
            // user code begin {1}
            // user code end
        } catch (java.lang.Throwable ivjExc) {
            // user code begin {2}
            // user code end
            handleException(ivjExc);
        }
    };
    return ivjTextField1;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(Throwable exception) {

    /* Uncomment the following lines to print uncaught exceptions to stdout */
    // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
    // exception.printStackTrace(System.out);
}
/**
 * Handle the Applet init method.
 */

public void init() {
    super.init();
    try {
        setName("AppletClienteDos");
        setLayout(new java.awt.BorderLayout());
        setSize(517, 328);
        add(getPanel1(), "Center");
        initConnections();
        // user code begin {1}
        // user code end
    } catch (java.lang.Throwable ivjExc) {
        // user code begin {2}
```

```
            // user code end
            handleException(ivjExc);
        }
    }
    /**
     * Initializes connections
     */

    private void initConnections() {
        // user code begin {1}
        // user code end
        getButton1().addActionListener(this);
    }
    /**
     * main entrypoint - starts the part when it is run as an application
     * @param args java.lang.String[]
     */
    public static void main(java.lang.String[] args) {
        try {
            Frame frame;
            try {
                Class aFrameClass = Class.forName("com.ibm.uvm.abt.edit.TestFrame");
                frame = (Frame)aFrameClass.newInstance();
            } catch (java.lang.Throwable ivjExc) {
                frame = new Frame();
            }
            AppletClienteDos aAppletClienteDos;
            Class iiCls = Class.forName("AppletClienteDos");
            ClassLoader iiClsLoader = iiCls.getClassLoader();
            aAppletClienteDos = (AppletClienteDos)java.beans.Beans.instantiate(iiClsLoader,"AppletClienteDos");
            frame.add("Center", aAppletClienteDos);
            frame.setSize(aAppletClienteDos.getSize());
            frame.setVisible(true);
        } catch (Throwable exception) {
            System.err.println("Exception ocurrio en main() of java.applet.Applet");
            exception.printStackTrace(System.out);
        }
    }
}
```