DISSERTATION

Securing Web Services in a User-to-Application Model Based on Certificate Private Extensions and Smartcard Technology

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

O. Univ.-Prof. Dr. Dietmar Dietrich
Institut für Computertechnik 384

eingereicht an der Technischen Universität Wien,
Fakultät für Elektrotechnik und Informationstechnik

von

Dipl. Ing. Blerim Rexha
Matr.-Nr. 9527190
Favoritenstrasse 33/2/16, A-1040 Wien

Wien, im März 2004 .... [illegible]

ABSTRACT

Web Services represents a new way of invoking remote functions over standard Internet protocols. They are basic building blocks of the distributed computing over the Internet. Security and privacy are central issues for the acceptance of Web Services in particular and the growth of the Internet market in general. Public Key Infrastructure and X.509 Certificates have been established as the most trustworthy methods for assuring online security. In this thesis are compared the existing approaches for securing Web Services and proposed new approaches for increasing security by avoiding privacy violation using X.509 certificate private extensions and storing these certificates in smartcards. Adopting the Internet for every possible transaction has lead to a situation where a user has to enter extra information for completing his real profile. The aim of the thesis is to increase user privacy in online transactions. This is achieved through extending certificates with private extensions. Each extension holds encrypted data for user properties, such as: credit card number, insurance number, address, etc., and thus each online participant understands the general (public) data on the certificate and one relevant encrypted private extension.

KURZFASSUNG

Web-Dienste stellen ein neues Verfahren dar, um entfernte Funktionen über Standard Internet Protokolle auszuführen. Sie sind die fundamentalen Bausteine für verteilte Applikationen über das Internet. Sicherheit und Privatsphäre haben eine bedeutende Rolle, da von ihnen die Akzeptanz der Web-Dienste, insbesondere das Wachstum des Internet-Marktes, abhängt. Public Key Infrastructure und X.509-Zertifikate sind seit Jahren bewährte Methoden um die Sicherheit der Online Transaktionen zu erhöhen. In dieser Arbeit werden existierende Ansätze über die Sicherheit von Web-Dienste verglichen. Ausserdem werden neue Ansatze vorgestellt, wie die Sicherheit erhöht werden kann, indem die Verletzung der Privatsphäre durch Verwendung der privaten Erweiterungen in X.509 Zertifikaten und speichern dieser Zertifikate in Smartcards vermieden wird. Die Verwendung des Internets für jegliche Transaktionen hat dazu geführt, dass der Benutzer noch zusätzliche Informationen eingeben muss, um sein Benutzerprofil zu vervollständigen. Ziel dieser Arbeit ist es, die Privatsphäre der Benutzer in Online-Transaktionen zu erhöhen. Das wird erreicht durch Hinzufügen von privaten Erweiterungen in das Zertifikat. Jede Erweiterung besteht aus verschlüsselten Benutzereigenschaften wie z.B.: Kreditkartennummer, Versicherungsnummer, Anschrift, usw. Somit versteht jeder Online-Teilnehmer die allgemeinen Daten im Zertifikat und den für ihn dazugehörigen verschlüsselten Anteil.

ABSTRAKT (in Albanian)

Web shërbimet (Web Services) paraqesin një mënyrë të re për qasjen e funkcioneve në largësi përmes Internetit. Ato janë blloqe bazike të aplikacioneve të shpërndara nëpër internet. Siguria dhe intimiteti (privacy) i shënimeve kanë rëndësi primare prej të cilave mvaret pranueshmëria e web shërbimeve në veçanti dhe rritja e Internet tregut në përgjithësi. Infrastruktuta e çelësave publik (Public Key Infrastructure) dhe X.509 çertifikatat (X.509 certificates) janë vertetuar si metoda të sigurta në Internet për realizimin e sigurisë së shënimve digjitale. Në këtë punim krahason qasjet egzistuese për sigurinë e web shërbimeve dhe propozohen qasje të reja se si të rritet siguria, duke e mos lënduar intimitetin e shfrytëzuesit, me përdorimin e zgjerimeve private në X.509 çertifikata dhe duke i ruajtuar këto çertifikata në smartcardsa (Smartcards). Përdorimi i Interentit si medium për çdo llojë transakcioni ka çuar deri në situatën kur shfrytëzuesi duhet dhënë informata shtesë për ta kompletuar profilin e vetë të vertetë. Qëllimi i këtijë punimi është që të rritë intimitetin e shfrytëzuesit në online transakcione. Kjo është arritur duke i zgjeruar çertifikatat me zgjerime private. Secili zgjerim përmban shënime të enkryptuara të vetive të shfrytëzuesit siç janë: numri i kredit kartelës, numri i sigurimit, adresa etj., dhe kështu që secili online pjesëmarrës i kupton shënimet e përgjithshme (publike) në çertifikatë si dhe një zgjerim të enkryptuar i cili është i rëndsishëm për të.

ACKNOWLEDGMENTS

First I would like to thank my parents for their continuous support and encouragement to complete this work. I received infinite love from them. My wife, Shahadije, I thank not only for supporting me in my early studying years but also for sharing a love with me for our two children, Gent and Gresa. Without her love, encouragement and support this thesis would never be finished. For support during my studies I want to thank also my brothers, Burim and Bujar, and my sister, Afërdita.

Special thanks go to my supervisor Prof. Dietmar Dietrich. Discussions with him and his useful suggestions and recommendations helped me successfully to complete this work and to join a new field. His valuable comments and criticism improved my scientific background.

My thanks also go to my Siemens colleagues, Heinrich Giefing, Franz Schiller, Hermine Buder and Johann Honigschnabl for contributing to a friendly working atmosphere. Herbert Jank, my department manager, I thank also for supporting me writing this thesis.

Special thanks go also to my friend Astrit Ademaj for his useful suggestions and proof readings of this thesis. Thanks to Yll Haxhimusa and Nysret Muslija for proof readings parts of this thesis.

PREFACE

In the last years Web Services (WS) technology has gained great attention from the information technology (IT) community. Web Services were introduced in June 2000 as a key component of the Microsoft's .NET technology. Web Services opened a new era of distributed computing. Now, nearly every major software vendor including Microsoft, Sun Microsystems, IBM, Oracle and many others is marketing Web Services tools and applications.

Web Services include a set of Internet related standards that can enable any two computer applications to communicate and exchange data over a network. The main standard used in Web Services is the eXtensible Markup Language (XML), developed by the World Wide Web Consortium (W3C). XML is designed to describe data. An XML document is text-based and consists of user created tags. XML provides the basis for higher Web Service protocols, including the Simple Object Access Protocol (SOAP), the Web Services Description Language (WSDL) and the Universal Description, Discovery and Integration (UDDI). SOAP is an XML-based protocol that enables applications to exchange information over Hyper Text Transfer Protocol (HTTP). WSDL is an XML-based language (document) to define and locate Web Services and describe how to access them. UDDI is a framework that defines XML-based registries in which businesses can publish information about themselves and the services they offer. UDDI is similar to a telephone book.

The main criticism that Web Services received from their first release was the security issue. Using HTTP as the transport layer, Web Services require minimal network maintenance, since by default the HTTP port, 80, is always open. This feature, using the existing HTTP port, is seen at the same time as a great advantage and a great disadvantage of Web Services.

Network security in general can be implemented in three layers: application, transport, and network. Building security at the application layer is most flexible, because the scope and strength of the protection can be adapted to meet the specific needs of the application. Pretty Good Privacy (PGP) is the best representative in this group. The approach presented in this thesis is based on application layer security. Implementing security in the transport layer means having a secure channel to communicate with. Typical representatives are Secure Socket Layer (SSL) and Secure Shell (SSH). Implementing security in the network layer, known also as the Internet Protocol (IP) layer, enables all services and applications above the IP layer to communicate securely with each other.

Public Key Infrastructure (PKI) consists of software, encryption technologies, protocols and services that enable entities to protect their communications and business transactions on an insecure network such as the Internet. Typical PKI includes services for issuing, publishing and revoking public keys through use of Certification Authorities (CA). Binding the entity name with its corresponding public key is done with a digital certificate. To achieve maximum interoperability during the exchange of certificates in different systems, the International Telecommunication Union (ITU) defined in 1988 a standard about X.509 digital certificates as part of their X.500 directory recommendations. In this thesis X.509 certificate reserved fields (so-called private extensions) are extended to carry extra information about its owner in encrypted form. This extra information, such are credit card number, address, insurance number etc. could be used in online transactions to complete the real user profile automatically.

Smartcards have been proved to be tamper-resistance, secure and reliable devices. The access to a smartcard is protected with a personal identification number (PIN) known only to the smartcard holder. Smartcards are used as secure storage for the private key and its corresponding X.509 certificate and as a secure processing device, since encryption with a private key (digital signature) is computed internally in the smartcard.

This thesis is organized as follows:

Chapter 2, Technology Landscape, introduces the terminology and describes the state of the art of the XML Web Services, digital certificates, PKI, smartcards, online payment systems and Web Services Enhancements (WSE).

Chapter 3, The Web Services Market, analyzes market forecasts for Web Services and the new challenges that Web Services bring to enterprises.

Chapter 4, Privacy Violation, describes the problem in detail. It compares the physical credit card payment process with current online credit card payment processes and analyzes the privacy violation points.

Chapter 5, Network Security for Web Services, discusses network configuration and different existing network security technologies which can be used in conjunction with Web Services. In particular, security technologies based on the network layer and Virtual Private Networks are discussed there.

Chapter 6, Identity Certificates, presents the first new contribution of this thesis. Here a new X.509 certificate structure is presented to carry extra user information like credit card details, address, insurance number etc. In this chapter the possibility of storing X.509 certificate in smartcards is discussed.

Chapter 7, Attribute Certificates, presents the second main contribution of this thesis. Here, as in Chapter 6, the new structure of attribute certificates is presented. This approach is divided into two categories: attribute certificates that are valid for a long time (at least as long as the identity certificate) and single-use attribute certificates.

Chapter 8, Non-repudiation, presents different flow protocols between user, merchant and bank to accomplish an online payment. At the end of the chapter pros and cons of each approach are discussed and comparisons are made against well known existing solutions.

Chapter 9, Case Study – Virtual Web Drive, presents detailed information about a test application that uses the new structure of X.509 certificates. In this chapter are analyzed the technology used, results, and problems faced during implementation.

Chapter 10, Summary, presents the achievements of the work presented in this thesis and a discussion of possible future work.

In Appendix A, are presented screen shots of the graphical user interface of a test application, which is realized as the practical part of this thesis.

Contents

Chapter 1

Introduction and Motivation

Web Services are part of a natural evolution of the Web into an open medium. They are basic building blocks toward the distributed computing over the Internet [Wol01]. Web Services are configured to be firewall friendly and in this way they have the capacity to bypass corporate security polices and expose valuable corporate data, applications and systems to a variety of external threats [CNW01]. Current e-business trends require integrating different distributed systems, interfacing easily with existing systems and not introducing security risks. Two thirds of the 100 US and European CIOs [1] surveyed by Merill Lynch [2] stated that they were investing in Web Services [Wes02].

1.1 Introduction

Many people and companies have discussed about the exact definition of Web Services [JZ02]. In this thesis, a Web Service is "any piece of software that makes it self available over Internet and uses a standardized eXtended Markup Language (XML) messaging system“ [Cer02]. The functionality offered by Web Services is accessible through standard internet protocols; such as the Hyper Text Transfer Protocol (HTTP).

The communication with Web Service is XML encoded. A client invokes a Web Service by sending an XML message and waits for corresponding XML response as Figure 1.1 represents.

Since XML is platform independent Web Services are not tied to any particular operating system or programming language. Windows applications can communicate with UNIX applications and vice versa: Web Services are platform independent [BCG$^+$01]. The rapid growth of XML traffic on the network and the widespread adoption of Web Services are two forces that are transforming IT today [BS02b]. The remote technologies like Common Object Request Broker Architecture (CORBA), Common Object Model (COM), and Remote Method Invocation (RMI) promised to invoke a method over network, but they were constrained to computers in a specified domain and to particular operating systems [Was02]; in any case cross domain

[1] Chief Information Officer (CIO) - A person, in middle and big companies, who is responsible for design and implementation of Information Technology (IT) programs and initiatives.

[2] http://www.ml.com

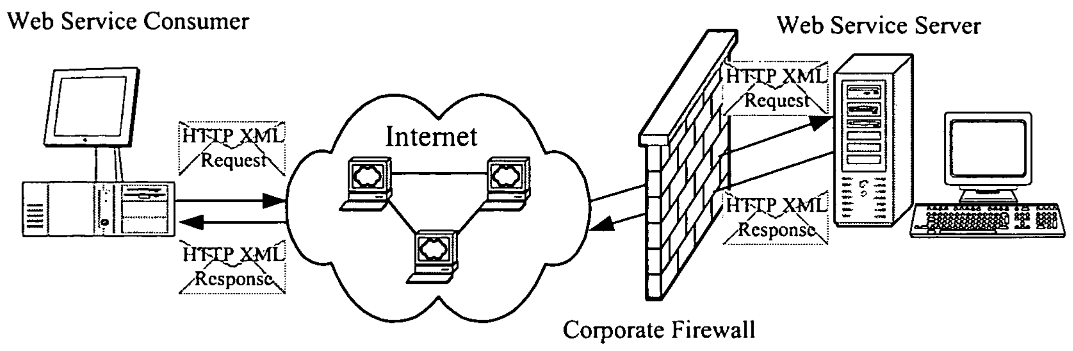


Figure 1.1: Web Service request

invocation calls were impossible [BCG+01]. Most of the trouble come from network architecture: firewalls and proxy servers often block most ports [BS02b].

XML communication between Web Service participants uses HTTP, an inexpensive and easy to use network transport protocol [Fer02] available on every modern operating system platform. Using HTTP has another advantage regarding a network configuration. Most networks firewalls have the HTTP port open, which means Web Services need minimal network administration. In this way Web Services can bypass the standard firewalls and expose valuable corporate data in Internet. The XML firewall (content aware firewall) need is born with Web Services [BS02b].

The scale on which Web Services will be accepted by the IT market depends critically on how secure they are and how the security is implemented [Ras02]. Everyone who exposes services to the community wants in some way to know who accesses his services and in what way he can allow (enforce) the clients (Web Service consumers) to pay for used services.

1.2 Motivation

Exchanging information with Web Services in XML format means sending and receiving data in plain text. After its first public release, Web Services have received critical attention with respect to their lack of security [JZ02, Gai02, Fer02]. In the IT community it was claimed that, rather, security is a transport issue [MSD01c, Vas01].

What makes the security of Web Services so challenging is the distributed nature and platform independent of these services. The technology for Web Services is based on the operation of many different software applications running on distributed and different operating systems connected via Internet [Pro02].

A X.509 certificate is data structure, which format is a joint standard of International Organization for Standardization [3] (ISO) and the International Telecommunication Union – Telecommunication [4] department (ITU-T). X.509 certificate binds the users name and his corresponding public key. X.509 certificate is digitally signed by certification authority (CA). Any changes made at the X.509 certificate after issuing

[3]http://www.iso.org
[4]http://www.itu.int

are easily detected by Public Key Infrastructure (PKI) aware applications. X.509 certificates are usually used for authenticating users on the Internet by making use of PKI.

Smartcards have been established as secure hardware devices in the market. They have proved very secure against different threat models [SS99], as a safe place to store valuable information such as private keys, account numbers, digital certificates etc. Smartcards can be used also as secure processing devices to perform off-line operations such as private key encryption and decryption. With the introduction of personal computer smartcard (PC/SC) architecture [Wor97] smartcards and smartcard readers are becoming standard devices for personal computers.

Modern people have little privacy. "(Information) Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how and to what extent information about them is communicated to others [5]" [Bra00]. Carrying a mobile phone their location and moving route is always known [Pei01, HW00]. Using the credit cards for shopping their shopping habits are also known. Using static Internet Protocol (IP) addresses, cookies, or simply sniffing into the communication channel their interesting topics can also be discovered [Iac02, Wha02].

The work presented in this thesis is a contribution toward increasing the privacy of modern people using X.509 certificates.

1.3 Problem definition and objectives

Web Service providers do not want to give their services for free. In most cases they want to know as much as possible about the users who access their services, but in any case they want to charge consumers of their services. The technology to provide high level of security for Internet payments already exists [Has01].

Confidentiality, authentication, integrity, and non-repudiation are the basic components for secure online transactions. These components require the implementation of PKI technology. PKI uses digital certificates to address these requirements [Hun01]. Current digital certificates do not have enough information for a one-to-one mapping to a real user profile. Digital certificates, introduced about 30 years ago, binds the public key with the name of the public key holder [Gut00]. They do not carry any information about credit cards or shipping addresses or other properties like: bank account numbers, insurance number etc.

The most frequent form of payment implemented today is to send the user's credit card number over a Secure Socket Layer (SSL) or Transport Level Security (TLS) enabled web browser to a merchant server. There are two reasons for this widespread usage [AJSW97]:

- from a merchant point of view it is very easy to receive and process these payments, and
- all known "secure payment systems" are classified as too complex to implement.

[5]One of the most cited definitions of privacy is from Columbia University professor Alan Westin in his work *Privacy and Freedom*, 1967. It has been used as basis for the U.S. Privacy Act of 1974 [Bra00].

Analyzing a typical, simple online payment scenario, as represented in Figure 1.2, *one can ask why should the Web Service provider (merchant) know the user's credit card number, and why should the bank know about the goods or services the user has bought?*

In a real case Internet payment transaction there will be a lot of messages travelling from three parties involved, but the essence a request travels from the consumer to the Web Service provider (merchant) and to the bank [Kin99]. The request contains consumer information, information about goods or services bought and details, like credit card number for the bank to pay the merchant, see Figure 1.2. Ideally, all parties involved in a payment transaction should be authenticated against each other, and a secure communication path should span form the consumer to the bank [Dja02]. But SSL/TLS cannot secure the whole path. SSL/TLS can secure only the path between any two end points, and it can not provide non-repudiation of the origin [FKK96]. Ideally too, the merchant should not know the credit card number nor the bank about the goods, as presented by gray boxes in Figure 1.2. But if the merchant does not know the consumer's credit card number, the question arises of how merchant gets the money for services offered.

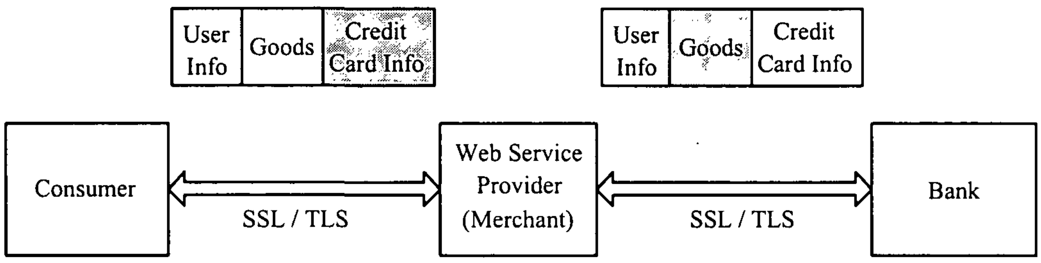


Figure 1.2: Simple Internet payment scenario

The Secure Electronic Transaction (SET) protocol fulfills the latest requirements, but beyond that in practice SET has been proved as complicated, slow in performance and unacceptable from the user experience [TY98].

The objective of this thesis is to advance the usage of X.509 certificates and to increase user privacy in online transactions. Private user information such as: credit card numbers, bank accounts, address etc., should be protected against unauthorized use.

Online payment systems are a typical example of where user privacy is violated. Therefore, *in this thesis digital certificates are extended to carry additional encrypted data about the private user information.* The user profile stored in a X.509 certificate will be extended with credit card numbers, bank accounts, address etc. In this way the new user profile will be more compact and better match a real user profile.

Since the X.509 certificate is exchanged with almost everyone or, as is usual, stored in a public directory, the question arises of how the merchant and bank know that the request (order) is being made by the legitimate credit card holder and not by someone else. *A simple transaction protocol based on random numbers, transaction ID and digital signatures is presented in this thesis to achieve non-repudiation and to avoid reply attacks in online transactions.*

With the new Web Service Enhancements (WSE) it is possible to encrypt and sign different parts of message with different keys and to force message route path [IM02b, Evj03], so the problem depicted in Figure 1.2 is solved with WSE. *However, this thesis presents a novel approach based only on X.509 v3 certificate private extensions.*

The X.509 certificate extensions are supplementary data structures defined with the latest version (v3) of the X.509 format [ITU00]. These fields, known also as private or proprietary extension, may contain any data regarding the certificate holder, a certificate enabled application or issuing authority.

The approach presented in this thesis is not limited for use only with Web Services. It can be used with any application, that understands X.509 v3 certificates. Web Services present sophisticated certificate transport mechanisms: from user to merchant to bank. And due to their firewall friendly nature it is possible to exchange transaction data without changing existing network configurations [BS02b].

It is not an objective of the work presented in this thesis to make anonymous transactions using X.509 certificates. Furthermore we are aware of the global unique properties of the X.509 certificates and their traceability as criticized by [Bra00, AE00], when used in an not encrypted communication channel.

Chapter 2

Technology Landscape

Chapter 2 presents the state of the art in the Web Services, security related technologies, smartcards, and online transactions. This chapter aims to serve as a basis not only for coming chapters but also for comparing and discussing the pros and cons of existing security technologies and online transactions.

2.1 Web Service definition

A Web Service uses the Internet to link applications, systems and resources inside and among enterprises, to enable a new approach to business processes and relationships with partners, customers and suppliers around the world [CNW01]. By using basic technologies such as XML and HTTP Web Services today are the state of the art for information exchange between applications and for interaction between business transaction services on the network [Sch02].

The concept of Web Services has received the greatest attention in the IT community since the .com boom [BS02a]. Many people and companies have debated about the exact definition of Web Services [JZ02]. A few definitions of Web Services are presented below (in alphabetical order):

IBM – "Web Services are self-describing, self-contained, modular applications that can be mixed and matched with other Web services to create innovative products, processes, and value chains. Web services are Internet applications that fulfill a specific task or a set of tasks that work with many other web services in a manner to carry out their part of a complex workflow or a business transaction. In essence, they enable just-in-time application integration and these web applications can be dynamically changed by the creation of new web services" [JZ02].

Microsoft – "A Web Service is a unit of application logic providing data and services to other applications. Applications access Web Services via ubiquitous Web protocols and data formats such as HTTP, XML, and SOAP, with no need to worry about how each Web Service is implemented. Web Services combine the best aspects of component-based development and the Web, and are a cornerstone of the Microsoft .NET programming model" [MSD03b].

Sun – "A Web service is, simply put, application functionality made available on the World Wide Web. A Web service consists of a network-accessible service, plus a formal description of how to connect to and use the service. The language for formal description of a Web service is an application of XML. A Web service description defines a contract for how another system can access the service for data, or in order to get something done. Development tools, or even autonomous software agents, can automatically discover and bind existing Web services into new applications, based on the description of the service" [IWV03].

W3C – "The same way programmatic interfaces have been available since the early days of the World Wide Web via HTML forms, programs are now accessible by exchanging XML data through an interface, e.g. by using SOAP Version 1.2, the XML-based protocol produced by the XML Protocol Working Group. The services provided by those programs are called Web services" [W3C02].

For IBM the important feature of Web Services is their "self-describing, self-contained and modular" nature, whereas for Microsoft and Sun accessing Web Services "via ubiquitous Web protocols" and the "XML data format" are more important features. In any case each software manufacturer is keen to make Web Services as compatible as possible to their products.

A short tight definition of Web Services then would be "encapsulated, loosely coupled, contracted software objects offered via standard protocols" [Mye02]. Web Services can be thought of as a universal client-server architecture that allows different and distributed systems to communicate with each other without using proprietary client libraries [Mye02].

It is worth pointing out what Web Services are not, many people look to new technologies to solve their problems. Web Services do not provide a solution to all Information Technology (IT) problems and they are not a strategy in themselves [FW02]. When implementing new technologies, companies will initially have to face costs in time and money. Each company would have different Return On Investment [1](ROI) mainly based on the technology used for implementing business processes [SS02].

2.1.1 XML – The Internet language

The EXtensible Markup Language (XML) is markup language designed to describe, exchange and store data. The greatest feature of XML is that anyone can define their own tags, and therefore XML will become the the most common format for data manipulation and data transmission [Sch03c]. XML is a project of the World Wide Web Consortium [2] (W3C), and the development of the specification is being supervised by their XML Working Group. XML is a public format i.e. it is not protected by the patent or trade mark of any company. The v1.0 specification was accepted by the W3C as a Recommendation on Feb 10, 1998 [Fly03].

In the real world, computer systems and databases contain data in different and incompatible formats. Exchanging data between different systems over the Internet represents a time consuming task for IT developers [Sch03c]. Converting the

[1]ROI represents a ratio of net benefits over costs expressed as a percentage [SS02].
[2]http://www.w3.org

exchanged data to a format, such as XML, that could be read by any system eliminates almost the data exchange problem. The new office suites, Star Office by Sun [Sun03] and Microsoft Office 11 [Mic03b], are now capable of saving content as XML.

XML is establishing inside enterprises as a basic tool for addressing a many yet unsolved or very hard solved problems like [BS02b]:

- document creation and management,
- web content, and
- Business to Business (B2B) communication.

Web Services in particular are quickly adopting XML. As a result, XML traffic on corporate networks in next few years is to expect to grow rapidly, as is presented in Table 2.1.

Protocol/Year	2002	2003	2004	2005	2006
FTP	0.06%	0.06%	0.04%	0.03%	0.02%
HTTP	11.33%	17.06%	24.17%	31.42%	36.57%
DNS	0.08%	0.07%	0.06%	0.04%	0.03%
Mail	3.16%	3.81%	4.32%	4.49%	5.18%
XML	**1.60%**	**3.34%**	**7.09%**	**13.84%**	**24.15%**
Other	83.73%	75.64%	64.30%	50.17%	35.04%

Table 2.1: Percentage utilization of network traffic by format [BS02b]

Table 2.1 shows a forecast from [BS02b] of the increase in network traffic in the XML format in the next years. The projection in Table 2.1 is important for network administrators and firewall developers. In the near future it will be not enough to have only a static firewall, i.e. a firewall that does not examine the content of messages and just blocks some known ports in TCP/IP layer (see Figure 2.1).

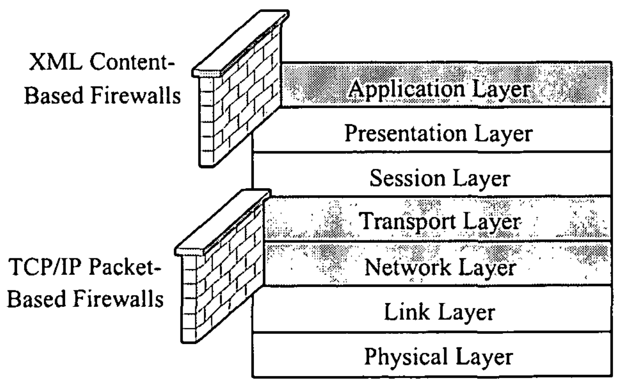


Figure 2.1: XML vs. TCP/IP firewall

Future firewalls must be content, i.e. XML content, aware [ONe03], as is presented in Figure 2.1. Figure 2.1 shows that XML firewalls must be build in the application layer.

One of the greatest features of XML is that using this format one could send to a server Hyper Text Transport Protocol (HTTP) POST and GET requests and receives responses in XML format. XML Remote Procedure Calls (RPC) work in this way, exchanging POST and GET method calls. XML-RPC software running on different operating systems and environments can make procedure calls over the Internet [Use03].

XML-RPC uses the HTTP as its transport mechanism and XML for encoding function parameters. Even it is designed to be simple, and it allows complex data structures to be transmitted, processed and returned. XML-RPC is described in detail in [Win99] and an example of the invoking method over XML-RPC is presented in Figure 2.2.

```
<methodCall>
  <methodName>sample.sumAndDifference</methodName>
  <params>
      <param><value><int>5</int></value></param>
      <param><value><int>7</int></value></param>
  <\params>
<\methodCall>
```

Figure 2.2: XML-RPC method call [Kid01]

XML-RPC supports the data types of: int, string, boolean, double, dateTime, Base64 type [3] , array and structures [Kid01].

2.1.2 Simple Object Access Protocol

The Simple Object Access Protocol (SOAP) is a platform independent and language independent XML-based protocol that enables applications to exchange information over HTTP [Sch03a]. The SOAP standard, as defined by W3C, explicitly states that HTTP is not the only transport protocol that can be used to send SOAP messages [W3C00], but HTTP is the default one. Using the stateless HTTP protocol provides powerful scalability as well as the ability to pass through existing firewall software unimpeded [BCG$^+$01]. In this way, with no modification to existing network configuration, message requests and responses can be routed over the Internet. Sometimes this feature is seen as a threat to existing network configurations [San03, Mid03].

SOAP 1.1 was suggested (by Compaq, HP, IBM, Microsoft, SAP and some other companies in May 2000) to the W3C as a protocol for exchanging information in a

[3]Base64 is a primitive encryption method used to embed non-printable characters in XML documents. Each sequence of three bytes (24 bits) is divided into four chunks of six bits each. Each of these chunks has one of the 64 values, each of which is mapped to a printable character in the range {A-Z, a-z, 0-9, +, /}. These characters are then converted back to a sequence of four bytes. Thus Base64 conversion has a 33% data increase as a side effect [MJ03].

distributed environment. The XML Protocol Working Group at the W3C is currently working on SOAP 1.2 [W3C03].

SOAP defines how to use XML and HTTP to access objects and methods in a platform independent way. The SOAP protocol can be thought of as the glue between different and distributed software components. If developers can agree on HTTP and XML, SOAP offers a mechanism to bridge competing technologies in a standard way [Sko00].

A SOAP message is an ordinary XML document; its structure is presented in Figure 2.3.

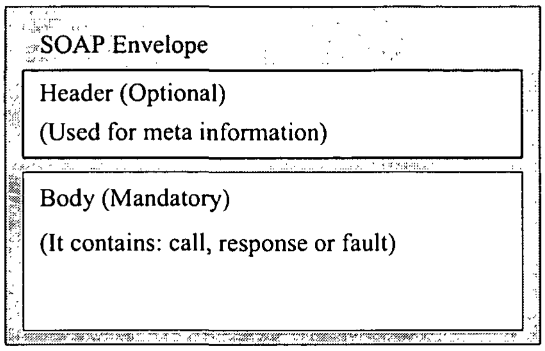

SOAP Envelope
Header (Optional)
(Used for meta information)
Body (Mandatory)
(It contains: call, response or fault)

Figure 2.3: SOAP message structure

Envelope – all SOAP messages are packed within an Envelope element (see Figure 2.3). This is the root element of the entire package. Inside the Envelope element there are always a single body element and optionally a header.

Header – is an optional element and is used for meta data, like transaction information, callers identity (username, password, X.509 certificate etc.) [Evj03]. The SOAP specification allows for a *mustUnderstand* attribute to be included in any element in a header. The *mustUnderstand* attribute forces the Web Service either to process (understand) the element with this attribute, or to respond with a fault message [W3C00].

Body – this is a mandatary element and its contents depends on the SOAP context. For a method call, it contains the serialized method invocation request with the name of the method and each parameter serialized in XML format [Box00]. The response is also encoded in XML format. In the case of an error a fault message is returned, the body containing the error information [W3C00].

The SOAP specification enables many different data types and data structures to be passed in SOAP messages. All these data types are serialized, therefore serialization is very important for SOAP. All systems must agree on a standard way to represent data types, so the SOAP specification [W3C00] explicitly states how each data type is serialized into a SOAP message. Some of the most used data types are: int, boolean, float, string, dateTime, binary, etc [W3C01].

2.1.3 SOAP vs. DCOM, CORBA and RMI

It is expected, that in the near future all RPC calls over networks will be done in SOAP [Cha02]. SOAP seems to fulfill what today's RPC technologies seek but have never achieved: remote calls over the Internet.

The properties of different RPC technologies are presented below.

Distributed COM (DCOM) – is an evolution of COM technology for distributed applications. COM is an object oriented and platform independent technology for creating software components that can interact [Swa00]. COM is the basic technology for Microsoft's object linking and embedding (OLE) and ActiveX technologies, as well as others [MSD03b].

COM is an object specification that defines interfaces for objects. Different objects can communicate to each other using these interfaces. COM is a language independent specification, thus it does not matter in which language the objects themselves are coded in as long as they implement the COM interfaces [Gri97].

Theoretically COM can be implemented on any operating system, though in practice its implementation in systems other than Microsoft Windows has been negligible [Sty02]. To enable communication between COM objects on different systems, the COM specification has been extended and called Distributed COM (DCOM).

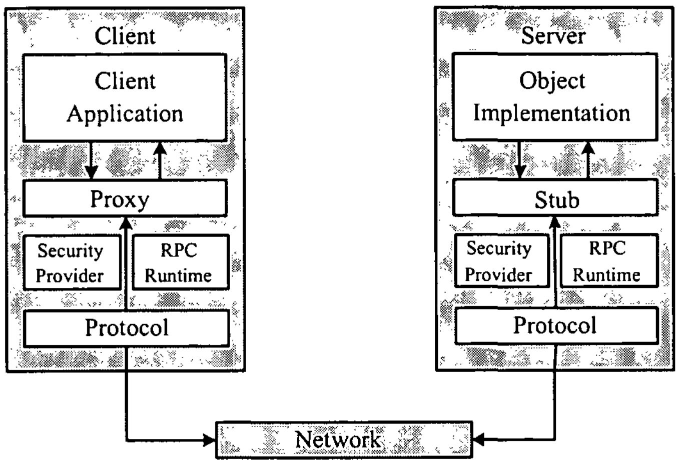


Figure 2.4: DCOM architecture [Gri97]

The DCOM architecture is presented in Figure 2.4. DCOM is connection oriented, i.e. there are many packets (*pinging*) exchanged to set up/maintain sessions [Box00]. This is necessary because a DCOM server must know when the client is no longer interesting in server objects or the client has forgotten (for any reason or has crashed) to release the freed memory by server objects. So with the *ping* mechanisms the server checks if its clients are still alive.

DCOM is usually described as COM with a longer wire. From the client and server architecture point of view, code for DCOM is transparent whether the object is on the same machine as the client or across a network. DCOM method calls are made over RPC, and it uses the RPC runtime protocol [Gri97].

CORBA – defines an environment where clients can make requests upon an object and the object can send responses back to the client. The client and object in a distributed systems are usually not in the same application memory address space so the connection is made by an object request broker (ORB) [Gri97], as presented in Figure 2.5.

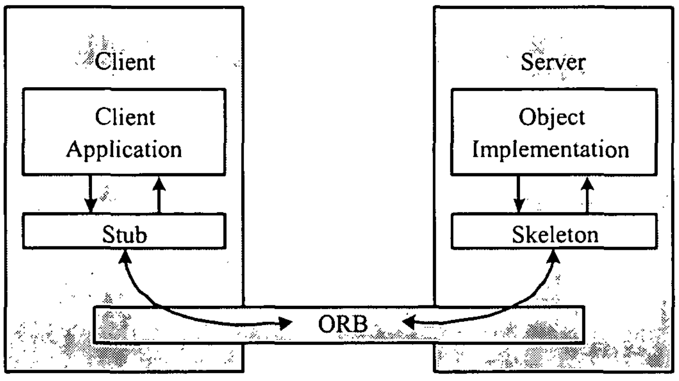


Figure 2.5: CORBA architecture [OMG03]

CORBA is a specification by the Object Management Group, which is a group of middleware vendors. CORBA is language independent and is more widely implemented across different platforms than COM [Sty02]. CORBA uses Internet Inter-ORB Protocol (IIOP) to communicate between different systems. The developer defines the object's interface using CORBA's interface description language (IDL) and compiles this with the IDL compiler. The result of compilation is **stub** code, which resides on the client machine, and **skeleton** code, which exists on the server machine [OMG03], as is depicted in Figure 2.5.

EJB / RMI / IIOP – are specifications by Sun Microsystems for the Java[4] Platform. Java's greatest feature is that when compiled it produces bytecode that can be interpreted by any computer that has a Java Virtual Machine (JVM) [Fla02].

Enterprise Java Beans (EJB) is platform independent technology, but is not language independent. All EJB objects are written in the Java language [Sty02].

To communicate between different systems, EJB uses a variation of IIOP called Remote Method Invocation (RMI) over IIOP [Gro01]. RMI is a mechanism to access the methods of a remote Java class. RMI is a protocol in Java language for communicating between different systems [Har00].

There are some problems with the DCOM, IIOP and RMI/IIOP protocols. The main problem is their incompatibility [Col03b]. A DCOM based system cannot communicate with an EJB based system and vice-versa. It is obvious that if an

[4]Java is a modern object oriented and platform independent programming language, see http://www.java.sun.com

company has different applications on different platforms, these applications cannot be integrated using the DCOM, IIOP and RMI/IIOP protocols. Another problem is that these protocols are not firewall friendly. Usually firewalls are configured to allow access only through few ports, the most popular being the HTTP port 80 [Sty02]. DCOM, IIOP and RMI/IIOP protocols use different ports, which are usually blocked by most corporate firewalls. DCOM for example requires User Datagram Protocol (UDP) and TCP ports 135-139 [Was02]. Thus, such distributed applications residing behind corporate firewalls cannot communicate with each other, even when on the same platform.

SOAP is the answer to these problems. SOAP unifies the HTTP and XML technologies. SOAP uses HTTP to transmit and XML to encode data among different applications. Since XML is a universal standard, all platforms can access and process the information coded in XML [Was02]. From the point of view of functionality, SOAP is closer to RPC/IIOP than DCOM/RMI [Box00].

A comparison of the different RPC technologies is given in Table 2.2, where can it be seen that method invocation across domains is only possible with SOAP protocol.

	DCOM	IIOP	RMI/IIOP	SOAP
Format	Binary	Binary	Binary	XML
Platform	Windows	Unix	Independent	Independent
Firewall friendly	No	No	No	Yes
Programming language	Independent	Independent	Java	Independent
Call across domains	No	No	No	Yes

Table 2.2: SOAP versus DCOM, IIOP and RMI [Sty02]

SOAP is already establishing as the application protocolof the future [Cha02], as depicted in Figure 2.6. SOAP is platform and programming language independent.

Response times between the SOAP and CORBA function calls are compared in [EPL02]. Standard SOAP calls are up to 400 times slower than CORBA calls. But some performance improvement techniques have been suggested in [EPL02] that lead to a decrease in of SOAP response times i.e. being only 7 times slower than CORBA function calls.

Microsoft, SUN, IBM and other operating system vendors are now supporting SOAP based environments and developer tools.

2.1.4 WSDL and UDDI

The first question after creating a Web service is: how do clients find the desired Web service on the Internet? After they have found it, how do they get an interface description of the supported methods? Web Services Description Language (WSDL) is an XML-based language document to define and locate Web Services. WSDL also describes how clients can access (invoke) methods that Web Services offer [BCG+01]. In the COM world, WSDL can be seen as synonymous to a type library [Was02]. WSDL is considered to be the "contract" between a Web Service and a client.

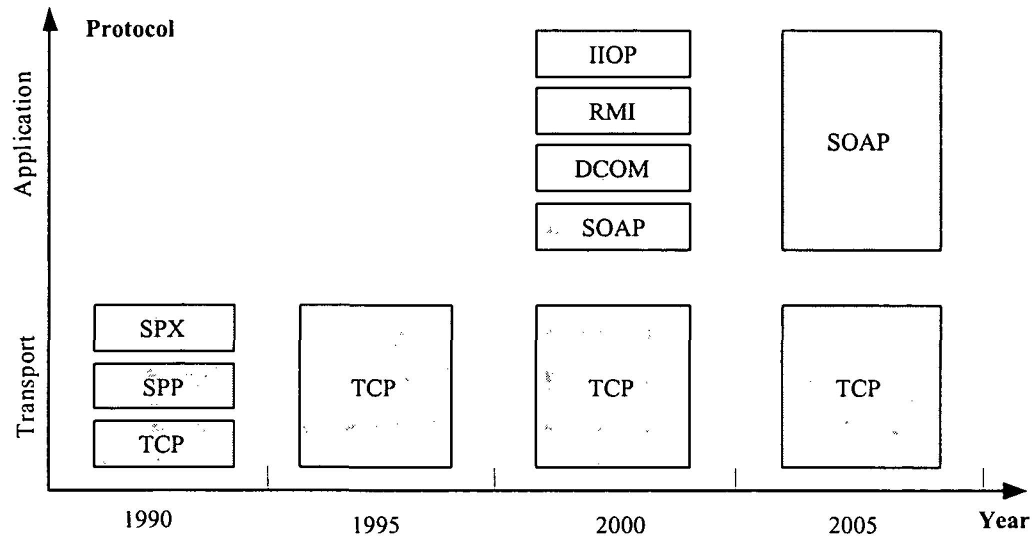


Figure 2.6: Analogy between TCP and SOAP [Cha02]

WSDL is a suggestion by Ariba [5], IBM and Microsoft for describing services for the W3C XML activity on XML protocols [HNN02]. Based on the work at W3C WSDL might become a W3C Working Draft before the end of 2002, and an official Recommendation before the end of year 2003 [W3C02].

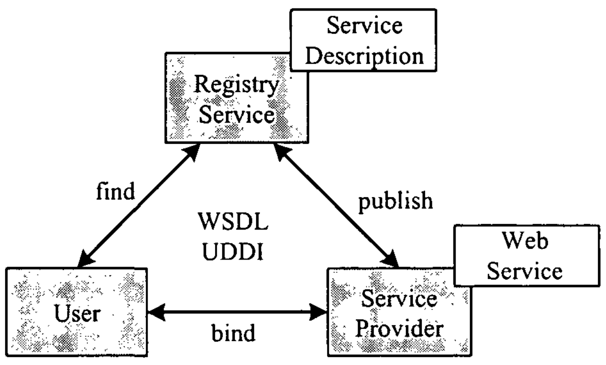


Figure 2.7: Principle interaction model of Web Services

Universal Description, Discovery and Integration (UDDI) is an open directory service where enterprises can register and search for Web Services [Sch03b]. UDDI uses WSDL to describe interfaces to Web Services [Sid02b]. The UDDI registry is planned to be like the yellow pages on the Internet. UDDI is a global business registry with its root directory under www.uddi.org [Sch02]. An example of a typical interaction of Web Services is presented in Figure 2.7.

Before the UDDI service, there was no industry accepted approach for businesses to reach their customers and partners with information about their products and services [Sch03b]. Analyzing the interoperability problem from the XML/SOAP

[5]http://www.ariba.com

point of view can be simplified in layers, as is presented in Figure 2.8. XML is used as the platform independent encoding format. SOAP, which is built on XML and HTTP, defines a simple way to exchange data over different and distributed systems [UDD00]. UDDI is built on top of the SOAP layer, see Figure 2.8.

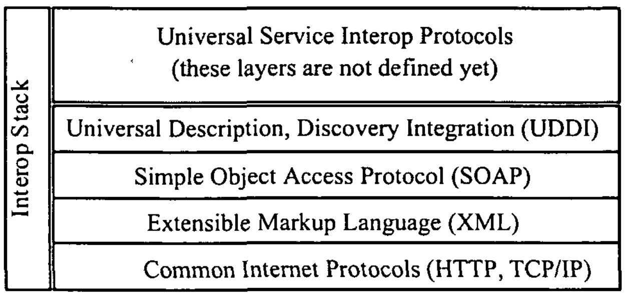

Interop Stack
Universal Service Interop Protocols
(these layers are not defined yet)
Universal Description, Discovery Integration (UDDI)
Simple Object Access Protocol (SOAP)
Extensible Markup Language (XML)
Common Internet Protocols (HTTP, TCP/IP)

Figure 2.8: UDDI is a next layer in Web Services stack [UDD00]

UDDI specification can help to solve the following problems [Sch03b]:

- discover the right business,
- reach new customers and increase access to current customers,
- expand offerings and extends market reach,
- solve customer specific needs and enables them rapid participation in the Internet market, and
- describe services and business processes programmatically in a open environment.

[Vol01] has criticized the automatization process in Web Services, specially the UDDI, since it tries to vanish the business to business(B2B) and business to consumer(B2C) relationship.

Over 220 companies are members of the UDDI community [Bro01].

2.2 Cryptographic mechanisms

Cryptography is the science of writing in secret code. It is, to most people, concerned with keeping communications private. In data storage and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet [Kes03]. "Cryptography is about communication in the presence of adversaries. As an example a classic goal of cryptography is privacy: two parties wish to communicate privately, so that an adversary knows nothing about what was communicated" [Riv94]. Cryptography provides mechanisms that enables communicating parties to achieve [RE99, DK02]:

- Authentication – the process of proving one's identity,

- Confidentiality/Privacy – ensuring that no one can read the message except the intended receiver,
- Integrity – assuring the receiver that the received message has not been changed during transmission, and
- Non-repudiation – preventing the sender from claiming afterwards that he did not send the message.

Security nowadays is not treated as a comfort for paranoid people or a feature "*nice to have*", but as "*must have feature*" in many aspects and applications, including e-commerce, e-payments, private communications, etc.

2.2.1 Symmetric encryption

In symmetric encryption, all participants involved in a transaction use the same key, K for encryption and decryption, as presented in Figure 2.9. If Alice wants to send a message to Bob, she enciphers her message with a symmetric key, using a cryptographic algorithm in encryption mode, and transmits the cipher text to Bob. Bob, at the other side, knows the symmetric key, deciphers the message with the same cryptographic algorithm in decryption mode [Kes03].

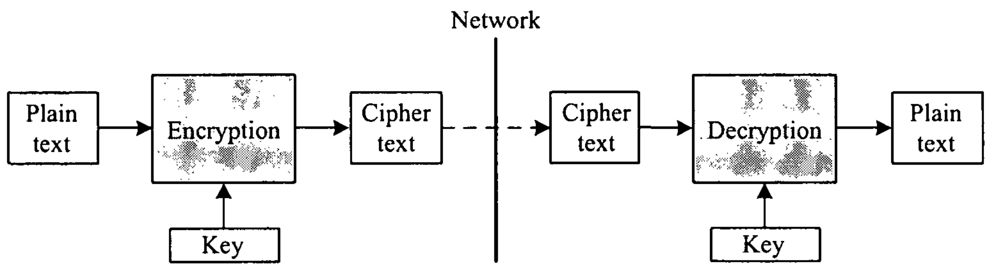


Figure 2.9: Symmetric key encryption

If the plain text is denoted by M and the cipher text is denoted by C, the symmetric encryption E and decryption D can be defined as follows:

$$E(K,M) = C \tag{2.1}$$

$$D(K,C) = M \tag{2.2}$$

The encrypted message may be sent over an insecure communication channel, that is why it needs to be encrypted, but the the key must not be sent over the same channel. Key distribution is one of the main disadvantages of the symmetric encryption mechanisms [FF01].

In [Sch96] several widely used secret key cryptographic algorithms [6] are presented. They are categorized in:

[6]Often an encryption algorithm is referred as a cipher or cryptographic schema.

- stream ciphers – operate on a single bit, byte, or (computer) word at a time, and implement some form of feedback mechanism so that the key is constantly changing, and
- block cipher – operate on one block of data at a time. The two most important block ciphers are [RE99]:
  - Electronic Codebook (ECB) mode is the simplest encryption algorithm. The message block M is divided into small blocks (usually 8 bytes) as M_1, M_2, ..., M_n then each M_i message is encrypted with same key K. Thus two identical plain text blocks will always generate the same cipher text block. This mode is very vulnerable to different brute force attacks.
  - Cipher Block Chaining (CBC) mode adds a feedback mechanism to the encryption algorithm. In this mode, plain text blocks are exclusively-ORed (XORed) with the previous cipher text blocks before encryption. The first block is helped with an initialization vector (IV). In this mode, two identical blocks of plain text always encrypt to different cipher text.

The most widely used secret-key cryptography algorithm today is the Data Encryption Standard (DES) [Kes03]. DES was designed by IBM in the 1970s and adopted by the US National Institute for Standards and Technology [7] (NIST) in 1977 for commercial and government applications [ANS81]. More information regarding DES background can be found in [Sch96]. DES is a block algorithm that uses a 56-bit key [8] and operates on 64-bit blocks. The message is divided into 64-bit blocks and if the last block is smaller than 64-bit it is filled (padded) with zeros.

Nowdays DES is considered a weak algorithm. It is breakable within minutes with today's technology with brute-force attack [EFF99]. Therefore, new DES improvements and new algorithms have been proposed. The triple DES is now widely used as a stronger variant of DES, in which two different keys, K_1 and K_2, are used:

$$C = E(K_1, D(K_2, E(K_1, M))) \tag{2.3}$$

This was possible because DES is not an algebraic group, as was proven by Cambell and Winer in 1992 [Has01]. This means that for the given keys K_1 and K_2 there is no key K_3 such that:

$$E(K_2, E(K_1, M)) = E(K_3, M) \tag{2.4}$$

There are a number of other secret-key cryptography algorithms that are also in use today:

- CAST-128 is a DES-like crypto algorithm, described in [Ada97] that uses a 128-bit key operating on a 64-bit block. Its successor, CAST-256 uses a 128-bit block size and a variable key length (128, 160, 192, 224, or 256 bit). CAST is an open algorithm and is internationally available [AG99].

[7] http://www.nist.gov

[8] Indeed the key is 64-bit long, but since each byte has a parity bit, the effective key size reduces to 56-bit.

- The International Data Encryption Algorithm (IDEA) is a DES-like 64-bit block algorithm that uses 128-bit keys. It applies the same basic cryptographic techniques and is twice as fast as DES [Sch96]. This was the European answer to DES and to the US export restrictions on cryptographic algorithms [Has01].
- RC2 is designed by Ron Rivest for RSA Data Security Inc.. RC stands for Rivest cipher and has a variable key size on a 64-bit block cipher and was designed to be a replacement of DES [Sch96].
- Skipjack is the National Security Agency (NSA) encryption algorithm for the Clipper [9] chip. The algorithm is secret and therefore its details have never been published. It is implemented only in tamperproof hardware [Sch96].
- Blowfish is a symmetric 64-bit block cipher invented by Bruce Schneier. It is optimized for 32-bit processors and it is significantly faster than DES. Blowfish uses a variable key length from 32 to 448 bits. It is available for free and used in over 80 products [Kes03].

In 1997, NIST started a procedure to develop a new secure crypto-algorithm for US government applications. In April 1999, NIST announced the five finalists:

1. MARS (Multiplication, Addition, Rotation and Substitution) is developed by IBM, supports 128-bit blocks and has a variable key size [BCD+99].
2. RC6 is a block cipher designed by Ron Rivest and RSA Laboratories. RC6 is a parameterized algorithm where the block size and key size are variable with maximal 2040 bits key size [RRSY98].
3. Rijndael is a block cipher designed by Joan Daemen and Vincent Rijmen. The cipher has a variable block and key length [DR99, NBB+00].
4. Serpent is a 128-bit block cipher designed by Ross Anderson, Eli Biham and Lars Knudsen [ABK99].
5. Twofish is a block cipher with a variable key size up to 256 bit. Twofish is developed by team lead by Bruce Schneier. It is available for free [Sch99].

In Table 2.3 the properties of the five algorithms are summarized:

	MARS	RC6	Rijndael	Serpent	Twofish
General security	3	2	2	3	3
Impl. of security	1	1	3	3	3
Software performance	2	2	3	1	1
Hardware performance	1	2	3	3	2
Smartcard performance	1	1	3	3	2
Design features	2	1	2	1	3

Table 2.3: Properties of the five NIST finalists (1=low, 3=high) [NBB+00]

In October 2000, NIST announced their selection: Rijndael. It become the official successor to DES in December 2001 [Kes03].

[9]Also known as the MYK-78T chip, this is an NSA designed tamperproof VLSI chip designed for encrypting voice conversations [Sch96].

2.2.2 Asymmetric encryption

Asymmetric encryption uses two different keys, one public and one private. The public key is freely shared among communicating participants but the private key is kept secret. Figure 2.10 presents the basic principle of asymmetric encryption. Systems where asymmetric encryption is used are also known as public key cryptography (PKC). PKC was first described in 1976 at Stanford University by Martin Hellman and Whitfield Diffie [Sch96].

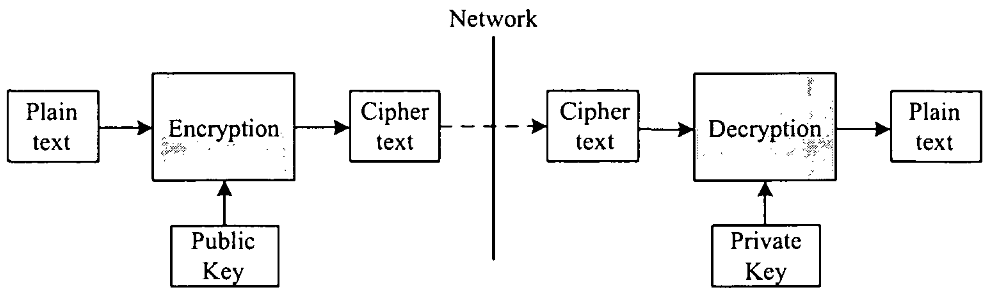


Figure 2.10: Asymmetric key encryption

In PKC when the encryption is applied, always it is assumed that the encryption is made with public key[10] and decryption is always performed with private key. Asymmetric encryption is about 100 to 1000 times slower that symmetric encryption [FFW99]. Therefore, asymmetric encryption is rarely used to encrypt large amounts of data. It is usually used in combination with symmetric encryption whereby only the secret key is encrypted with asymmetric encryption methods and data blocks are encrypted with symmetric encryption methods [Zim00].

The two most commonly used asymmetric encryption methods are:

- RSA (named after its developers, Rivest, Shamir and Adleman) is the most popular public key algorithm. The key length is variable usually between 512 and 2048 bits. A longer key increases the security but also decreases efficiency and produces more cipher text. The plain text block can be of any length but it must be smaller than the key length. The encrypted cipher text block is the same length as the key [FFW99].

  Mathematically, the RSA public key has the form (e, n), where the e is the exponent and n is the modulus. The n is also the key length in bits. The modulus n is chosen randomly and is obtained by multiplying two large prime numbers p and q. The values of p and q are kept secret. The private key has the form (d, n), where d is the multiplicative inverse of the e modulo $(p-1)(q-1)$ [Has01]. Interesting and simply described mathematical details about RSA can be found at [MB01].

  If Alice sends an encrypted message to Bob she calculates:

[10]Encrypting with private key would not make any sense regarding privacy, because anyone who has the sender's public key can read the message! But this does ensure all receivers that the message comes from the claimed sender.

$$C = M^e \ modulo \ n \tag{2.5}$$

where (e, n) is the Bob's public key. Bob decrypts the message by calculating:

$$M = C^d \ modulo \ n \tag{2.6}$$

where (d, n) is the Bob's private key. The security of RSA algorithms is based on the difficulty of factoring the modulus n [Sch96].

- Elliptic curves were proposed for use to be used in cryptography in 1985 by Neal Koblitz and Victor Miller. Elliptic curve methods have been proved to require much shorter keys than RSA to achieve the same level of security [Pie00]. This feature is very important in cases when the processing is done on devices with limited processing capacity such are smartcards. A mathematical explanation of elliptic curve is much more complicated than of RSA [Cer00]. A 1024-bit RSA key is approximately as secure as a 160-bit elliptic curve key [Has01].

2.2.3 One way hash functions

One way functions $h(x)$ are relatively easy to compute for a given x, but significantly harder (indeed it could take million of years), to compute the reverse function $h^{-1}(x)$. A trapdoor one way function is a special form of one way function, whereby some secret information y exists that helps computing $h^{-1}(x)$ [Sch96].

A hash function also known as the message digest (fingerprint or cryptographic checksum) is another building block in cryptographic protocols. A hash function takes as input parameter a variable length data block and outputs a fixed length (generally smaller) data block. One way hash functions work only in one direction, i.e. it is easy to compute the hash value of a given data block but it is very hard to generate the input data block from hash value [FFW99]. The secrecy of the one way function is that it works only in one direction.

Among the most common hash functions used today in commercial cryptographic applications is a family of Message Digest (MD) algorithms, all of which are byte-oriented schemes that produce a 128-bit hash value from an arbitrary-length message [Kal92, Riv92a, Riv92b]. The Secure Hash Algorithm One (SHA-1), proposed by NIST for their Secure Hash Standard (SHS), is being used in commercial products today. SHA-1 produces a 160-bit hash value [Has01].

2.2.4 Digital signatures

Signatures written by hand are used as proof of an authorship and agreement with the contents of the document. With the use of asymmetric encryption, it become possible for the first time to sign a document digitally, i.e. an equivalent to a handwritten signature. This idea was first presented by Diffie and Hellman in 1976. Encrypting a document with a private key produces a digital signature [Sch96].

If Alice wants to send a signed document to Bob, she does the following:

1. Alice encrypts the document with her private key, thereby signing the document,
2. Alice sends the document to Bob, and
3. Bob decrypts the document with Alice's public key, i.e. verifies the signature over the document.

Alice will find it difficult to repudiate the fact that she sent the message because she is the sole owner of her private key. In [Dev01] various criticism of such signing and decrypting procedure are presented.

In real applications the document is usually hashed and the hash value is encrypted with the owner's private key, as presented in Figure 2.11. Signing the complete message requires more computing power and performance degradation [FFW99].

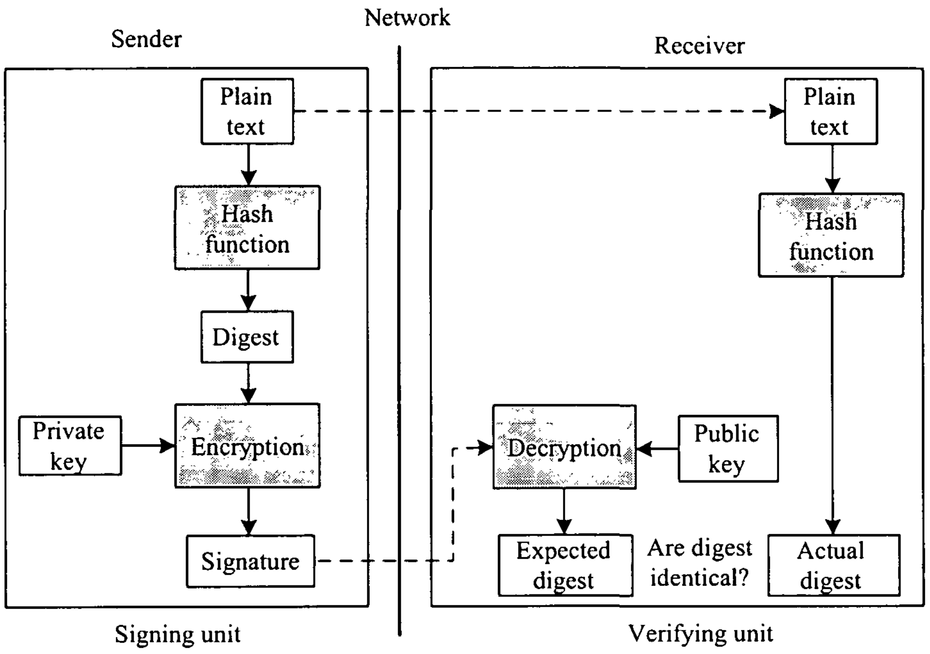


Figure 2.11: Digital signature generation and verification

Mathematically the signature can be represented as:

$$S = E(h(M)) \tag{2.7}$$

To verify the signature it is necessary to know (see Figure 2.11):

- the message (M) over which is signature computed,
- the signature (S) of the message,
- the signer's public key (e, n),
- the hash function, and
- the signature algorithm used to generate the signature.

The digital signature algorithm and RSA are the two algorithms for digital signature generation and verification recommended by the Digital Signature Standard [NIS00].

There are cases when a trusted party signs a message presented by other parties. In order to prevent that the signer from reading the document before signing, the blind signatures were introduced by David Chaum [Cha88].

2.2.5 Key management

Choosing the right algorithm, key length, key store, key backup, and key distribution are only a few problems that key management must face.

Generally the security of a cryptosystem must not depend on keeping the encryption algorithm secret, but instead on keeping the key secret. This is known as Kerckoff's principle, named after the Auguste Kerckhoff presented in his book "La cryptographie militaire" in 1883.

The security of a cryptosystem can be represented as a function of two parameters [Sch96]:

- strength of the algorithm, whereby one algorithm is considered cryptographically secure if it does not provide any method to break the algorithm other than trying all possible keys [11], and
- key size (represented in bits) simply because the longer the key size, the greater the number of all possible keys.

The total security of the cryptosystem is at least as strong as the weakest component in the system [ES00]. The length of a secret key must be large enough to prevent a brute-force attack. The Electronic Frontier Foundation (EFF) in January 1999, with distributed computers over the Internet cracked a DES key in less then 22 hours and since then standard, simple DES is considered insecure [EFF99]. Table 2.4 estimates the cost of sufficient computing power to crack the DES algorithm in 2005. The values are derived from estimates in 1995 in [Sch96] and Moore's Law, which projects that computing power doubles every 18 months [12].

Cost($)/Bits	40	56	64	80	112	128
100 K	20 ms.	21 min.	4 days	700 years	10^{12} years	10^{17} years
1 M	2 ms.	2 min.	9 hours	70 years	10^{11} years	10^{16} years
10 M	0.2 ms.	13 s.	1 hour	7 years	10^{10} years	10^{15} years
100 M	0.02 ms.	1 s.	5.4 min.	245 days	10^{9} years	10^{14} years
1 G	2 micro s.	0.1 s.	32 s.	24 days	10^{8} years	10^{13} years

Table 2.4: Average time estimates for a brute force attack in 2005 based on key length in bits

[11] This method is also known as brute-force attack.

[12] This means that the costs go down by a factor of 10 every 5 years.

The nature of brute force attacks against public key algorithms differs from secret key algorithms. A brute force attack on asymmetric algorithms is transformed into the mathematical problem of attempting to recover the private key based on knowledge of the public key. The attack involves factoring very large numbers in the RSA case or taking the discrete algorithms in very large finite fields in case of elliptic curves algorithm [FFW99]. A comparison between key length in different cryptosystems is presented in Table 2.5. More about selecting the right key size can be found in [LV01].

Secret key	RSA public key	Elliptic curves
56	384	80
64	512	106
80	768	132
96	1024	160
120	2048	211

Table 2.5: Relative strength in bits of secret keys vs. public keys [FFW99]

In a symmetric encryption mechanisms one the biggest problems is key distribution. How do Alice and Bob exchange their keys if they cannot meet personally? One solution would be dividing the secret key into n pieces and then transmitting it over n different channels [Sch96], as presented in Figure 2.12.

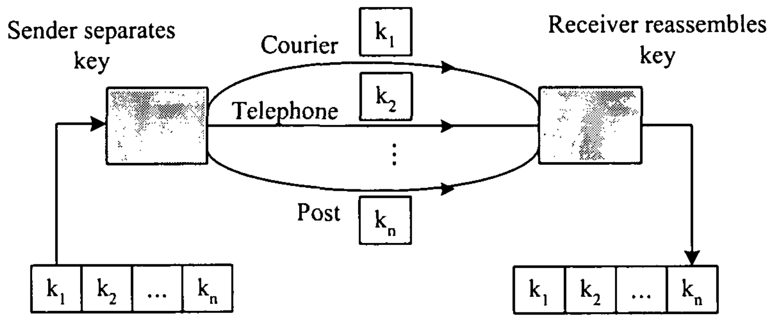


Figure 2.12: Key distribution via n parallel channels [Sch96]

Exchanging the symmetric key in a large network requires more effort than exchanging asymmetric keys. For an n person network there are $n(n-1)/2$ key exchanges needed. For this reason the notion of a Key Distribution Center (KDC) (or Trusted Third Party (TTP)) was invented. These centers suffer from lack of scalability and requires a considerable amount of work for key administration [MB01]. When Bob receives a public key from Alice, he must be sure that the public key really belongs to Alice, otherwise someone else can masquerade as Alice. Secure distribution of public keys is achieved through digital certificates and certification authorities (CA), see Section 2.3.

Key storage is the responsibility of the user. The easiest and most secure approach is to store the key in the user's brain and not in a computerized storage system [Sch96]. The user is then responsible for remembering the key and entering it whenever necessary. These sorts of keys are called mental keys. Another method for storing the key is dividing the key into two parts, the first part is stored in the terminal and second part is stored in a hardware device like a smartcard for example (see Section 2.4). Smartcards have been proven to be tamper-resistant, secure and reliable devices in the market [Bra00]. The access to a key is protected with another key, such as a mental keys or so called personal identification number (PIN).

Key management has to face with the problem of *what to do in the case where the secret key is cracked* or it become widely known. The most common rules of key management, based on [RE99] are:

- To use derived keys instead of a main key. The derived key is obtained from the main key using any known cryptographic algorithm and some unique property.

  $$DerivedKey = E(MainKey,\ UniqueProperty) \tag{2.8}$$

  Usually as the encryption method a triple DES algorithm is used.

- To use different keys for each application instead of using one key for all applications.

- To use key versions, which means that one key is valid only for a specified time interval, after which it becomes invalid. The system must be capable of switching automatically to another key.

- To use so called dynamic (session) keys instead of a main key. This approach is widely used on the Internet. The session key is generated cryptographically, usually with triple DES, using the main key and some random numbers.

  $$SessionKey = E(MainKey,\ RandomNumber) \tag{2.9}$$

  The advantage of this approach is that for every session a different key is used and therefore even cracking one's session key does not compromise the overall security of the system.

2.2.6 Authentication and Authorization

The process of proving the identity of the communications partner is called authentication [Knu98]. Authentication mechanisms are based on three models [FFW99]:

- something you know, whereby the communication partner [13] knows a shared secret, such as a password or PIN,

- something you have, the communication partner demonstrates the possession of something, such as a physical key, smartcard, USB dongle etc., and

- something you are, means the communication partner has something immutable such as fingerprint, face or retina pattern, voice etc.

[13] Communication partner can be a person, a PC or any electronic device.

Withdrawing money from an automatic teller machine (ATM) is the best example of requiring a combination of something you know (PIN) and something you have (bank card).

Authentication can be one-side or mutual depending whether one communicating partner authenticates the other or they are authenticating each other. Figure 2.13 presents a classification of authentication methods. Authentication between ATM and bank card is usually a mutual authentication that uses a challenge-response method based on symmetric algorithms [RE99].

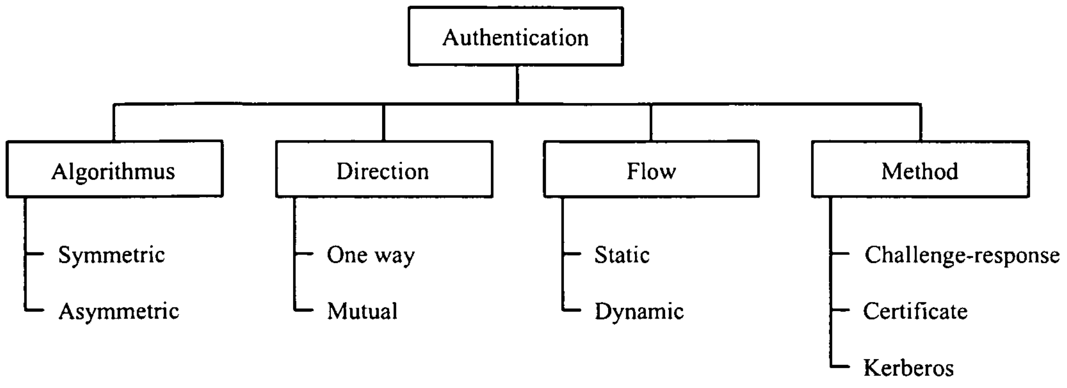


Figure 2.13: Classification of authentication methods [RE99]

Even authorization does not belong to cryptography, it is widely used in conjunction with authentication. After successful authentication, the communication partner gains the rights to execute different actions at the other side or terminal. Authorization is the process of allocating rights to the communication partner [Man98].

2.3 Public Key Infrastructure

Public Key Infrastructure (PKI) is the "set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke public key certificates (PKC) based on public-key cryptography" [AT02]. PKI includes the certificate storage resources of a server, and also provides users a set of services and protocols for managing public keys. The main feature of PKI is the introduction of what are known as a Certification Authority (CA) and a Registration Authority (RA) [Zim00].

PKI is a basic protocol on which relay Secure Multipurpose Internet Mail Extensions (S/MIME), Transport Layer Security (TLS), Internet Protocol Security (IPSec), Virtual Private Network (VPN), online shopping and online banking [Hun01]. These protocols provide services such as confidentiality, data integrity, authentication and non-repudiation.

2.3.1 Architecture model

A PKI consists of five types of component [BDNP97]:

- The Certification Authority (CA) is the highest (root) instance, which is trusted by participants to generate, assign and revoke public key certificate (PKC).
- The Registration Authority (RA) is an optional entity. Its responsibility is to verify that the subjects's identity values matches the PKC request. The RA is also responsible for verifying that the subject possesses the private key as stated in the PKC request.
- PKC holders who get the issued certificates. PKC holders (also called end entities) can sign digital documents and decrypt documents using their private keys. A PKC is also known as an identity certificate (IC).
- PKI enabled applications that validate digital signatures and their certification paths from the known public key of a trusted CA. PKI enabled applications can encrypt documents using the public key from certificates of PKC holders.
- Repositories are public online resources that provide certificates and certificate status information [Bur98].

Figure 2.14 presents a simplified view of the architectural model assumed by the PKIX[14] Working Group [AT02].

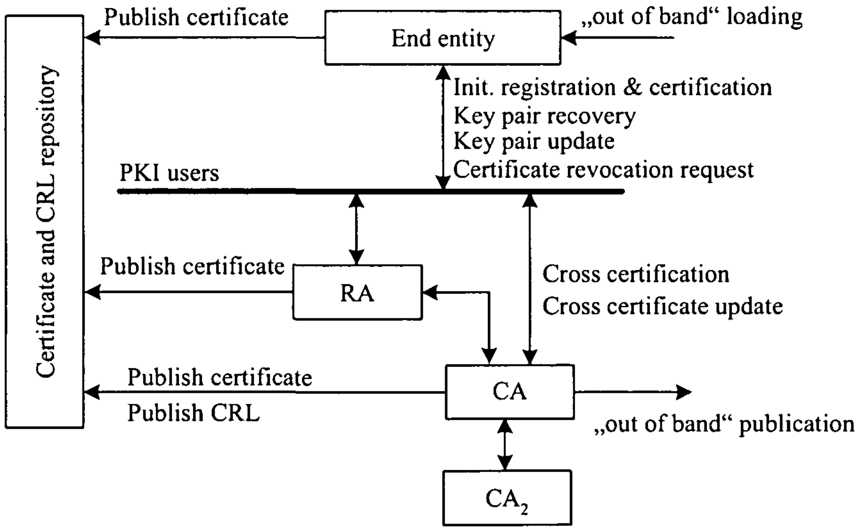


Figure 2.14: PKI entities

The end entity sends its certificate request to the RA (or in the case when RA is missing, direct to the CA) for approval. If it is actually approved, it is forwarded to the CA for signing. The Certification Authority verifies the certificate request and if it passes the verification, it is signed and the certificate is created. In practice there are two modes distinguished for generating digital certificates: the automatic mode where the certificates are generated by software without human intervention and the manual mode where each certificate must be approved by an administrator [Mic00a, Ope03b, Xen00]. To publish the certificate, the CA sends the certificate to the repository where it can be obtained (usually downloaded) from the end entity.

[14]http://www.ietf.org/html.charters/pkix-charter.html

2.3.2 X.509 digital certificates

Digital certificates were first proposed by Loren Kohnfelder, in his master thesis at MIT [15] in 1978, as a solution for efficient and authentic binding of private keys to end entities. This makes the use of public key cryptography practical for digital signatures. Digital certificate simply binds the public key with the corresponding name of some entity [Gut00]. Digital certificates can be sent through insecure networks and can be stored in insecure media, because if any changes are made to the certificate the signature will be invalid [AFPS99]. Digital certificates are usually not confidential and therefore can be freely stored in insecure public repositories.

A digital certificate can be compared with passport in everyday life. A passport binds personal information such as photo, name, gender, address and birthdate to a person and is valid for a finite period of time. All this personal information in the passport is verified by an entity like a government department; similarly the information in a digital certificate is verified by a CA [Chi02].

In order to fulfill interoperability during the exchange of certificates in different systems the International Telecommunication Union (ITU) defined a standard in 1988 about X.509 Certificates as part of the X.500 directory recommendations [ITU00]. The current version of X.509 certificates, (v3), was released in June 1996 [AFPS99].

The format of an X.509 v3 certificate is presented in Figure 2.15. It consists of a set of required and optional fields.

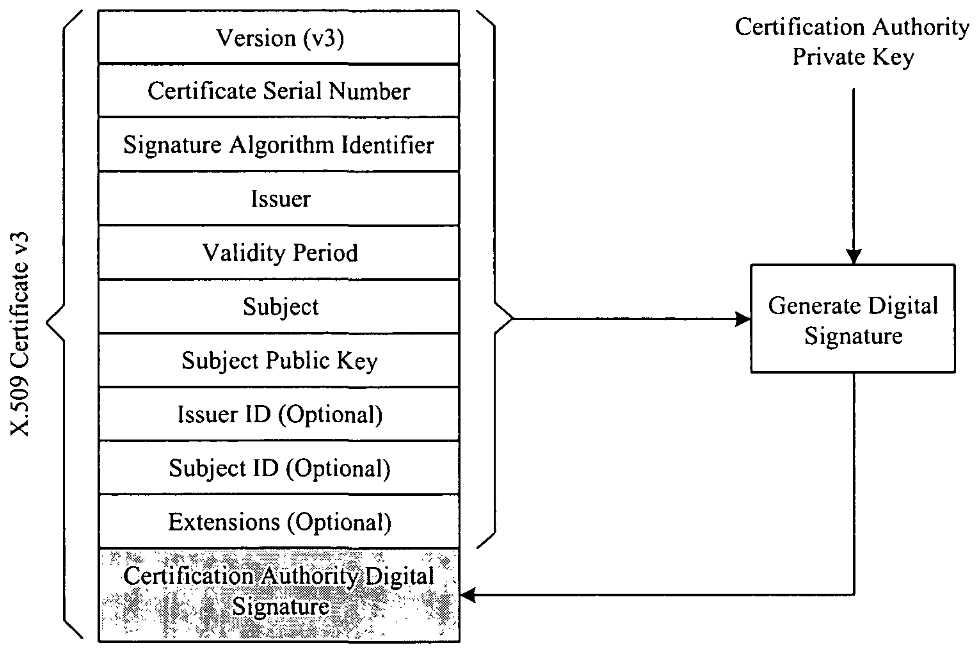

X.509 Certificate v3
Version (v3)
Certificate Serial Number
Signature Algorithm Identifier
Issuer
Validity Period
Subject
Subject Public Key
Issuer ID (Optional)
Subject ID (Optional)
Extensions (Optional)
Certification Authority Digital Signature
Certification Authority Private Key
Generate Digital Signature

Figure 2.15: X.509 v3 certificate format

A detailed description of X.509 certificate fields can be found in [Gut00, AFPS99]. The extension field is very interesting, since anyone can define one's own extensions and put them into certificate.

[15]http://www.mit.edu/

The format of an X.509 certificate is specified in Abstract Syntax One (ASN.1)[16] and encoding follows in Distinguished Encoding Rules (DER). The certificate data structure fields so encoded are reduced to a lists of more fundamental elements, whereby each element is either a primitive type or another list. The primitive types are then encoded in tag, length and value format (TLV) [FFW99]. [Paa00] suggest that the certificates should be encoded in XML, in order to avoid the problems with TLV encoding and to enjoy all the benefits of XML.

An X.509 certificate has a finite validity period. In the case of premature withdrawal of the certificate or if the associated private key is compromised, the question arises of how PKI enabled applications are to be informed about it. Certificate revocation lists (CRL) are mechanisms that are used to publish and distribute information about revoked certificates to PKI enabled applications [Rih98]. A CRL is data structure, digitally signed by the CA, that contains date and time of the CRL publication and the serial number of all revoked certificates [ITU00]. The X.509 CRL format is an ITU and ISO standard with current version 2 (v2), a detailed description of all CRL fields is in [Gut00, AFPS99]. The immediate question that arises about CRLs is how oft they should be updated. Even if the frequency is determined by CA, [Riv98] states that the acceptor should set the recency requirement on the CRL since he faces a risk dealing with invalid certificates. If PKI enabled applications do not check the CRL they are close to useless [Ger00]. Generally there are three methods that allow CRL propagation [FFW99]:

- Polling – of the current CRL, is a mechanism used by PKI enabled applications after each time the CA issues a new CRL. The update schedule is kept in the CRL data structure. The disadvantage of this approach is that the problem of having the actual CRL still exists. The update period of CRL can be kept short, but *short* is not tolerated by some time critical PKI enabled applications even if the CRL is updated hourly.

- Pushing – is used by the CA as soon the CA revokes a certificate. The advantage of this approach is that the PKI enabled application always receives the current CRL. [FB01] describes the problems that appear in real PKI enabled applications, specially in the network infrastructure, using this approach.

- Online – status checking is the most reliable method to determine the revocation status of the certificate [Arn00]. The advantage of this approach is that it does not require to push or pull a large amount of data over the network. However this approach requires that the CA should be available and reliable all the time.

In [Gut02] interesting workarounds are proposed for minimizing the CRL shortcoming, such as to differentiate when key is compromised between user group changed or to use a so called delta CRL.

The lifetime of a certificate depends on several factors, as pointed out in [Sim97], and the optimal life time should be shorter as weeks. This approach leads to the idea to use certificates only once and thus completely eliminate the need for CRLs.

[16] ASN.1 is ISO notation for describing abstract types and values.

2.3.3 Pretty Good Privacy

Pretty Good Privacy (PGP) is public key encryption mechanism in the application layer, mainly used for email encryption, originally written by Phil Zimmermann in 1991. PGP is available worldwide in freeware and commercial versions.

PGP is a hybrid cryptosystem. PGP combines the features of symmetric and asymmetric cryptography. When a user encrypts plain text with PGP, PGP first creates a session key, which is a random number generated from keystrokes the user types and random movements of the mouse. The session key then is used with any symmetric methods to encrypt the plain text. The session key itself is encrypted with the recipient's public key (asymmetric encryption), as in Figure 2.16. This public key-encrypted session key is transmitted along with the cipher text to the recipient [Zim00].

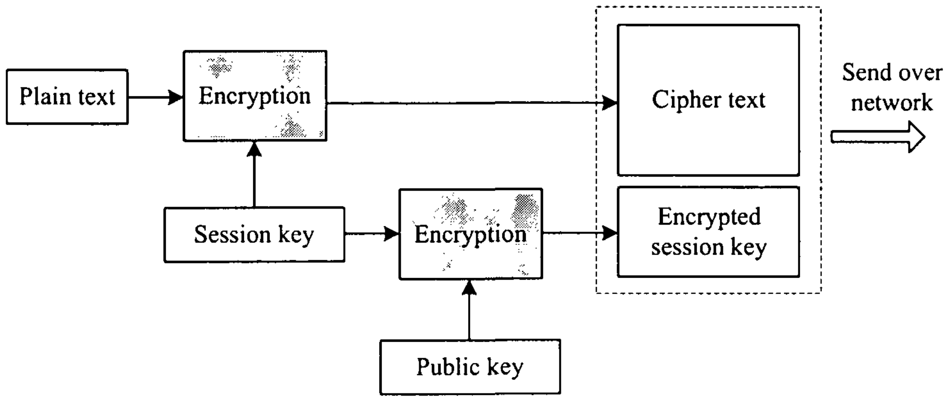


Figure 2.16: PGP encryption flow [Zim00]

Key distribution in PGP is done without any hierarchy. Public keys are simply sent from user to user, whereby each user could sign the key and therefore certify the key as trustworthy, i.e that it belongs to the user stated in certificate. This is known as introducing a key [Ger00], thus creating a so-called web of trust. A PGP certificate contains personal information identifying the owner, such as [Zim00]:

- name (and usually email address) of the key holder,
- public key, and
- at least one signature over two fields.

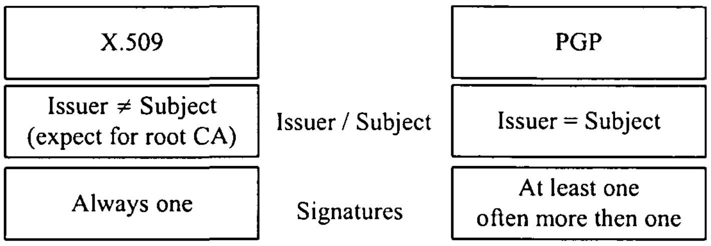
The PGP certificate can have more than one signature, as any user can sign it. If it has only one signature it is called self signature and is part of every PGP certificate.

The main disadvantage of PGP certificates is with self signed signatures because they are easy to forge [MB01]. Withdrawing a PGP certificate in PGP PKI is very difficult since there are no central authorities to manage the revoked certificates.

2.3.4 Comparing X.509 and PGP certificates

X.509 and PGP have similar features. X.509 and PGP are both based on public key encryption and require certification to ensure that the public keys used are valid and belong to those whom they claim. X.509 and PGP are both key transport protocols and both depend on two external references: trust and keys [Kho99].

The main difference between X.509 v3 and PGP certificates is that X.509 uses a hierarchical model, where the root CA is responsible for managing the certificates, and PGP use a web of trust model. Another difference is the format, whereby PGP allows more signatures to be attached to a certificate (see Figure 2.17) and X.509 allows only one way linked certificates. It is hard to use the PGP in a commercial situation since PGP does not have an entity responsible for certificate managing [Ger00].



X.509		PGP
Issuer ≠ Subject (expect for root CA)	Issuer / Subject	Issuer = Subject
Always one	Signatures	At least one often more then one

Figure 2.17: X.509 certificate vs. PGP certificate [MB01]

Compared with an X.509 PKI, a PGP based PKI has less bureaucracy that must be managed and administered. In classical PGP based PKI the central CA does not exist and thus users are responsible themselves for trusting and managing the digital certificates [Car00].

Revoking a key in PGP is also much more difficult than in X.509. This is due to its decentralized approach (all users that have the key have to be informed that the key is revoked). In PGP no central place exists where the information regarding the revocation of a key can be placed. This information has to be propagated to all users having the public key in their environment. This is because each user has it's own trust model, therefore all of them have to be updated. This can be a very difficult task if the user has distributed his public key widely. Thus in PGP there can be a lack of synchronization. The users might not have the latest key states in their individual trust model [Kho99]. In X.509 this feature is provided by the CA.

With PGP version 8.0 it is possible to import and export X.509 certificates into PGP, furthermore with this version it is possible to generate an X.509 certificate request [PGP03].

2.4 Smartcards

One of the fundamental problems of cryptographic algorithms is the need for tamper-resistant and secure storage of keys [Her02]. Otherwise cracking the encryption algorithms with highly sophisticated and expensive devices would be worthless if the keys are free accessible. The most secure place to store keys is the user's brain [Sch96], but who can or has the will to remember a 2048 or 4096 bit RSA key?

A smartcard is a portable, tamper-resistant processor with a programmable data store and data processing [GJ98]. It has different size formats, but mostly has the shape and size of a credit card and holds sensitive information in the range from 1 to 64 kilo bytes (KB). The new smartcards, like the Infineon [17] SLE66CX family, have a central processing unit (CPU) a 16 bit microcontroller, 64 kB of EEPROM [18] and an advanced crypto engine (ACE) for cryptographic operations such DES, triple DES and RSA operations [Inf01a].

Smartcards have proved secure, tamper-resistant and very suitable devices for storing sensitive information such as keys and certificates [Bra00]. Smartcards can process data in an intelligent form by taking different actions based on secret data that never leaves the card. Memory access to smartcards is guarded by PIN and only after successful PIN presentation does the smartcard perform security operations. In the case of several false PIN presentation the smartcard is blocked for further memory access, thus protecting the information it holds from unauthorized access [GJ98].

2.4.1 Background and classification

Diners Club [19] was the first company to use synthetic plastic cards to enable cash-less payment transactions, from the beginning of the 1950s. The acceptance of this card was limited at the beginning to only luxury restaurants and hotels, but with the introduction of VISA [20] and MasterCard [21] the acceptance of cash-less cards has grown worldwide. But plastic cards have a big disadvantage: they are very easy to forge. Due to progress in microelectronics in the 1970s, it was technically possible in a few square millimeters to integrate a chip with modest data processing and data storage. Almost at the same time in Germany, France and Japan there were patents applied regarding the smartcard. The break through of smartcards was achieved in the mid of 1980s when France and Germany decided to use the smartcards in public phone terminals, as a prepaid card. More details about history of smartcards can be found in [RE99] [22].

Smartcards can be classified into different groups depending on technology, security, programming language, communication interface etc. A classification of smartcards based on chip type and communication interface is presented in Figure 2.18.

Smartcards are fully standardized through the ISO/IEC [23] 7816 standard for contact based smartcards and through standards 10536, 14443 and 15693 for contactless cards [RE99, Fin00].

Smartcards are divided into three main groups [RE99]:

- Memory cards – are used en masse as prepaid telephone cards. The counter value stored in the card can only be decremented due to the built in security

[17] http://www.infineon.com

[18] Electrical Erasable Programmable Read Only Memory.

[19] http://www.dinersclub.com

[20] http://www.visa.com

[21] http://www.mastercard.com

[22] [RE99] is one of the most complete reference book on smartcards today, stretching from smart-card history to the real application cases nowadays and analyzing smartcards from application view as well as in bits and bytes.

[23] International Organization for Standardization and International Electrotechnical Commission respectively.

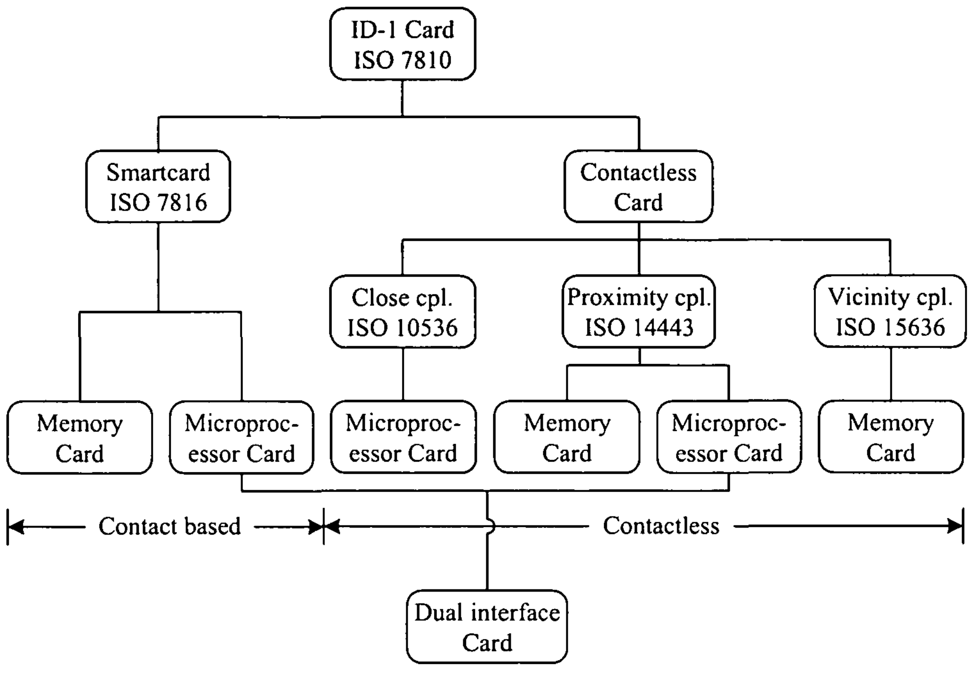


Figure 2.18: Smartcard classification tree with respective ISO standards [Fin00]

logic. And thus, when the counter reaches zero the card is worthless, but due to their cheap manufacturing process they have been profitable. Beyond the public phones the memory cards can be used anywhere where cashless infrastructure exists such as vending machines, parking, cafeterias etc.

- Microprocessor cards – were first used in France as bank membership cards. The number of applications to run on microprocessor cards has grown rapidly due to the price fall of the microprocessors. Applications that run on such cards are: access control, electronic purse, digital signature, digital ID etc. The new microprocessor cards have separated the operating system (OS) layer from the application layer, thus enabling many application to run on a single card without violating the security and integrity of other applications. With the introduction of the Java card [HMGM02, GJ98] it is possible to load applications onto card after the card is issued to the end user.

- Contactless cards – have an antenna embedded inside the card and communicate with a reader via radio frequency (RF). Contactless cards based on their operating distance are divided into three subgroups [Fin00], as in Figure 2.18:

  - Close coupling cards, which are standardized in ISO 10536 and have an operating range of 0 to 1 cm. These systems can work in any frequency to 30 MHz. Due to the close contact it is possible to transmit high energy values from reader to card, allowing the use of microprocessors with huge power consumption [Fin00].
  - Proximity coupling cards operate on the 0 to 10 cm range and are standardized in the ISO 14443 standard. These cards support multi applica-

tion and key management, and it is possible to download third applications after the card is personalized [Inf01b]. These cards are ideally suited for access control, electronic purse, ski and public transport ticketing etc.

- Vicinity coupling cards operate in long ranges up to 1 m and are very suitable for tracking the goods in automated systems. A complete list of possible RF applications is given in [Fin00].

In [YKMY01] a PKI based payment system is presented based on contactless smartcards.

The smartcard market is still growing, especially contactless cards, which are going to dominate in the market. Due to their fast memory access and the flexibility to attach them to every article such as books, newspapers, every food article, even passports and banknotes, make them perfect to incorporate in automated tracking systems.

2.4.2 Architecture

A typical smartcard architecture is presented in Figure 2.19 [24]. The typical smartcard contains a central processing unit (CPU) which might be a 8 or 16 bit processor. The smartcard operating system (OS) is loaded into read only memory (ROM) which has a size of 16 to 128 kilo bytes (KB). The user data, in some cases some parts of the operating system data [25], and application data are stored in EEPROM. The EEPROM can be of size up to 64 KB. To satisfy modern digital signature laws, i.e. to use an asymmetric encryption algorithm like RSA in a reasonable time, the smartcards have a cryptographic coprocessor to perform these operations. Smartcard architecture follows the modern operating system layered structure, where applications are defined on the top of the operating system.

ISO 7816-3 specifies the intelligent parts of the smartcards. This standard describes [ISO97a] the relationship between the smartcard and the reader as one of slave (the smartcard) and master (the reader). Smartcards communicate with smartcard readers through a serial interface, for contact-based or through antennae for contactless cards. The speed of the serial interface in early days was usually limited to 9600 bits per second (bps). The latest smartcards, such as the Infineon SLE66CUx640P [Inf01a] and SLE88CX720 [Inf03b] use an internal clock generation and thus achieving data transmission up to 112 Kilo bits per second (Kbps). The communication channel is single threaded, i.e. once the reader sends a command to the smartcard, it blocks the communication channel until a response is received. The new smartcard from Infineon (SLE88CX720P) supports so called multichannel, i.e. it is multi threaded [Inf03b].

When the smartcard is inserted into a reader, the smartcard sends the answer-to-reset (ATR), hereby establishing a simple communication path. The ISO7816-4 [ISO95] standard specifies the content of the messages, commands and responses transmitted

[24]The figure is updated from [RE99, Inf01b].

[25]Putting part of the OS in EEPROM degrades the performance of the smartcard regarding the slower time access and limited write cycles. EEPROM has a limited number (usually 100.000) of write cycles and thus putting a OS variable which needs to be frequently updated decreases the number of write cycles.

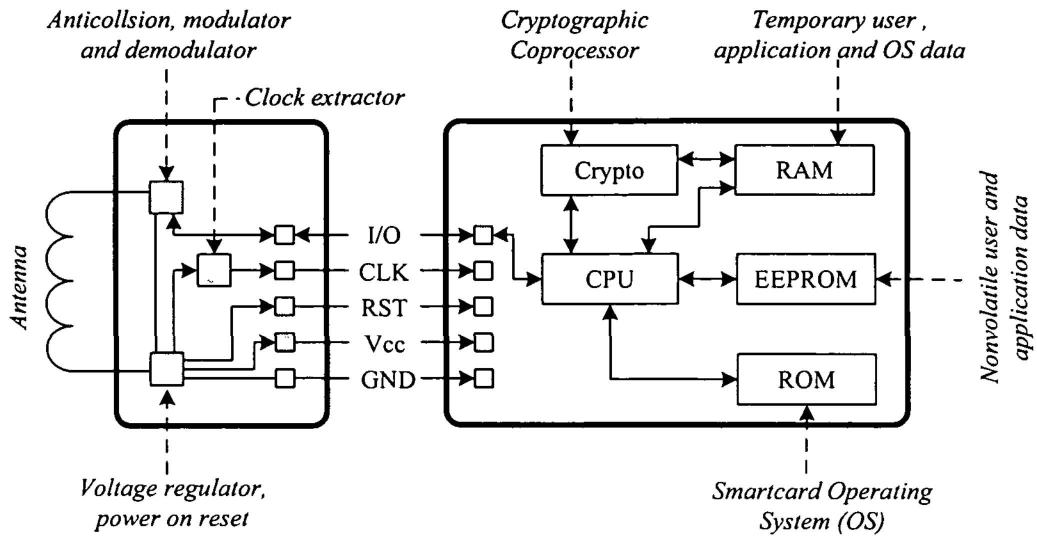

Anticollsion, modulator and demodulator
Clock extractor
Cryptographic Coprocessor
Temporary user, application and OS data
Antenna
I/O
CLK
RST
Vcc
GND
Crypto
RAM
CPU
EEPROM
ROM
Nonvolatile user and application data
Voltage regulator, power on reset
Smartcard Operating System (OS)

Figure 2.19: Typical smartcard architecture with a contact based and contactless interface

by reader to smartcard and vice versa. Furthermore the ISO7816-4 standard defines two application protocols:

- File structure - is accessed through a predefined set of application programming interface (API) functions, whose main aim is to provide a consistent file system for the storage and retrieval of information on the card.
- Security services - are a set of API functions which allow the card and the card reader to authenticate each other's identity. Security services provide the mechanisms through which the two communicating sides can keep their information exchange private.

In order to support API calls from the application layer the ISO7816-4 has defined a protocol message structure that is exchanged between the smartcard and the smartcard reader. The message structure contains the functions calls, their associated parameters, and status response parameters. The protocol message structure is divided into so called application protocol data units (APDU), which are exchanged between the smartcard and reader by the link layer protocol (generally either a T=0 or T=1 protocol) [GJ98]. In Figure 2.20 APDU message structure is presented.

The APDU message can be either [ISO95], as in Figure 2.20:

- Command APDU - which consists of the four mandatory header bytes (CLA, INS, P_1, P_2) and conditional body of variable length. The CLA (class) and INS (instruction) byte codes contain an application class and instruction group as described in the ISO 7816-4. The P_1 and P_2 parameters are used to qualify specific commands and are therefore given specific definitions by each [CLA, INS] instruction. The body of command APDU is a optional component that has variable length. It is used to transport additional information to the smartcard. Thus combining the header and body the ISO 7816-4 standard defines four cases for command APDUs.

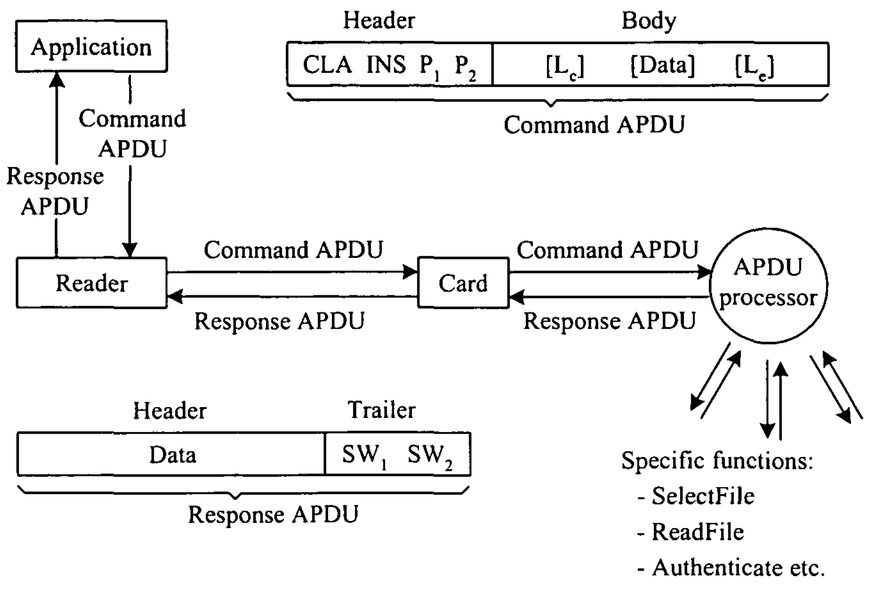


Figure 2.20: APDU structure and application communication architecture [GJ98]

- Response APDU - which consists of a conditional body of variable length and obligatory trailer of two bytes (SW_1, SW_2). The body of the response APDU contains the data returned from the smartcard, if any. The trailer of response APDU, SW_1 and SW_2, indicates the status of the last command. The numbering scheme defined by ISO 7816-4 states that the first byte, SW_1, is used to show the error category and second byte, SW_2, is used to show specific status or error suggestion.

The application, as the top layer, starts the communication with the card through the reader based on these APDU block structures. The software component in the smartcard interprets these APDU command and sends the response back to the higher layers as APDU response, as presented in the Figure 2.20. It is worth pointing out that this standard allows further standardization of additional commands specific for different smartcard manufacturers.

Until 1998 there was no reader independent smartcard API. As stated in [GJ98] this happened for two reasons:

- Card readers were like a link-bridge between smartcard and application. Smartcard manufacturers' primary interest was to explain to users how to use their card readers, but not how to use them with diverse smartcards. A smartcard manufacturer could not know even which smartcards would be used with their reader.
- Most of the smartcard systems were based on particular and non-standard reader interface.

Developers want to build smartcard applications that can be used with different smartcards and readers. System integrators also want to be able to integrate different card reader manufacturers with different smartcard manufacturers in smartcard systems. The most known and widely distributed general purpose smartcard interfaces are the personal computer smartcard (PC/SC) API and the open card framework (OCF) [GJ98].

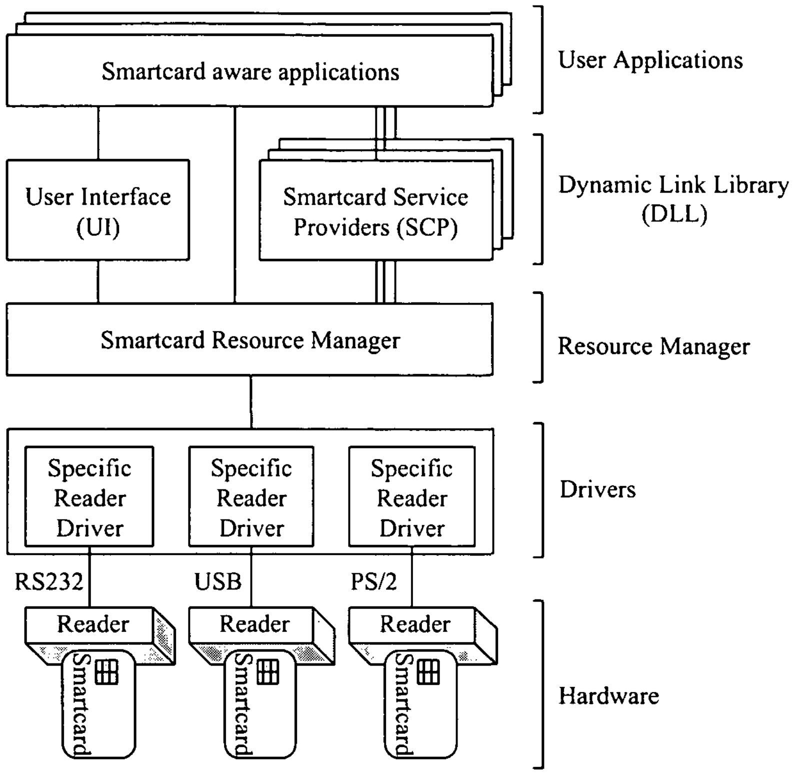


Figure 2.21: The Microsoft PC/SC architecture [MSD01a]

The PC/SC architecture, presented in Figure 2.21, defines a general purpose architecture for including a smartcard in a personal computer [MSD01a]. The specification [Wor97] first published in December of 1996 has clearly defined the parts for:

1. Operating system developers who wish to support smartcards as standard peripheral devices,
2. Original equipment manufacturers (OEMs) interested in developing smartcard readers,
3. OEMs interesting in developing smartcards for use with PCs, and
4. Application developers who wish to create smartcard applications

in order to achieve interoperability and independence between different parties but have the results of their effort work smoothly together.

The PC/SC architecture (see Figure 2.21) defines the interface between smartcard readers and (smartcard) resource manager so that from application point of view all readers behave in the same manner. Thus a smartcard reader is treated by the system just like as floppy disk reader or CD-ROM reader.

Currently, the PC/SC Workgroup Technical Committees are working on PC/SC specification 2.0 [Wor97].

OCF is an object oriented Java software framework for accessing smartcards [IBM98]. OCF's main objectives are [HMGM02]:

- to hide the specific parts of the reader devices, smartcard OS and smartcard issuers,
- to enable smartcard developers rapid application integration and development (RAID) by providing high level APIs , and
- to enable portability of smartcard applications to many different platforms.

The PC/SC and the OCF have the same main goal, supporting the independence and interoperability of different components of the smartcard system. The main goal of PC/SC was to achieve the interoperability of diverse smartcards readers and integrating them into Microsoft Windows OS. The OCF defines a framework that supports smartcards on different platforms, such as different network PCs, smart phones, set-top boxes etc. Further comparisons between PC/SC and OCF can be found in [Sei99, HMGM02].

Recently smartcards in the form of USB dongles are gaining in popularity. The USB dongle is seen in the PC/SC architecture as a reader with an inserted smartcard [Inf01a]. Due to the high speed of the USB interface, these cards have high performance.

2.4.3 Security features

Preventing unauthorized access to sensitive information contained on the card is one the most important security features of the smartcard. The advantage of smartcards over other cards is that smartcards have a processor which manages access to data storage through the use of a "user secret", the PIN. The ISO 9564-1 standard recommends that PIN length should be between the 4 and 12 digits. A PIN of four digits is becoming widespread but the PIN length primary depends on the application's security requirements.

It is in the interests of security not to send the PIN in plain text over a communication line to a computer system for verification, which can easily be captured. Therefore for high security applications, such as e-government and e-payment transactions, card readers with integrated keypad and display, such as Siemens Signator [Sie02], should be used. In this case the PIN is directly passed to the smartcard and not to the terminal or PC where the card reader is connected.

In general smartcard security features are guaranteed through software and hardware security mechanisms of the microprocessor. The security mechanisms of the smartcard must be satisfied at every step of the smartcard life cycle, from the chip manufacturer, card issuer and during usage by the end user [GJ98].

Hardware security aspects of smartcards must prevent the analyze of data and structure of the chip. The Infineon SLE66CX160S chip communicates and stores the data internally in encrypted form [Man98]. The software security aspects make it impossible, through the use of secure messaging, for an attacker to make use of replay attacks and through PIN mechanisms access is gained only to authorized users. The latest smartcard high-end microcontrollers, like Infineon SLE66CX320P support two PIN levels, a so called user PIN and administrator PIN [Inf01a], thus implementing simple PIN management and proving very useful in network administration. The retry counter of the user's PIN [26] is set to 3 and of the administrator counter is set to 7. In the case of three failed entered user PINs, the smartcard is blocked and only the knowledge of administrator PIN can reset the user PIN, thus enabling the smartcard again. But if the administrator PIN is entered wrong 7 times than the card is blocked and becomes useless. It is no longer "smart" it is just a piece of plastic with a "dummy" silicone chip.

Secure messaging, also defined in the ISO 7816-4 Amendment 1 [ISO97b], allows sending together with the APDU its cryptographic check sum. Furthermore the standard specifies that the entire message ($APDU + checksum$) can be encrypted between the card and reader. The cryptographic check sum is also known as a message authentication code (MAC); when it is append to the plain message it assures the receiver that data are has been not changed in transit. In the case where the message is encrypted, the possible attacker gains no information other than APDU header about the data exchanged between the card and reader. This assures the privacy at the lower layer.

There is a recent tendency to replace PIN authentication with biometric authentication, like fingerprint. During the smartcard personalization process the fingerprint is scanned and stored in the smartcard. During the authentication process the user fingerprint is scanned and compared against the fingerprint stored in smartcard. The matching of the fingerprints is done in the smartcard and for these smartcards it is said to have a so called "matcher on card" [Str01]. But it seems that fingerprint protection is not going to succeed as well as was once thought. It has been verified that taking the fingerprint from a glass to foil [Kre01] or simply making a wafer-thin silicon dummy of a fingerprint, as described in [vdPK00], in most cases can match the fingerprint.

2.5 Electronic payment systems

All currently available payment systems have the same main aim of transferring monetary values between parties involved in the transaction. Most of the payment schemes that allow payment across computer networks were proposed in 1980s [OPT97], but they were little of use to those not connected to a network. With the arrival of the Internet this obstacle was removed and new markets have opened.

[26]The user PIN is always set to 3 and after each successfully entered PIN is reset to 3. The possible thief has only 3 possibilities to guess the right PIN. If it is assumed the PIN is four digit and there are no leading zeros the probability to guess the right PIN is: $p = 3/(9999 - 1000) = 0.033\%$.

2.5.1 Introduction and classification

Electronic payment systems are derived from traditional payment models and consequently both systems have much in common. In Web Services the user of a service is called the consumer, so we will continue to use this name also in payment systems. The complete list of all players then would be [Opp00]:

- Consumer – is the user who consumes the Web Service [27] and is paying for used services. In the payment world is called the payer.
- Merchant – is the Web Service provider. In the payment world is known also as payee, merchant or seller.
- Issuer – is a financial institution whose role is to provide the consumer with instances of "virtual" monetary values which are used in payment protocols to transfer the "real money" from consumer to merchant [Fis02].
- Acquirer – is a financial institution used by merchant to transform "virtual money" received from the consumer to "real money".
- Payment gateway – is an optional element in a payment system, in which the consumer and merchant both have registered accounts.

Finally, certain payment systems may involve more players, such as registration authorities (RA), CAs or any other TTPs.

Electronic payment systems based on their connection to the acquirer (bank) can be divided into two models [Has01]:

- Off-line – means that the merchant and consumer are online against each other but not to the acquirer (or payment gateway). Thus the merchant has no possibility to request an authorization from the acquirer, so he is not sure if he can ever get his money. Another disadvantage of this system is that nothing prevents the consumer from spending more money that he has in his account.
- Online – payment systems require a online presence of the acquirer or payment gateway. Through the payment gateway the merchant gets his authorization to charge the consumer. It is obvious the online model requires more communication between involved participants but is more secure than an offline system.

In [Abr01a] all electronic payment systems are divided in two basic groups based on how the money transfer is organized:

- Electronic cash or token-based systems – exchange electronic cash or tokens between participants in a transaction. Electronic cash or tokens are special data structures that represent distinct values, just as banknotes represent the value of paper money.

[27] Indeed the user does not consume the Web Service directly – it is the user's application that consumes the Web Service.

- Credit-debit or account based systems – exchange the money, which is represented as numbers in bank accounts and these numbers are exchanged over computer networks between parties involved in payment transaction.

Electronic money exists is consumer computers and therefore it is very easy to duplicate. Based on its security electronic cash can be divided into systems [AJSW97] either:

- With smartcards – where the electronic cash is stored in the smartcard and is thus impossible to duplicate. The most known representatives are: Mondex [28], Proton [29], Quick [30] etc; or

- Without smartcards – where the digital money exists only in the online environment [Abr01a]. These systems are also called "online cash" or "Web cash". Ecash [31] and Netcash [32] can be included into this category.

Electronic payment systems can also be divided based on their anonymity properties. There are two levels of anonymity [AJSW97]:

- Untraceability – means that the consumer's identity cannot be determined during a transaction, i.e. it is impossible to trace (link) electronic payments to a customer.

- Unlinkability – means that two different electronic payments from the same account (consumer) cannot be linked.

[Opp00] classifies payment systems based on the payment time (see Figure 2.22) into:

- Prepaid payment systems – where a sum of money is withdrawn from the consumer account before any purchasing is made. This money can be used afterwards for payment. Smartcard-based electronic purses and electronic cash falls into this category.

- Pay-now payment systems – where the consumer account is debited at the same time as purchasing. Several debit cards (VISA, MasterCard etc.) belong to this category.

- Postpaid payment systems – the merchant account is credited with the amount of the purchase. The consumer is debited usually on a monthly basis with the purchased amount. Credit card transactions fall into this category.

Based on how the payments are carried out payment systems can be classified into two categories [Abr01a]:

[28]http://www.mondex.com
[29]http://www.proton.be
[30]http://www.quick.at
[31]http://www.digicash.com
[32]http://www.isi.edu/gost/info/netcash/

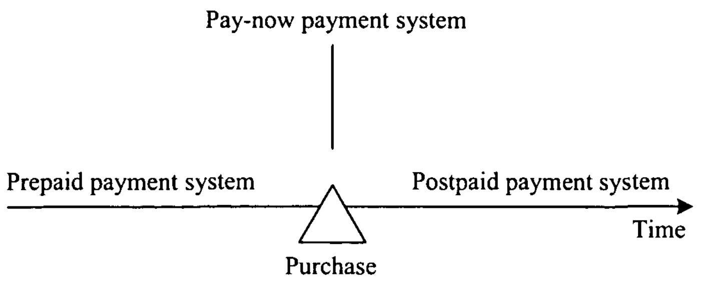


Figure 2.22: Prepaid, pay-now and postpaid payment system [Opp00]

- Payment per transaction – means that the consumer does not have to arrange anything with the merchant in advance, but when he finds something he "must" have, he arranges the payment at once. Prepaid, pay-now and postpaid systems are typical representatives of this category.
- Payment through accounts – means that the consumer and the merchant have set up an account in advance, which the merchant can charge. In this category are: subscriptions, purchase orders and Internet accounts.

Payment system based on the amount of the money exchanged per transaction can be divided into [AJSW97]:

- Macropayment systems – refereed systems which exchange a large amount of money. Credit card systems obviously fall into this group.
- Micropayment systems – those systems that exchange less then 5 euros per transaction [Has01]. Millicent and IBM *i*KP [BGH+95] fall into this category.

Further payment classifications can be found in [Abr01a, AJSW97].

2.5.2 Payment transaction security

Electronic payment systems have different requirements, depending on what they are designed for. It is obvious that systems designed for micropayments will have different security requirements from macropayment systems.

In general, the security in electronic payment transactions must protect [Has01]:

- the consumer, against merchants and diverse outsiders eavesdropping on the communication line, who might monitor and record the transactions and later on duplicate them for their personal interests;
- the merchant, against users who might make more unauthorized copies of their digital money. Since digital money is stored in a consumer's computer hard disk, it can be copied perfectly and in abundance.

From the results of a survey presented in [Abr01b] the security of online payment systems has the highest priority for the 84.7% of respondents.

Payment transaction security can be summarized as [Has01]:

Payment authentication both players must prove their payment identities. Depending on the system requirement their identity must not match their real identity. In case where user anonymity and location untraceability are required a set of security services must be invoked. Consumer anonymity can be achieved by employing pseudonyms instead of real names. In order for the user to stay completely anonymous, the originating host must be not traceable. A simple solution is to route network traffic through a anonymous proxy, so that the network traffic appears to originate from the proxy host. A protocol based on series of anonymizing proxies or *mixes* to achieve user anonymity and location untraceability was proposed in 1981 by David Chaum [Cha81]. This protocol, which is payment system independent also provides protection against traffic analysis. This protocol is based on asymmetric cryptography.

Payment integrity means that the transaction data can not be changed undetected by unauthorized parties. Transaction data, depending on the payment system, consist of consumer, service (product) and merchant information. Payment integrity is achieved through different integrity mechanisms, as described in Section 2.2.

Payment authorization assures the consumer that no money can be taken from his account or his smartcard without his explicit permission. This is very important feature of a payment system in building trust between consumer and merchant. Payment authorization is achieved through: out-band authorization, passwords and signatures [OPT97]. In the out-band authorization case a third party notifies the consumer about the transaction, and he can then decide to accept or reject it. In password-based authorization, a cryptographic checksum is included in every message. This checksum is generated using a secret, e.g. a password, shared only between the verifier (the bank) and the authorizing party. Authorization based on signatures means that the verifying party requires the digital signature of the authorizing party, instead of the cryptographic checksum based on passwords mentioned above. The merchant also must be protected against replay attacks, i.e. if someone is eavesdropping the communication line it should be impossible to replay exchanged messages. This is achieved with use of *nonces* [OPT97], which are simply random numbers and time stamps.

Payment confidentiality must cover more complex cases than just protecting the data from eavesdropping. It most cases it must allow some selected parties to see particular sets of data. Assume the transaction consists of *order information* (OI) and *payment information* (PI), like credit card number and credit card holder name. The consumer would like that OI, relevant only for the merchant, only be readable by the merchant, and that PI, relevant only for the bank, be seen only by the bank, without compromising the integrity of the transaction data [Opp00].

Including security in the payment systems increases the complexity of the system. It is the task of the system architect to maintain the system to be usable, efficient and easy to use.

2.5.3 Secure Electronic Transaction

Secure Electronic Transaction[33] (SET) is a set of protocols designed for secure credit card transactions on the Internet. It was developed by VISA and MasterCard as main developers and also by IBM, Microsoft, VerySign, Netscape, RSA etc. SET was first published on May 31, 1997. In fact, it is an open specification for encryption and security in credit card payments [SET97a].

The Figure 2.23 presents an overview of the SET payment process. The credit card holder (consumer) starts the payment with the merchant using SET. The merchant uses SET to get his authorization from the payment gateway. The payment gateway works as the front end of the financial (bank) network, through which the credit card issuer can be contacted to authorize each transaction explicitly [OPT97]. The financial (private) networks have their own set of specific security protocols usually operating on dedicating lines. SET assumes the existence of such networks, but its specification is based only on data exchange between card holder, merchant and payment gateway [SET97a].

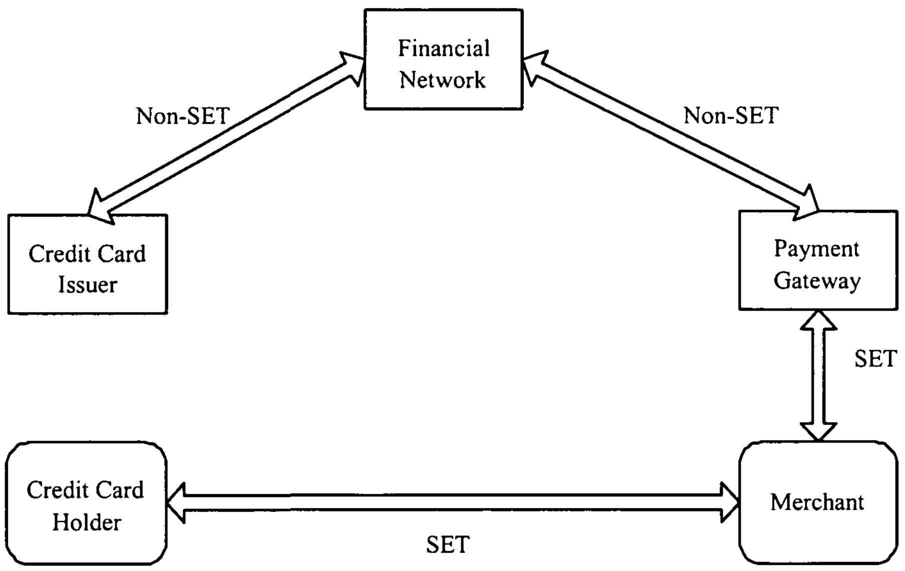


Figure 2.23: Phases of a credit card payment addressed by SET standards [OPT97]

The SET protocol uses cryptography to:

- provide confidentiality of information,
- ensure payment integrity, and
- authenticate both credit card holders and merchants

and was designed from the ground up not to be tied to any specific OS or hardware platform [SET97a]. Its main objective was to achieve global acceptance in the online

[33]http://www.setco.org

market and not change the existing relationship between payer, payee and payment gateway.

Viewing SET from the security point of view results that all participants must have a public key pair and corresponding X.509 certificate. Usually SET participants, merchant and payment gateway have two different public key pairs [Opp00]:

- a public key pair for key exchange, and
- a public key pair for digital signature.

The decision about which key to use in which case is determined through employment of X.509 v3 extension field, *UsageRestriction*, which specifies the usage of a key approved by issuing CA. SET uses 6 private extensions for defining extra values, such as certificate type, merchant data, certificate requirement etc. [SET97b].

The SET certificate management architecture consists of the nine components described in detail in [SET97b]. The architecture is based on the trust hierarchy, defined for the management and verification of SET certificates by certificate authorities (CAs). Figure 2.24 presents the link between the CA and payment participants.

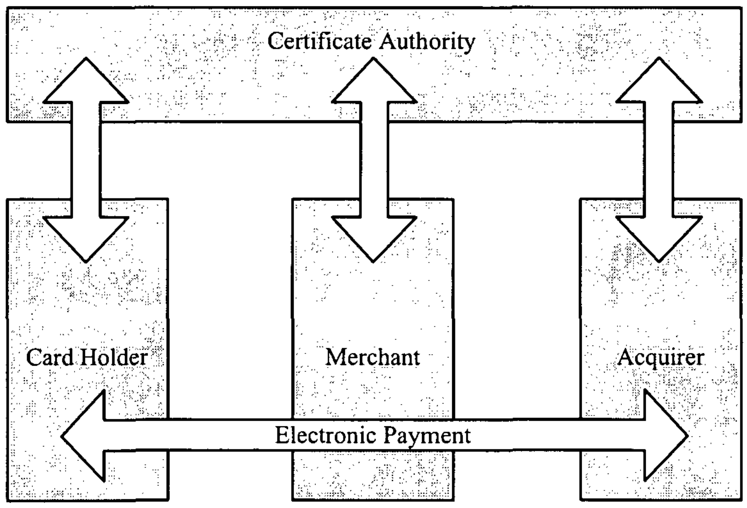


Figure 2.24: Payment system participants [SET97b]

The credit card holder, merchant and payment gateway need to register to CA in order to get their certificates prior to doing any payment transaction. Consequently the use of the SET protocol requires a fully operable X.509 based PKI, including certificate issuing, revocations, CRLs etc.

The SET protocol is a major step regarding user privacy. In SET the merchant understands only the data regarding the goods and services that he sells. The payment gateway understands only that part of information regarding the payment and thus a user privacy is achieved. This separation of information is achieved through a cryptographic mechanism known as a *dual signature*. In SET the credit card holder prepares two sets of information [SET97a]:

- Order information (OI) – identifies the order description at the merchant as is depicted in Figure 2.25. To prevent against replay attacks the OI contains the merchant challenge, time stamps and random numbers. The OI is encrypted with the merchant's public key PK_M.

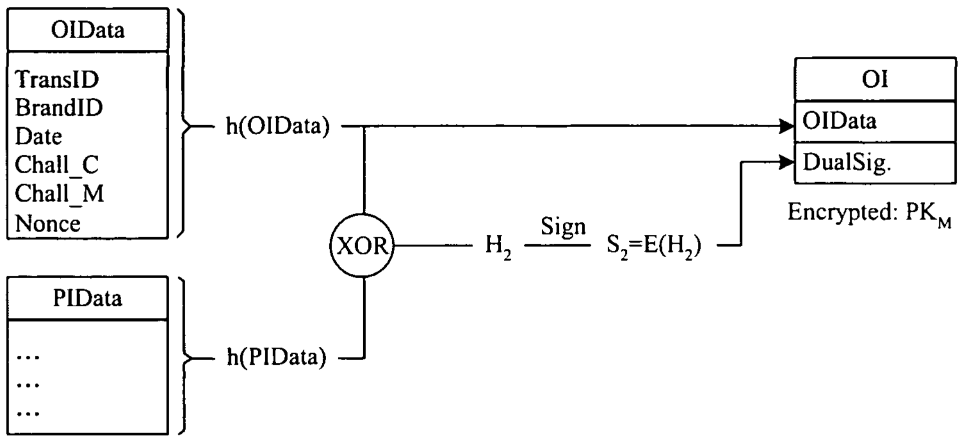


Figure 2.25: Construction of the order information element [OPT97]

A dual signature is created using the hashes of OIData and PIData. The new hash is generated by XOR-ing two hashes (see Figure 2.25) and its value is signed, i.e. a dual signature S_2 is produced. The dual signature is sent to the merchant and payment gateway.

$$H_2 = h(OIData)\ XOR\ h(PIData) \tag{2.10}$$

$$S_2 = E(H_2) \tag{2.11}$$

And thus any entity that possesses OIData or PIData and the S_2 can verify the signature without having to know the other. The payment gateway, to prove the authenticity and integrity of information, would first calculate $h(PIData)$. The payment gateway can calculate this since PIData are encrypted with the public key of the payment gateway. It decrypts the PIData with its private key (see Figure 2.26) and calculates $h(PIData)$. The hash over OIData is obtained from signature like:

$$h(OIData) = H_2\ XOR\ h(PIData) \tag{2.12}$$

And thus has both hash parts: $h(PIData)$ and $h(OIData)$. In order for any instance to verify the signature, the credit card holder certificate is attached to S_2.

- Payment information (PI) – contains card data, purchase amount, order and transaction identifier (Figure 2.26). PIData contains the hash value of the order. It serves the card holder as a unique ID, without giving away what the order is, and serves the acquirer for verifying the message integrity and authenticity. The credit card data contains random numbers to protect against replay and dictionary attacks.

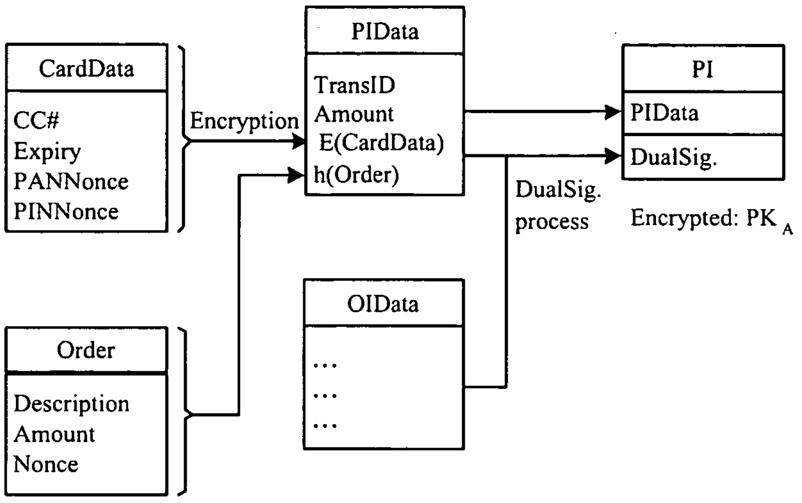


Figure 2.26: Construction of the payment information element [OPT97]

The same dual signature, created for OI, is attached to PI. The PI is encrypted with the acquirer's public key PK_A, so the merchant cannot view its contents.

The SET protocol is payment system independent, but the role of the players is fixed. There must always be one credit card holder, merchant and payment gateway. With SET it is not possible to transfer funds from one consumer to another [Opp00].

[TY98] has made a comparison between SSL and SET regarding payment systems and performance. The SET has received much criticism for its performance. For a large e-commerce server application, support of SET requires an additional hardware acceleration module known as a hardware security module (HSM), which increases the server costs about 5-6% [TY98].

The SET protocol is also used for mobile commerce, a possible approach is presented in [FAKB02].

2.5.4 Ecash

Ecash is a completely anonymous and secure payment system developed by DigiCash to be used over the Internet [Bol99].

DigiCash [34] founder David Chaum is a pioneer in the field of digital cash. The security and privacy of ecash payments are based on both symmetric and asymmetric

[34]DigiCash was founded in 1990. The company initially sold smartcards for closed systems, which was very profitable for years. In 1993 David Chaum invented the ecash system, which made possible anonymous and secure payments over the Internet. The product got great respect among the computer and internet community. Ecash as a product was described by Nicholas Negroponte as "The most exciting product I have seen in the past 20 years". Afterwards DigiCash received many offers to incorporate ecash in:

- MS Windows,

(public key) cryptography. Ecash coins were used on the Interent as monetary values for the first time in October of 1995, when the Mark Twain bank of St. Louis, Missouri, started issuing them [OPT97].

Ecash is based on mechanisms developed by David Chaum to *blind* the connection of the issued coins and the identity of the consumer who originally obtained them [Cha88]. This mechanism, based on RSA signatures, is known as a *blind signature*. A detailed mathematical description of blind signatures can be found in [OPT97]. The ecash model is presented in Figure 2.27 and consists of client (consumer), merchant and ecash bank.

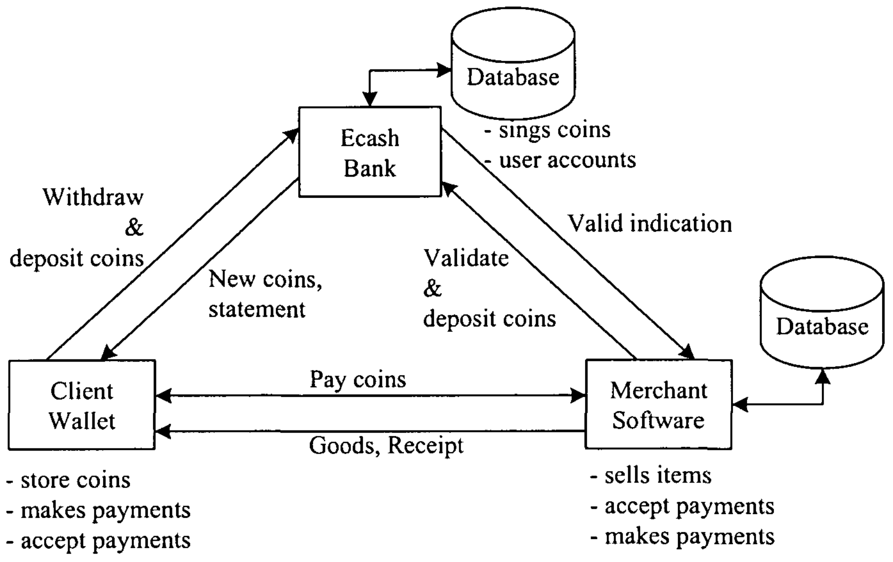


Figure 2.27: Entities and their functions within ecash system

The ecash coins have a serial number, usually a 100 digit random number, that is not seen (blinded) by the bank but is signed by it. The bank signs coins without knowing their serial numbers and thus cannot link coins to a particular user. A merchant who receives the coin must before delivering goods to the user, make sure that the coin is not spent before. If the coin is valid it will be stored in the merchant's account and marked as spent by the bank. A merchant can make payments to clients using same the methods. It is obvious that the client and merchant must have accounts at the same bank or there must exist a system that enables the interchange of coins between banks. Further details of the flow can be found in [OPT97].

Since digital coins reside in client PCs, they can be copied perfectly by dishonest users in order to spent them many times. This problem, also known as the double spending problem, is avoided with a central database from the ecash bank (see Figure 2.27).

- Netscape,
- VISA etc.

but its manager David Chaum always wanted more money from them. Bill Gates, CEO of Microsoft offered him 100 million US $ to incorporate ecash into the Windows95 operating system, but he refused to sell it for less than 1 or 2 US $ per sold copy. In September 1998 the DigiCash company went bankrupt. Obviously being a good scientist and a good manager are two different things. Anyway he was far ahead of his time – too far [Gri99].

Each digital coin spent is flagged as spent and therefore by the next query is triggered as an invalid coin. Details regarding the double spending problem, stealing and exchanging coins can be found in [Has01].

2.5.5 Microsoft Passport

Passport [35] is an online authentication system managed by the Microsoft Corporation. After a user signs on to the Passport service other sites that the user visits get the information about the user identity. Passport uses a central database for storing user identity information, which is also maintained by Microsoft. Microsoft states that it manages over 200 million Passport accounts and performs more than 3.5 billion authentications each month [EPI02, Mic03a].

Systems where authentication can be provided for multiple services and users need to remember only one username and password are said to be single sign-on enabled. Kerberos, one the most known network authentication protocols, is an example of a system where users provide a password and receive a ticket in exchange [MIT03]. The received ticket can be used to authenticate users against different network services. Kerberos uses a centralized database containing keys that are shared with each used service [KN93]. Due to different implementations across web sites, single sign-on on the web is more difficult then in the Kerberos protocol [KR00b]. Passport is Microsoft's approach to provide single sign-on on the web.

Figure 2.28 presents a standard flow between user's browser, merchant (IBM.com) and Passport server. When the user first need to be authenticated on the Internet, he will be redirected to the Passport server, steps 2 and 3 in Figure 2.28.

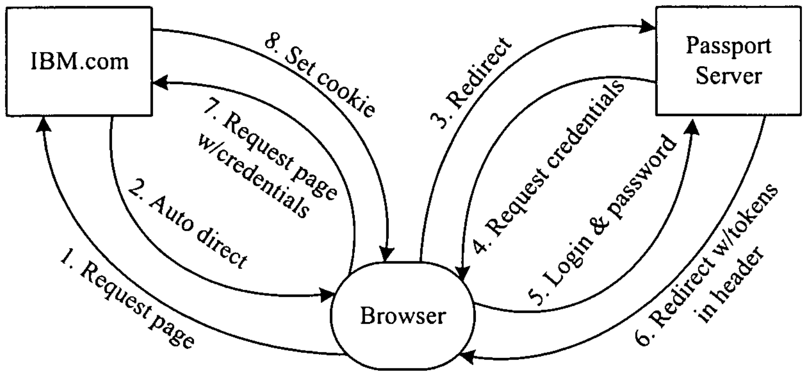


Figure 2.28: The Passport architecture [KR00b]

The login to the MS Passport server is done over a SSL/TLS connection. After successful login the Passport server redirects the user back to the merchant server. The redirected message is encrypted using a triple DES key which was established beforehand between Passport and the merchant server. The merchant server sets this as an encrypted cookie in the client's browser [KR00b].

A cookie is a text-only string used by a Internet browser and is set by a web site. If the lifetime of the cookie is set to be longer than the time user spends at that site,

[35]http://www.passport.com

then this string is saved to a file for future reference, thus enabling user personalized sites [Wha02].

Using cookies enables the merchant web server to verify in any time if the user is already authenticated and if his credentials are valid. During the authentication process the Passport server also sets a cookie. Thus, if a user visits another site, for example siemens.com, when the browser is redirected to the Passport server, the user is not enforced to login because the previous cookie is used. If this cookie contains valid credentials, the client's browser is redirected back to the Siemens server without user intervention [KR00b].

With the introduction of the .NET (dot New Enterprise Technology) platform by Microsoft in early 2001, Microsoft changed the name of the MS Passport to MS .NET Passport.

The .NET Passport consists of the:

- Passport single sign-in service,
- Passport express purchase, and
- Kids' Passport service.

Passport express purchase allows consumers to establish a ".NET Passport Wallet", where their billing, shipping, and credit card numbers can be stored [Pro01]. Merchants supporting this service allow consumers a convenient payment method, since consumers do not have to enter their address and payment information manually for each web site.

In [Sle01] MS Passport is criticized for its cookie-based technology, which makes MS Passport not as secure as Microsoft claims. Furthermore in [Sle01] an example presented where a malicious user can extract MS Passport data including the wallet information. Another security hole was discovered in MS Passport in April 2003, where a malicious user could redirect a so called "reset password" to any desired account, thus gaining access to any MS Passport account [AM03].

The Kids' Passport service allows parents to modify their child's profile and set certain restrictions and conditions. The Kids Passport service also allows visiting sites to only collect certain personal information, which are allowed by their parents and conform the US Children's Online Privacy Protection Act (COPPA) [EPI02, Mic03a].

Microsoft has stopped the .NET Passport express purchase service as of March 2003 [NN03c].

2.6 Web Service security efforts

Security plays an important role in Web Services. Competing organizations are working together in order to achieve a higher degree of security and wider acceptance of Web Services in the market. It is the first time that major vendors like Microsoft, IBM, Sun and others are making their interfaces open and free of charge for public use [Kno02]. Security efforts for Web Services started from the day they were released. Web Service vendors knew that wide and general acceptance depends on security, but in any case "Web Services are coming and they are going to be as common as TCP today" [Cha02].

When the SOAP protocol was developed security was not in the foreground. The key goal of SOAP then was to utilize existing standards and Internet architecture instead of reinventing new protocols. After its first public release SOAP has received much criticism for its lack of security [JZ02, Ras02, HNN02].

2.6.1 Is SSL/TLS not secure enough?

Web Services use SOAP, and SOAP uses the XML format to request and receive data over the Internet. Since the XML format is plain text data it means that data over the Internet is travelling in plain text mode.

There are opinions that SOAP's job is not to provide security [Vas01]. This statement is not unreasonable, since SOAP is high-level message protocol it is assumed that underlying transport protocols will take care of security.

Security in a system can be introduced in different layers, as presented in Figure 2.29. The lowest level of security will be implementing security in the IP layer. This is known as IP Security (IPSec), and any service or application using IP layer communicates securely with the outside world. IPSec is an extension of the existing IP protocol designed to protect IP packets from snooping or modification as well as providing a defense against network attacks [Sch00].

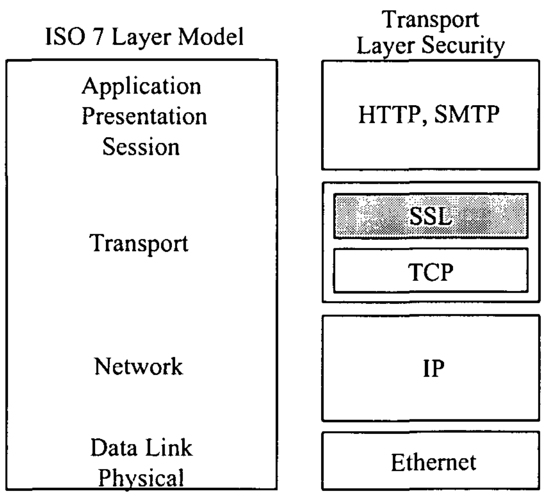


Figure 2.29: Transport security [Fer02]

SSL/TLS [36] (Secure Socket Layer/Transport Layer Security) is a standard for securing web connections in a point-to-point manner. It secures the entire channel from client to server at the transport level and all applications using this channel can securely communicate with each other. An SSL enabled browser such as Internet

[36]TLS is point-to-point protocol that follows the Security Socket Layer (SSL) designed by Netscape and from version 3.0 was adopted from Internet Engineering Task Force (IETF) [FKK96].

Explorer or Mozilla, uses encryption to send encrypted data from browser to web server. In [CHVV03] a security hole in SSL/TLS is presented, the so called "timing attack", whereby one could intercept the password in an SSL/TLS channel. To eliminate the "timing attack" problem a set of measures were proposed to update the SSL/TLS protocol [CHVV03].

Application security has an (dis-)advantage compared with low level security, because security is implemented in an application that wants to communicates securely with other services or applications, as presented in Figure 2.29.

Employing SOAP over SSL/TLS would be secure [MMVD02]. This would be sufficient in the case where a consumer communicates with only one Web Service, which in turn does not need to forward user credentials to any third Web Service. Figure 2.30 presents a case where user wants to communicate with Web Service A, then with B and D. The respond should be routed through Web Service C, i.e. a different route from the request (D–C–A).

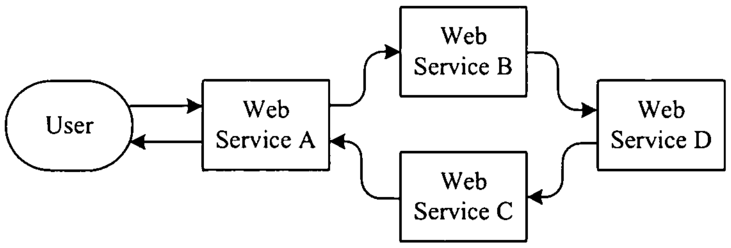


Figure 2.30: Invoking Web Services with intermediaries

SOAP over SSL/TLS has no mechanism for providing authorization, auditing, non-repudiation and features like identification, authentication, integrity and confidentiality stops at SSL/TLS endpoint [Col03a].

If the user requires that messages are encrypted with different keys so that each Web Service understands only its specific part, and transmitted in specified routes, this is impossible with SSL. As explained above SSL can secure only point-to-point connections. the user can make a SSL connection to Web Service A, or B or D but not to all Web Services in the path (A–B–D).

Web Service Enhancements (WSE) provides features like scalable encrypting and routing to address the problem presented in Figure 2.30.

2.6.2 Web Service Security

This proposal was a joint effort by Microsoft, IBM and VeriSign [37] and was published on April 2002 [IM02b, ADLH+02]. The specification suggests a group of SOAP extensions, that can be used to build secure Web Services.

The main goal of WS-Security is to associate a security token to message content. It describes also how to encode security tokens such as X.509 certificates and Kerberos tickets [Aps02]. "WS Security is flexible and is designed to be used as the basis for the construction of a wide variety of security models including PKI, Kerberos and

[37]http://www.verisign.com

SSL" [ADLH+02]. With the WS Security specifications applications can build secure SOAP extensions.

The WS Security protocol supports [IM02a]:

- multiple security tokens for authentications or authorizations,
- multiple trust domains,
- multiple encryption technologies, and
- end-to-end message-level security.

Microsoft and IBM agreed in July 2002 to give the WS Security specification to OASIS [38] without any licence fees for further standardization [Kno02].

Web Services are challenging the IT community due to their distributed and platform independent nature. Web Services are able to integrate different applications and operating system technologies and made them to communicate with each other over Internet. This interaction between different application across domains is based on standard protocols like: HTTP, XML, SOAP, WSDL, and UDDI. In many cases Web Services are implement in different tiers (intermediaries) inside a domain [Pro02], as presented in Figure 2.31.

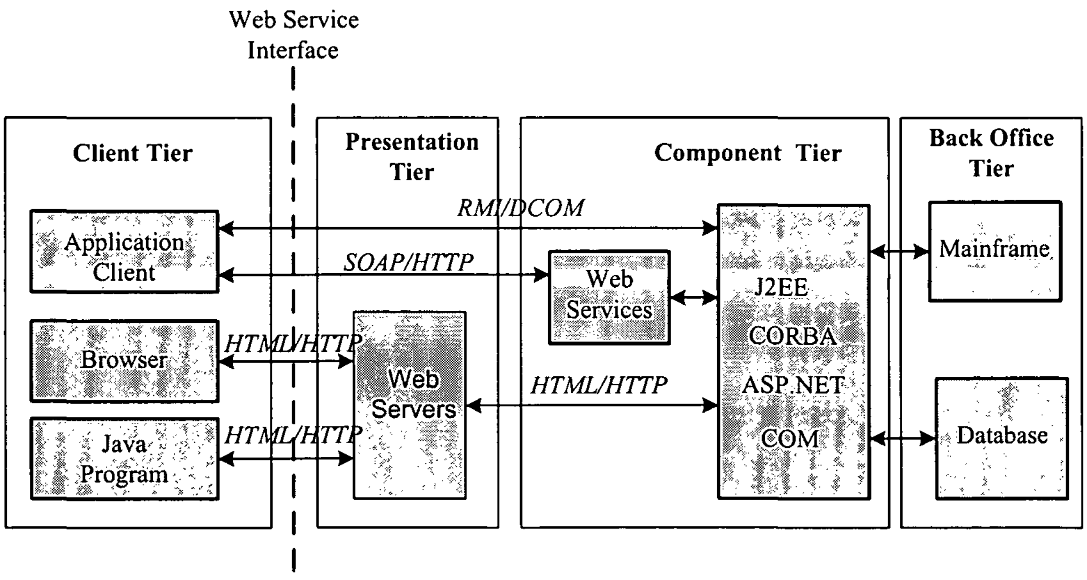


Figure 2.31: Typical Web Service environment [Pro02]

Inside corporate domains the principals (user or system components) are registered (in a central database) and after authentication all operations are identified with authenticated principal profile. Microsoft Active Directory (AD) is an excellent place for storing such profiles [MMVD02]. Web Services expect users and components to come from the Internet, unknown beforehand. The question arises of what kind of security criteria users and application coming from the Internet must fulfill . In the

[38] http://www.oasis-open.org

case when Web Service invokes another Web Services on the behalf of the client a question arises: should the middle tier pass its credentials (this process is known as delegation) or those from the client (see Figure 2.31). In such cases, the commonly used authentication schemes are [MMVD02]:

- authentication with impersonation,
- authentication without impersonation, and
- authentication using fixed identity.

The main features provided by a WS Security implementation is the capability to encrypt, decrypt and digitally sign partial SOAP messages [Evj03]. This is achieved through XML encryption and signatures.

XML encryption – SSL/TLS is a secure and reliable point-to-point protocol and XML encryption is not intended to replace the SSL/TLS protocol [Sid02a].

With XML encryption it is possible to exchange not only primitive data types but also complex data structures. XML encryption provides a mechanism for security requirements that are not covered by SSL/TLS such as [Sid02a]:

- encrypting only specific parts of the XML message, and
- secure session between more than two parties.

With XML encryption, the document could be partially encrypted and in the same document may be exchanged the plain and encrypted data [Ent01].

The W3C Encryption Working Group is working toward a standardization of XML encryption. In March 2002, the Working Group published a XML Encryption Requirements W3C Note. In December 2002, XML Encryption Syntax and Processing was published as Proposed Recommendation by W3C Working Group [IDS02].

XML signature – A digital signature provides integrity and a non-repudiation of a message [FB01]. The main feature of XML signature is the ability to sign only specific parts of XML document [BBF$^+$02].

There are three possibilities for the location of the XML signature relative to XML source [Ent01]:

- enveloped, the XML signature being inserted in XML source,
- enveloping, XML signature wrapping the XML source, and
- external, XML signature located in a separate XML document.

XML syntax allows different representations for objects through *element* and *attribute* values [Sch03c]. For the entity that it is checking the message integrity is important to have unique *elements* and *attributes*, otherwise the verification will fail. *Canonical XML* addresses this problem [Boy01]. During signature generation, the hash value is computed over canonical form of the XML document [San03].

XML digital signatures can be added to the existing XML applications, whereby a proxy can be inserted between application and Internet to examine and sign XML context as presented in the work of [TU02].

The Digital Signatures working group at W3C is a joint effort of W3C and IETF and was formed in 1999. The XML Signature Syntax and Processing specification was published as a W3C Recommendation in February 2002 [BBF+02].

There is also one very important requirement for security in general: legislative support. Figure 2.32 presents the correlation of security frameworks. In most European Countries a digital signature now has the same weight as a handwritten signature and in other countries legislation framework are in progress [NN03a].

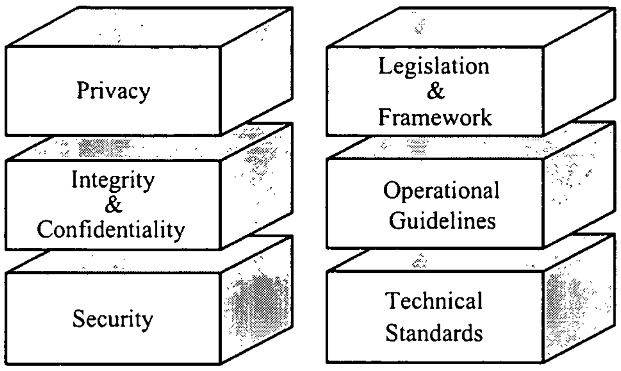


Figure 2.32: Matching concepts and standards

The European Union developed a privacy directive [Com00b], which came into effect in October 1998. Each European Union (EU) country has to fulfill privacy laws that are in accordance with the requirements of the directive. Furthermore the directive demands establishing a privacy agency with execution capabilities. The directive forbids exchanging consumer related data with companies in countries that do not have adequate privacy laws [Gol00].

2.6.3 Web Service Routing Protocol

The Web Services Routing Protocol (WS Routing) is a stateless SOAP protocol that enables exchanging SOAP messages between an initial sender and the final receiver [NT01]. This message exchange is usually done via a group of intermediaries, as is presented in Figure 2.33.

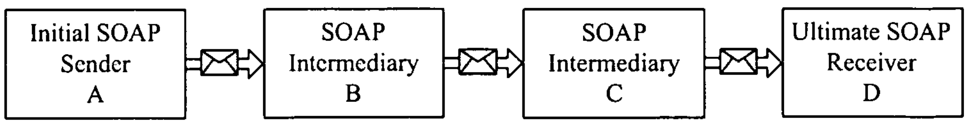


Figure 2.33: SOAP Processor A sending a SOAP message to D via B then C [NT01]

Routing can be implemented in hardware or software using a variety of different algorithms and networking protocols. Routing algorithms can perform anything from network address translation (NAT) to more advanced content analysis, where routing decisions are based on what is in the message (also known as content-based routing) [Com00a].

Using WS Routing it is possible to define different paths for forward and reserve SOAP messages. WS Routing supports [Sko03]:

- the SOAP message path model,
- full-duplex, one-way message patterns,
- full-duplex, request-response message patterns, and
- message correlation.

WS Routing's objective is not to define protocols for secure and reliable SOAP messaging. These features are available through additional SOAP extensions.

Any SOAP message that implement WS Routing inserts an extra command, named *path*, in SOAP header. The path command contains these elements [Evj03]:

- a "*from*" element for the message originator (A),
- a "*to*" element for the final receiver (D),
- a "*fwd*" element to contain the forward message path, and
- a "*rev*" element to contain the reverse message path.

The *rev* element enables bidirectional SOAP message exchange between participants. The elements *fwd* and *rev* contain "*via*" element that describes the intermediaries (B and C in Figure 2.33) [Sko03].

The Web Service Referral specification makes it possible to change the routing path of SOAP messages dynamically [NCL01].

The main advantages of WS Routing are: the ability to make transparent the physical network topology to the client (i.e. hiding the intern network infrastructure) and load balancing [Sko03].

2.6.4 Web Service Attachment and Direct Internet Message Encapsulation

In a SOAP message it is possible to transmit different attachments such as images, word and pdf documents etc. just by inserting them into the SOAP envelope. Since these attachments are not in text format, they need to be converted (serialized) first in a text format (such as Base64). The disadvantage of this approach is the cost of processing time at sender and receiver [Evj03].

Web Service Attachment and Direct Internet Message Encapsulation (DIME) specification answers the problem above. WS Attachment and DIME allow attaching documents outside the SOAP envelope instead of having to serialize and insert them into the SOAP body [BTN00].

DIME allows encapsulation of binary data in a series of records, as is presented in Figure 2.34. *MB* and *ME* in Figure 2.34 denote message begin and message end respectively. A DIME message has a binary format with 32 bit length fields. Each DIME record consists of DIME header and body [Evj03].

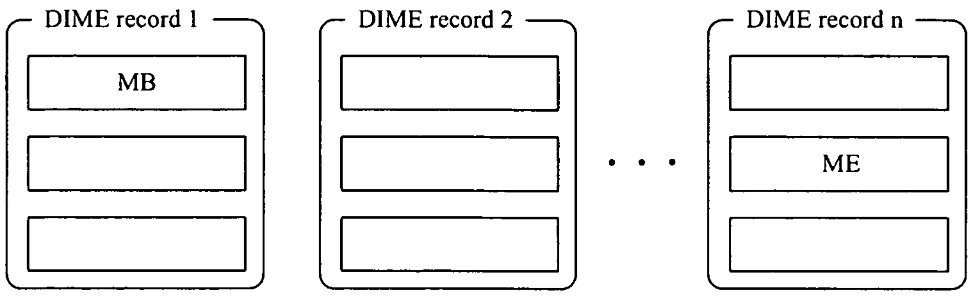


Figure 2.34: DIME records [Evj03]

Using the DIME packaging mechanism it is possible to put together many records of arbitrarily formatted data [Pow02].

WS Attachment defines how DIME packaging can be used to insert different attachments into SOAP messages [NCF02]. The SOAP message attachment mechanism posses the capability to identify the attachment through use of the *href* attribute. The *href* attribute is the pointer to an attachment in the form of a Unique Resource Identifier (URI) [Pow02].

2.6.5 Web Service Coordination and Transaction

Web Service Coordination (WS Coordination) is an extensible framework that provides protocols for coordinating the activities of distributed applications. Coordination protocols are used to support those applications that need to maintain a stable and consistent result over distributed transactions [CCF+02].

The coordination service consists of these services [CCF+02]:

- activation service – which enables the application to create a coordination instance,
- registration service – which enables the application to register for coordination protocols, and
- coordination protocols – which allow selecting the preferred protocol for a selected service.

A transaction is a group of (usually database) operations combined into a logical unit of work that is either wholly committed or, in a failed case, rolled back [Mic02b].

Web Services are stateless by the definition (see Section 2.1) and this in many cases is considered a missing feature [Evj03]. The Web Service Transaction (WS Transaction) is meant to address this feature of Web Services. WS Transaction uses WS Coordination to a define Web Service as a transaction process. The WS Transaction specification provides the definition of two coordination types [CCC+02]:

- Atomic transaction (AT) – is a coordination type used to coordinate actions that are executed in not fully trusted domains and usually have a short execution time. This coordination type has an "all or nothing" property, therefore it is called atomic transaction.
- Business activity (BA) – is complementary to AT and is meant for long-lived transactions that can not have locks on data or physical resources.

WS Coordination and WS Transaction alone are not secure. They have to implement WS Security in order to fulfill authentication, confidentiality, integrity and non-repudiation.

Chapter 3

Web Services Market

"Web services represent a tectonic shift in the software industry; this is the next wave of computing and we are at the beginning of a massive 10-year build-out" [KG02].

There are many companies recognizing the arising market for Web Services and reorganizing their services to adapt to Web Service interfaces. There are many vendors supporting Web Services, including, among others:

- operating system vendors,
- development environment vendors, and
- security vendors.

"The Web Services will be as common as TCP, architects and developers will use them routinely" [Cha02].

XML Web Services are changing the security policies of enterprises. Traditional distributed computing was modeled by closed and isolated user groups. User coming from an intranet are considered to be trusted and users from the Internet are not. This grouping of users as "trusted" vs. "not trusted" in service models is breaking down, because through Web Services untrusted users are gaining access to all enterprise resources. Enterprises must change their security policies to accept untrusted users. There is a market need for Web Services and XML traffic security tools [BS02b].

3.1 Future of Web Services

Web Services still have a long way to go before developers can use them to create standard, secure and robust enterprise applications. Currently, most of the Web Services available for public access are relatively simple, but this may change quickly [Ort02].

The evaluation of Web Services is divided into three phases [Wil01]:

Phase one (2001-2002) – In this phase the standards are written. The technology in this phase enables organizations to improve Enterprise Application Integration (EAI) and achieve B2B integration between trusted partners. In this phase simple Web Services start to appear for public use.

Phase two (2002-2004) – this phase is already underway. This is the improvement of Web Service infrastructure. The main goals of this phase are increasing the security and quality of Web Services, as is presented in Figure 3.1.

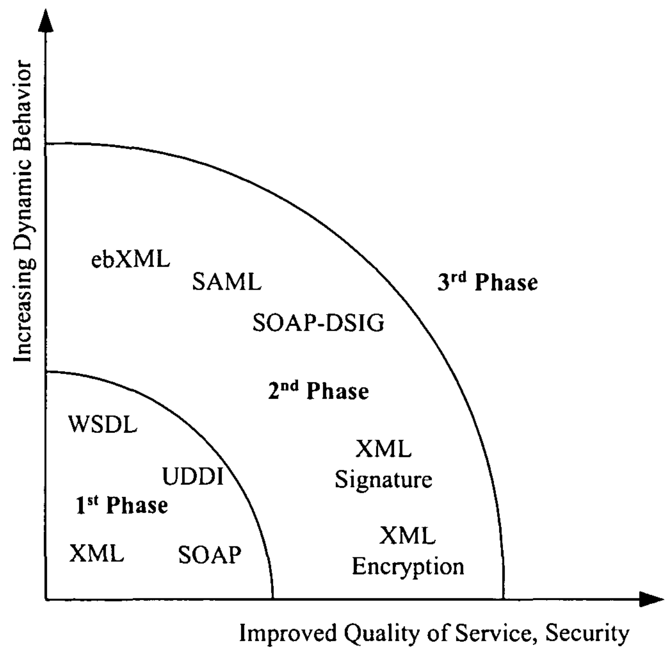


Figure 3.1: Future of Web Services [Wil01]

"If the Web Services are to be realized based on this graphic (Figure 3.1), then business standards must be agreed at an industry level" [Ort02].

Phase three (2004 +) – It is assumed that once enterprises have switched to use the benefits of Web Services they will also change their business models to take full advantage of the dynamic behavior of Web Services.

A similar forecast for Web Services can be found in [Ste02].

3.2 Single sign-on and PKI vendors

Single sign-on (SSO) describes a mechanism whereby a user provides his credentials only once (like username and password, X.509 certificate etc.) to access different services in different and distributed systems (a similar technology and approach to the Microsoft Passport, presented in Section 2.5.5) [Ope98]. The mechanism has been proved very useful in enterprise applications giving employees access to different internal and external applications and resources [Kar02].

SSO supports many features like [Wed03b, Wed03a]:

- hiding URL in which the user does not have access,
- smartcard support,

- encryption,
- digital IDs,
- X.509 certificates, and
- cross-domain authentication etc.

Using Web Services to access other Web Services, as is presented in Figure 3.2, the question arises of what credentials are passed to intermediate Web Services, or whether the user should be asked to provide appropriate credentials for each accessed Web Service. Single sign-on enables Web Service clients to authenticate only once and use these credentials for further authentication. Single sign-on solution has been found to improve user productivity and decrease management costs [Wed03b].

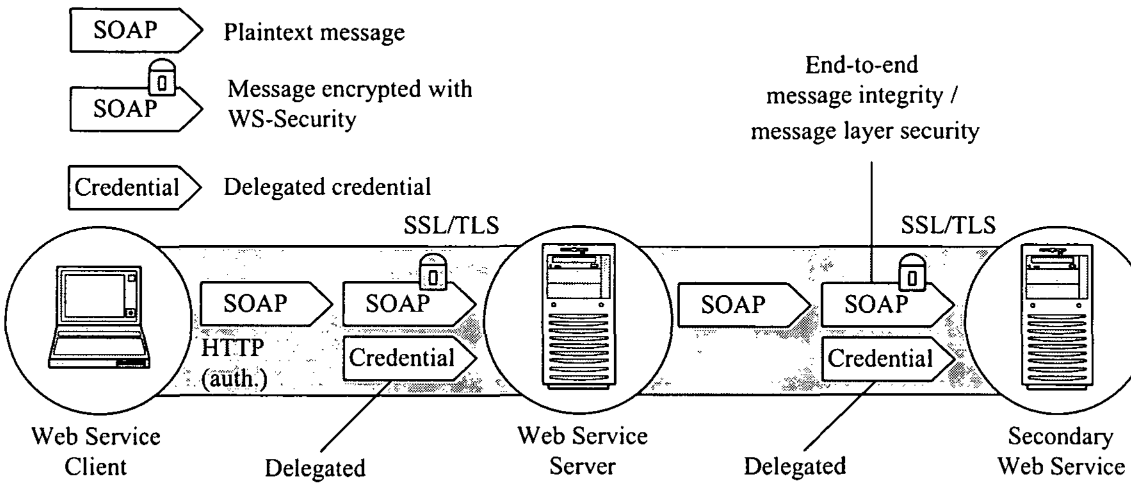


Figure 3.2: Web Service delegation [Wed03b]

An SSO solution is proposed in [Wed03b] for accessing Web Services using Java 2 Enterprise Edition (J2EE), .NET and Active Directory (AD). This supports delegation, as presented in Figure 3.2, and end-to-end security. An SSO with smartcard integration using J2EE is presented in [Wed03a].

There is one disadvantage to using SSO. If one link in the chain is compromised, then the entire chain is compromised.

PKI technology is established today as the state of the art in information security in the Internet. Vendors offering their PKI services are hurrying to incorporate PKI inside Web Services. The biggest world wide PKI vendors such as:

- RSA Security,
- Certicom [1],
- Entrust [2],

[1] http://www.certicom.com
[2] http://www.entrust.com

- Baltimore Technologies [3], and
- VeriSign [4]

are already in technical committees for WS security and OASIS and are observing the market changes for Web Services [OAS03]. Another question that arises in this context is: where to store the private key issued from PKI vendors? One possibility is in smartcards. Smartcards have been established as the best places to store valuable information such as private keys, public certificates etc. Smartcards are also been seen to be fault tolerant on different attacks [SS99, RE99].

3.3 Web Services - new challenges

Web Services imply a new challenge in the market. The growth of XML traffic on networks requires new XML solution for enterprises. Current firewalls, router, switches and proxies are not usable for securing Web Services since they are not content aware. All these protocols operate at the link or network layer of the OSI model (see Figure 2.1). Standard firewalls block the network traffic except the web traffic on so called known ports such as HTTP on port 80 and HTTPS on port 443, and email traffic on port 25 [Com00a].

Standard firewalls can operate either in [Yeo03]:

- static mode – where the data packets are denied or forwarded based on the network address of the packet, or
- dynamic mode – more resource expensive, where the filtering is based on the contents of data packets.

However, the TCP/IP model is too simple for dealing with XML and SOAP messages. Therefore the firewall, instead of being IP-aware, must be content aware, even more precisely it must be XML aware. It must fully understand the XML and SOAP traffic that flows through network [BS02b].

A SOAP-level firewall should be able to [Vor03]:

- identify whether the incoming SOAP request is targeted at an available Web Service, and
- identify whether the SOAP message has valid content.

This is similar to the dynamic mode at the network layer, where IP packet contents are inspected. SOAP firewalls implemented in the application layer require the knowledge of what data the Web Service expects. In this way, simple threats such as passing wrong or unexpected parameters to Web Services may disable them for further usage. But until such firewalls are not applied, the Web Services present a new form of threat for gaining access to private enterprise resources [Vor03].

[3]http://www.baltimore.com
[4]http://www.verisign.com

Another requirement that SOAP firewall rules must fulfill is always to stay synchronized with Web Services. UDDI and WSDL should be used for this purpose. UDDI, as described in Section 2.1.4, is used to return a list of deployed services. A dynamic SOAP firewall queries the UDDI registry for receiving the recent Web Service interface description, thus ensuring that only allowed (correctly formatted) traffic is passed through firewall [ONe03].

3.4 Current and future state of the market

Web Services are able to connect different existing applications through the Internet with application servers. They represent a new model for integrating enterprise applications and services. Web Services enhances the communication model that enterprises have with their partners and customers [SR03].

Web Services are opening a new market for application and service integration functionality inside and among enterprises. They promise to automate existing business processes and relationships. Interoperability with different devices and vendors is a key issue for the success and wide acceptance of Web Services. Web Services will enable exchange of application data with handheld devices, which was an old dream of field personnel working directly in the field or with the customer [Win02b].

Figure 3.3 presents a market forecast for Web Services for the period 2002-2007 by Winter Green Research [5]. Web Services markets are expected to be \$152 million in 2002 and reach \$3.1 billion by 2007 [Win02b].

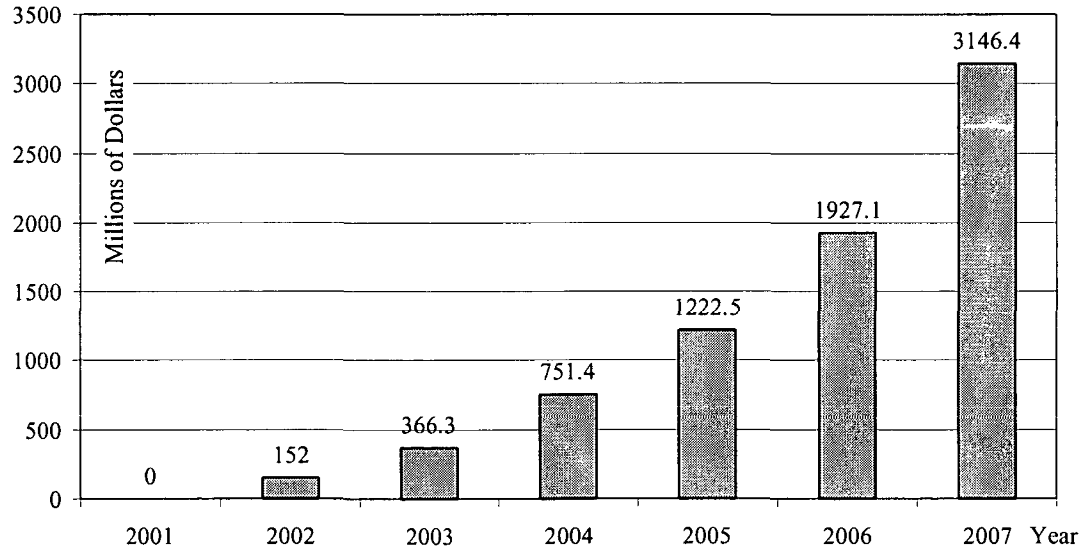


Figure 3.3: Web Services market 2002-2007 [Win02b]

Another forecast of the Web Service market is to be found in [KG02], stating that "the Web Services software market will reach US \$1.7 billion in 2003".

[5]http://www.wintergreenresearch.com

Even these two forecasts, [Win02b] and [KG02], predict different values regarding the Web Service market, if met would ensure the profitability of many enterprises involved in Web Services.

A report from *Gartner* [6] *research group* states that the advancement of Web Services is made weak by a slow economy, but that projects based on such services are continuing to grow. Furthermore the report states that only 1% of survey participants claimed that they stopped all Web Service projects and 6% postponed their Web Services projects [Bou03].

[6]http://www.gartner.com

Chapter 4

Privacy Violation

Chapter 4 describes privacy violation in online transactions in detail. Physical and online store purchase are described and compared with respect to violations in user privacy.

In everyday life we hear about theft of credit card numbers from merchant companies [Wil02], and the amount of credit card number theft is increasing rapidly [Blo03, CW02]. The question which arises here is why the merchants have to keep a database of customers' credit card numbers. Further, it happens that a user buys some goods online at merchant 1 and after two or three weeks he gets advertising from competitive merchants about the same goods that he bought. Due to the theft of customer personal information at merchant 1, the user gets advertising from all over the world. So another question arises: why the merchant has to keep a database of customer habits. This is a privacy violation.

"Information privacy is the claim of individuals, groups, or institutions to determine for themselves when, how and to what extent information about them is communicated to others" [Bra00]. Privacy is "freedom from unauthorized intrusion" [GS01].

4.1 Physical store purchase

This section describes a physical transaction in a real store based on [Kin99, Ver02b].

A customer enters the store. After looking around for the articles he needs, he selects the items and puts them in a shopping cart. After collecting all necessary items the customer proceeds to the checkout counter for payment.

The customer takes out his credit card and gives it to the checkout clerk. The clerk puts the credit card in the machine and waits for an authorization code from the credit card issuer to charge customer's credit card with the amount.

The credit card issuer (like VISA, America Express, etc.) compares the credit card data and purchase information with cardholder's expiration date, credit limit and current balance. Based on this comparison, the credit card issuer sends back the authorization code to allow or refuse the payment transaction.

Assuming the authorization was acknowledged, the merchant's machine prints a charge slip and customers signs it (*non-repudiation*). The clerk compares the signature on the credit card with the signature on the charge slip (*face to face authentication*). A customer takes a copy of his signed charge slip and departs from the store with his new goods.

In case that the authorization was acknowledged, the merchant's credit card machine automatically generates another transaction based on the authorization number issued by the credit card issuer. In this transaction the merchant receives the customer's purchase amount from the credit card issuer.

4.2 Current online store purchase

The number of computers connected to Internet is growing daily as shown by a report from United Nations [Ron02](see Table 4.1).

Online	August '00	August '01	% Change (+/-)
Africa	3.2 mil.	4.2 mil.	+24
Asia/Pacific	105 mil.	143 mil.	+28
Europe	114 mil.	154 mil.	+26
Middle East	2.5 mil.	4.7 mil.	+47
Canada/USA	168 mil.	181 mil.	+8
Latin America	17 mil.	25 mil.	+32
World Total	408 mil.	514 mil.	+20

Table 4.1: Global online population [Ron02]

The report also estimates that the growth in the world's online population of about 20% will hold for the next several years [Ron02].

A report from [Ver02b] in the USA states that over 60% of US households are online, and more than half of these households shop from home on a weekly basis, spending an average of about $500 for online shopping in the year 2002.

Consumers in Europe are beginning to rival US in their Internet spending habits, states a report from [NN03b]. The report states that Europeans spent on average 430 Euro online between August and October 2002 and US consumers spend an average of 543 Euro over the same period. Furthermore, the estimates shows that 41% of US online consumers hope to buy more products online in the next years, compared to 45% European online consumers [NN03b].

Companies that are not online are missing a very important income opportunity. Offering an online purchase possibility to customers is not treated any more as "extra feature" but as a critical step for managing a successful business. The number of online stores have increased permanently and most of them have adopted credit card payment technology, which has a successful history and great market acceptance [Ver02b].

The most common form of online payment today is to send the credit card number in an SSL/TLS encrypted channel from the consumer's browser to merchant's web server. This method has been accepted by merchants for two reasons: it requires minimal extra effort to receive payments over SSL/TLS, and other online payment methods have been seen as too complex and expensive [AJSW97, FK00].

In [Kin99, Ver02b] a standard online purchase at high level is described as follows.

A customer enters the online store by visiting the merchant's web site, as is presented in Figure 4.1. Like at a physical store, the customer selects items and puts them into a virtual shopping cart.

The customer starts a payment transaction usually by clicking a link like "Buy now". In the new form all the items and costs are summarized, and the customer is asked about his credit card and shipping information. The user fills the form, which is typically secured with SSL, and by clicking the submit button, the form is sent to the merchant's web server.

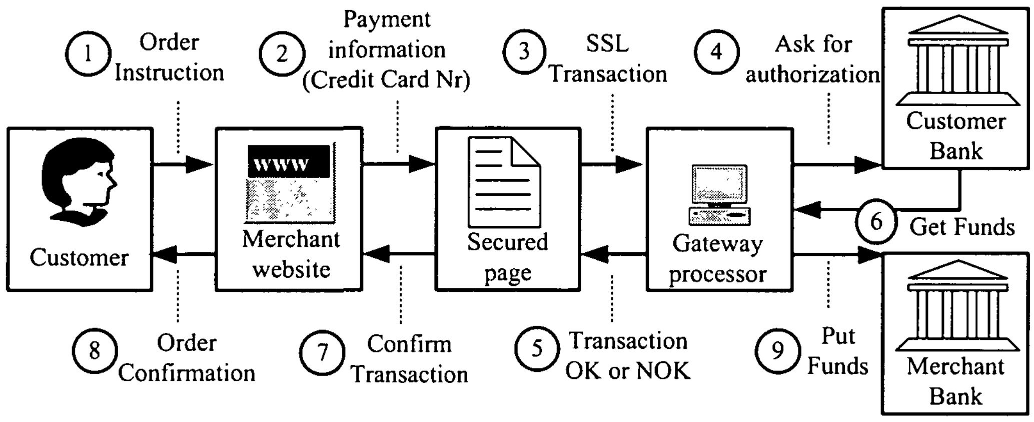


Figure 4.1: Online transaction

The payment software at the merchant's web server checks the customer's payment information and requests an authorization code from the credit card issuer for customer purchase.

The credit card issuer checks the cardholder's expiration date, credit limit and current balance and compares with purchase information. Based on this comparison, the credit card issuer either sends the authorization code back to merchant web server or refuses the transaction.

In the case that the authorization is acknowledged, the merchant's web server sends an approval page to the customer. The customer expects the merchant to ship the goods he has bought or, in the case of download possibilities (program files, music, games etc.), to make them ready for download as soon as possible.

4.3 Privacy and X.509 certificates

Every time a user makes a telephone (mobile) call, purchases goods by using a credit card, downloads different content from the Internet, that information is recorded (stored) in a database somewhere. Furthermore, all these records can be linked together to have a complete picture about user habits. Organizations often link such information for their own protection. If these information lands on some malicious user it does not provide protection, neither for the customer nor for the organization. It is more a threat to user privacy [GS01, HW00, Cha92].

A survey from Jupiter Media Metrix [1] shows that nearly 70% of US consumers think that their privacy is threatened during online transactions [Oet02]. Consumers are afraid that online merchants are gathering more information (habits) about them, that are needed for completing a online transaction. Another survey from Data Protection Commission [2] in Ireland, states that: 56% of online consumers now agree with the statement, "if you use the Internet your privacy is threatened" - compared with just 37% in 1997 [Swe03].

A digital signature over a message ensures the receiver that the message is originated from the claimed sender. Using X.509 digital certificates and PKI systems, (see Section 2.3 about digital certificates and PKI) makes each user action unique and traceable. Furthermore, "if the current visions about global PKI (i.e. the collection of all regional, national, and international PKIs) turn into reality, then everyone will be forced to transact and communicate in what will be the most pervasive electronic surveillance [3] tool ever built" [Bra00].

If the communication channel is not encrypted, the X.509 certificates, due to their properties (unique serial number and digital signature) make it easy for an outside wiretapper to identify communication parties.

Therefore using X.509 certificates a user privacy violation exists with respect to the following [AE00]:

- identity certificates (X.509) reflect identities,
- signature keys are uniquely recognizable,
- CA hierarchies and networks reflect organization hierarchy and personal relationships,
- CAs have access to confidential organizational information, and
- authorization certificates expose business and personal relationships.

A solution to the above X.509 certificate privacy threats will be the reduction of certificate information and the use of session keys [AE00].

4.4 What I don't know won't hurt me

This is an old German maxim [4]. Nobody is hurt by or concerned about things he does not know. Merchant web servers make an attractive target for attackers for reasons such as [Gho98, GS01]:

[1]http://www.jmm.com

[2]http://www.dataprivacy.ie

[3]Surveillance is the act of systematically monitoring, tracking, or evaluation the action of individuals.

[4]Was ich nicht weiß, macht mich nicht heiß

- customers' personal information like: credit card numbers, addresses, financial habits etc.,
- network access to different resources (such as product or payment information), and
- interruption of service in the form of denial of service attacks.

A question arises in this context: why is the merchant web server such a target for hackers and credit card thieves? Our assumption is because they can gain information to customer identities, like credit card numbers, addresses and their habits.

Merchants are becoming an attractive income source for criminals who commit Internet fraud. Furthermore it is easier for a malicious user to break into an online store undetected than an physical store. Gartner Group estimates that online transaction fraud is 17 times more common than in physical store fraud. Furthermore, only in 2003 online transaction fraud is estimated to reach US $1.8 billion [Ver03].

Online fraud effects merchants and consumers. It has been evaluated that about 16% of online consumers have been the victim of credit card fraud and 8% of online customers have had their identity stolen. But the most worrisome claim about Internet fraud is that it has a high grow rate [Ver03].

Without knowing consumers' credit card details (card number and expiration date) the merchant's web server cannot process any transaction. The credit card issuer would not charge its credit card holders only because a merchant knows the credit card holder names. Without credit card number or other equivalent evidence that the credit card holder has triggered the transaction, the credit card issuer will not charge its clients.

In this thesis we propose that merchant should have information about customer identities in such a form that for merchant and credit card thieves are worthless. But this information has value to credit card issuers, and credit card issuers can prove that the online transactions have originated from legitimate credit card holder. Customer identity details should be public information, equivalent to an RSA public key or a X.509 certificate of a customer, as described in Section 2.2 and Section 2.3. In this way merchants would no longer be an interesting target for credit card thieves.

Chapter 5

Network Security for Web Services

Chapter 5 describes different network security solutions and technologies that can be used with Web Services. Security solutions are discussed in details in application, transport and network layers.

The rapid growth in the use of XML and Web Services is challenging the security policies in enterprises. For enterprise administrators it is essential to understand the new approach and new risks that Web Services are introducing. Through Web Services enterprise data is becoming visible to the outside world, thus creating new security threats [BS02b].

5.1 Network configuration

One of the most important issues in the security of Web Services is choosing a proper network configuration. Hardware devices, including routers, firewalls, switches and servers, should fulfill following [HNN02]:

- Secure physical access, i.e. only authorized people inside an enterprise should have access to certain rooms.
- Secure identity access, which means that any changes made on these devices can be traced. Also the identity (user name and password) of local network users is kept secret and is controlled through a security policy.

In order not to compromise the local network, the server offering the Web Services should live in the so called "Demilitarized Zone"(DMZ) area. This area is open to Internet and has less security enforcement than in the local network. A typical DMZ with Web Services exposed to Internet is presented in Figure 5.1.

The elements of a typical network configuration are [Han01]:

Routers – are usually hardware devices located between networks with the main goal of forwarding data packets.

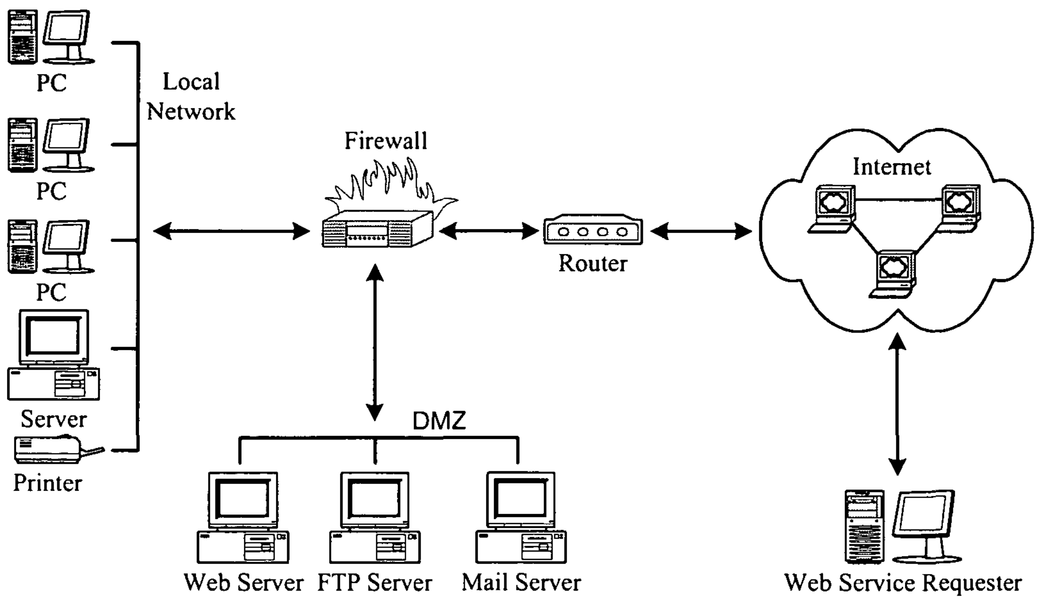


Figure 5.1: Web Service network configuration

Network address translation (NAT) is a method used by routers to allow many computers to connect to the Internet with one (or few) public IP addresses. Indeed, NAT is used wherever the number of hosts that need Internet access exceeds the number of public IP addresses that are assigned to corporate. "Each host [1] on the Internet is assigned a unique 32-bit number Internet address that is used in all communication with that host" [Com00a]. Thus IP addresses are not infinite and theoretically there are 2^{32} Internet hosts possible.

Each Internet address consists of an address pair $(netID\ , hostID)$, where *netID* identifies a network, and *hostID* identifies a host on that network [Rin02]. Internet private addresses are a way of conserving IP addresses. The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space [RMK$^+$94]:

- 10.0.0.0 - 10.255.255.255 ,
- 172.16.0.0 - 172.31.255.255 , and
- 192.168.0.0 - 192.168.255.255.

The first block of IP address represent a single class A [2] network number. The second block represents a group of 16 contiguous class B network numbers and third block is a group of 256 contiguous class C network numbers [RMK$^+$94].

It is the router that translates private IP addresses to public IP addresses. The router keeps track of the translations in an address translation table. Based on the entries in the translation table, routers determine the best route for

[1] A host is a single machine on the network that has an IP address associated.

[2] Internet addresses are divided in three basic classes A, B, and C based on three high order bits. A detailed description of Internet address classes can be found in [Com00a].

forwarding data packets between two networks. To keep the routing table small, routers base their decisions on the destination networks (netID) and not on host IP addresses [Com00a].

The number of hosts connected to Internet is growing daily, as is shown by survey by the Internet Software Consortium depicted in Figure 5.2. The exact number of users connected to the Internet cannot be calculated, but it can be estimated based on number of hosts connected to the Internet [Con03].

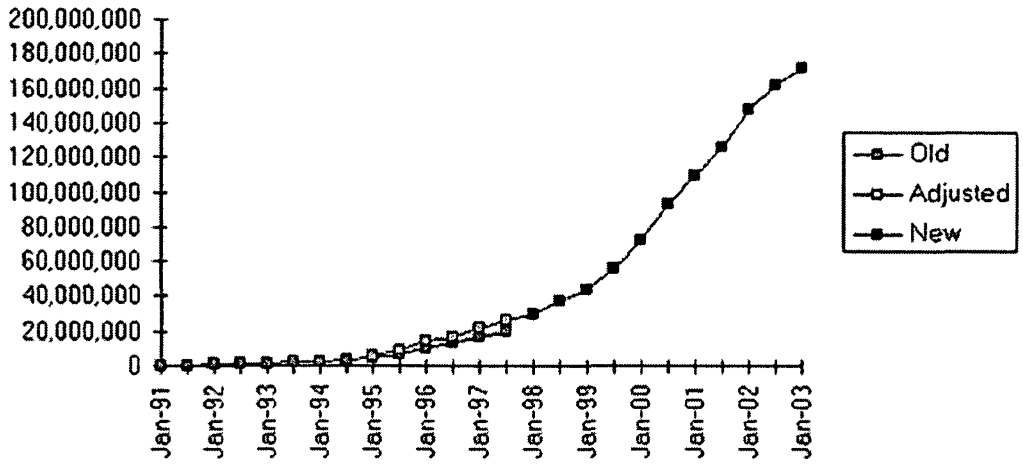


Figure 5.2: Internet domain host count [Con03]

Firewalls – traditionally, a firewall is used to protect one unit in a multi unit building from a fire in an adjacent unit. In computer networks, firewalls are used to isolate a network from danger posed by neighbor networks [Yeo03].

Firewalls protect the perimeter network, e.g. the DMZ shown in Figure 5.1, by simple examination of network traffic. Firewalls based on rules defined by the administrator forward or block the messages from one network to another. Firewalls protect enterprise networks from unauthenticated outside users. More sophisticated firewalls permit enterprise users to exchange data with the outside world, but block the data traffic from outside to inside [CR00].

There are currently two distinct types of firewalls in common use on the Internet today [Fre03]:

- Packet filtering firewalls – take advantage of a machine with several network interfaces and a set of rules to determine whether to block or forward data packets.
- Application level firewalls – act as proxy servers and decide to allow or refuse any connection between an internal network computer and outside. As an intermediary, the firewall can monitor the data exchange between the two networks and block any unauthorized access.

Configuring the firewall presents another security challenge for administrators; special attention should be paid to open ports and where these open ports are routed. Using a second firewall to access the local network could increase security. Passing through many security layers is more difficult for a potential intruder that through a single layer [Han01].

In a network with remote users, who needs more resources that are located in DMZ, the Virtual Private Network (VPN) technology could be used.

Web Server – is usually a computer that runs special OS services, HTTP/1.1 compliant, and listens on a specific port. The most best-known products are [Net03]:

- The Apache web server, the successor to an HTTP daemon developed at the National Center for Supercomputing Applications (NCSA), University of Illinois in 1995. Apache is a web server compliant with the recent HTTP protocols that provides full source code and comes with an unrestrictive license [Fou03].

  Apache has been the most popular web server on the Internet since April of 1996 [Net03]. Figure 5.3 presents the percentage of web sites using different web servers; the Apache server currently with 63% is the leading product.

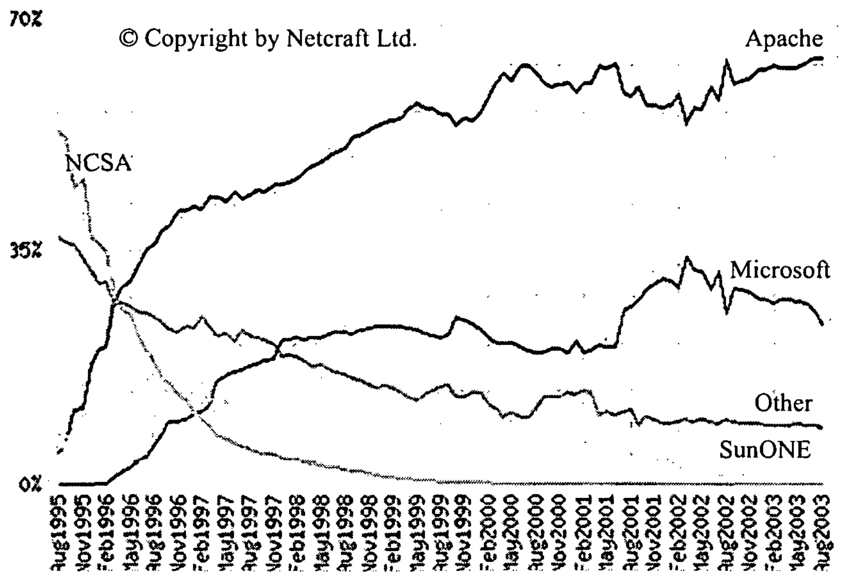


Figure 5.3: Market share for web servers across all domains: August 1995 – August 2003 [Net03]

- Internet Information Services (IIS) from Microsoft Corporation. IIS is a scalable web server. In the new version of IIS, version 6.0, Microsoft has added new features such as fault tolerant process architecture, automatic process recycling and rapid fail protection [Mic02a].

To increase security web servers support SSL authentication and authorization. Microsoft has stated that "50% of all SSL web sites run on IIS 5.0" and in IIS 6.0 it has made some improvements in performance, certification objects, and a selectable cryptographic service provider (CSP) [Mic03c].

5.2 Internet security

When Internet protocols were designed about 35 years ago, data security was not one of the issues considered; their primary goal was achieving interconnection among different isolated hosts and networks [3] [Com00a]. The introduction of HTML and HTTP by CERN [4] in the earlier 1990 made the basis for today's Internet wide spreading and acceptance [BLFF96].

The Internet hides the details of network infrastructure and allows distributed computers to communicate independent of their physical network connection. From the user's point of view the Internet is seen as a single, virtual network to which all machines connect through their physical connection [Com00a].

Security can be applied at three layers [Opp02], shown in Figure 5.4 with gray boxes.

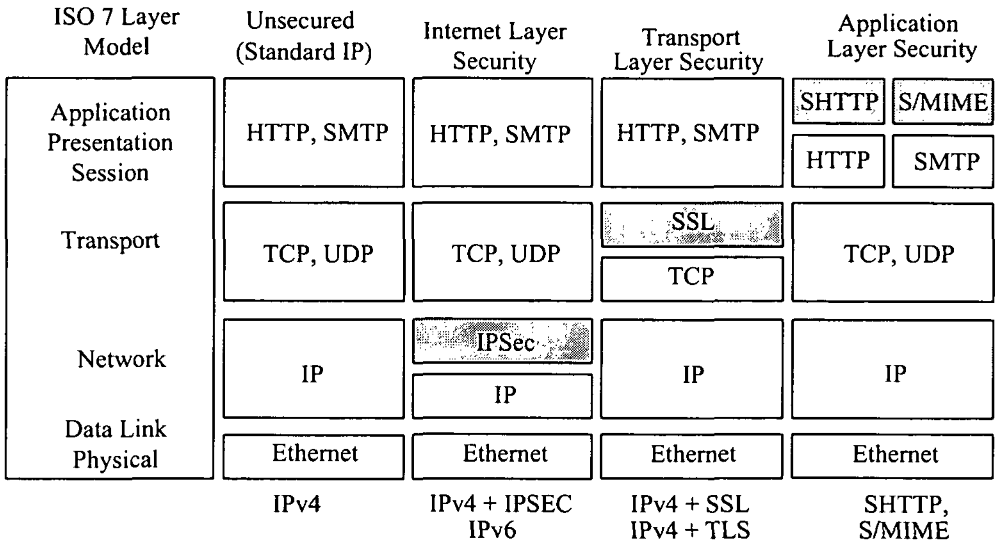


Figure 5.4: Network security levels [Fer02]

Security precautions in the application layer influence only a particular application, using common sub-protocols such as TCP and UDP. Security at the transport layer can be increased through use of SSL, and in the network layer security can be achieved by employing IPSec. The IPSec provides authentication, integrity and confidentiality of message using cryptographic mechanisms (described in Section 2.2). SSL and IPSec do not provide mechanisms for digital signatures [Fer02].

5.2.1 Security in the application layer

Building in security at the application layer is very flexible, because the scope and strength of the protection can be adapted to meet the specific requirements of the

[3]Defense Advanced Projects Research Agency (DARPA) was a project started by the US Department of Defense (DoD) in the mid 1970s that evolved into what today as is known as Internet [Com00a].

[4]http://www.cern.ch

application. Security services can be implemented in the application layer in two ways [Opp02]:

- security services are integrated into each application individually, or
- a generic security system is built, which is used to incorporate security services into any application.

The most representative applications that have implemented security in the application layer are:

Telnet – a terminal emulation program designed for any TCP/IP network. The Telnet enables users to connect to servers and execute commands remotely as if they were sitting physically at the server's console. For a valid Telnet session the user must log into the server by entering a valid username and password [NN03e].

The main disadvantage of Telnet is that the username and password are transmitted in plain text, which represents a great security threat. Several security-enhanced Telnet software packages have been developed to avoid this security risk, such as [Opp00]:

- slogin – is a utility in the *secure shell* (SSH) software package,
- secure Telnet (STEL) – makes use of strong encryption mechanisms to exchange data between client and server. Diffie-Hellman is used as standard key exchange protocol and for bulk data encryption the DES protocol (see Section 2.2) is used.
- secure RPC authentication (SRA) – is a software package developed by Sun Microsystems [5] to protect Telnet and File Transfer Protocol (FTP) connection handshakes against password-sniffing attacks.

Electronic mail – The basis for electronic mail (email) on the Internet is the Simple Mail Transfer Protocol (SMTP) specified in RFC 821, the text and ASCII message syntax specified in RFC 822, and Multipurpose Internet Mail Extensions (MIME) specified in RFC 1521 [Opp02].

SMTP, as specified in RFC 821 does not provide any security mechanisms. Email messages travels over the Internet in plain text. Servers and routers can gain information about senders and receivers with any traffic analysis software tool. Changing and forging message content or the claimed sender in such cases is an easy work.

The basic technique for achieving privacy in email content is so-called *digital enveloping*. Applying this technique means that the sender of email message randomly chooses a session key (usually a DES, triple-DES etc.) and encrypts the entire message with this session key. The session key is then encrypted with the receiver's public key and appended to the message. The receiver of the message first decrypts the session key with his private key and then with this session key he decrypts the email content. In some cases the sender may digitally sign the message.

The best known schemes for securing email are [Opp00]:

[5] http://www.sun.com

- Privacy Enhanced Mail (PEM) – is a standard format for encrypting and digitally signing electronic mail messages, using asymmetric encryption mechanisms (see Section 2.2). PEM is intended for use with existing Internet email systems, as defined by RFC 822 [Ken93]. In early 1993 the PEM working group came up with four specifications, which were approved by IETF as proposed standards for the Internet and are published as series of four RFCs (RFC 1421, RFC 1422, RFC 1423 and RFC 1424). A comprehensive summary of the PEM RFCs is given in [Ken93].

  The main goal PEM is to offer support for message authentication, data integrity and non-repudiation of the origin by using a so called message integrity check. PEM uses symmetric cryptography, DES or triple DES, to provide optional data confidentiality services.

  The PEM specification is compatible with the X.509 ITU-T recommendation, i.e. PEM requires a PKI hierarchy and a X.509 certificates for email clients. This is seen by many authors as fatal overhead of PEM. PEM does not support binary attachments [Opp00, Eli03].

- Pretty Good Privacy (PGP) – is rather an application than a protocol, which is used to secure email messages. The PGP software, which was originally developed by Phil Zimmermann in 1991, today is freely available for noncommercial use. Due to US patent claims and export restriction controls, the widespread availability of PGP has raised many legal and export questions worldwide [Zim00]. The latest version of PGP, 8.0, supports the X.509 certificates and is described in details in [PGP03]. See also Section 2.3.3.

  PGP provides support for data confidentiality, data integrity and non-repudiation of origin services by means of encryption and digital envelopes [Zim00]. The novelty of PGP is that it relies on a so called "web of trust", a decentralized trust hierarchy opposite from PKI (see Section 2.3).

- Secure MIME (S/MIME) – is a specification to add security to email messages in MIME format and is based on the Cryptographic Message Syntax (CMS), as specified in RFC 2630. S/MIME was originally specified by RSA in 1995 (version 1) and is continued by the IETF S/MIME working group. The current version is 3 and is defined in RFCs 2630 to 2634 from June 1999. S/MIME as specified by IETF needs a certification hierarchy compatible with the X.509 ITU-T recommendation. S/MIME relies on Public Key Cryptography Standards (PKCS) to ensure cryptographic compatibility and interoperability among different vendors. The most important PKCS in S/MIME are PKCS#7 (Cryptographic Message Syntax Standard) and PKCS#10 (Certification Request Syntax Standard) [RSA03b, Kei03].

The PEM message format is based on 7-bit text messages. S/MIME is designed to work with text and with MIME binary attachments. PGP relies on users to exchange keys and establish the "web of trust" simply by trusting each other. The "web of trust" works good if number of trusted users is kept small, but for a large number of users it becomes almost uncontrollable. S/MIME uses hierarchies in which the roles of the user and the certifier are predefined. Both

PGP and S/MIME are well integrated in email applications in a plug-in form, making application security simple for users [RSA03b].

WWW Transactions – The continuous growth of the Internet (see Figure 5.2) is mainly driven by the WWW and the HTTP protocol. When the WWW was introduced, by Time Berners-Lee in CERN, the primary goal was data exchange and security was not an issue. It was believed that data on the WWW would be of a public nature. Thus it would not require any form of encryption nor user authentication and authorization. Recently the situation has changed. Nowadays, it is necessary on the WWW to provide data confidentiality, authentication and authorization. In some cases, such as in electronic commerce, a non-repudiation service may be required too.

General security requirements and specific protocol enhancements for HTTP are studied by IETF Web Transaction Security Working Group [BCD97].

Microsoft's IIS web server (from version 5.1) supports these authentication mechanisms [MMVD02]:

- forms,
- MS Passport,
- kerberos,
- basic,
- digest, and
- certificate.

A detailed explanation and comparison of these mechanisms can be found in [Mic01b] and [MMVD02].

There are two basic approaches for authorization in web servers [MMVD02]:

- Role based – whereby users are divided into application-defined roles. Members of different roles have different privileges within the application. Based on the role membership users gain access to function calls. All web server resources are accessed using a fixed and trusted identity (such as a Web Service or application's process identity).
- Resource based – whereby individual resources are secured using so called Access Control Lists (ACL). The ACL determines the user access rights on a specific resource, like: read, write, delete etc. In this case resources are accessed using callers identity.

Secure HTTP (S-HTTP) was designed by Eric Rescorla and Allan Schiffman from Enterprise Integration Technologies (EIT) on behalf of the CommerceNet [6] consortium to secure HTTP connections. S-HTTP supports WWW transaction security by employing cryptographic enhancements to HTTP traffic at the application layer. The S-HTTP does not relay to any individual cryptographic mechanism or key infrastructure [Opp02].

The weakness of S-HTTP, according [Sho95], is the "in band" key exchange. An improper key exchange would be if key B is encrypted with key A. If an

[6]http://www.commerce.net

intruder has the key A, then he can obtain key B by just decrypting the received value with key A. The widespread use of SSL has replaced the use of S-HTTP.

5.2.2 Security in the transport layer

Implementing security in the transport layer is another approach for securing data exchange over the Internet. The best known security implementations in the transport layer are [Opp00]:

Secure Shell (SSH) – is a secure transport layer protocol that enables secure network services over an insecure network [YM03]. SSH is intended to replace Berkely r-tools completely, including *rlogin, rsh, rcp and rdist,* but most frequently is to be used as a secure copy tool. The SSH protocol authenticates hosts and secures the communication channel based on asymmetric cryptography (see Section 2.2.2) [Koe97]. SSH was developed by Tatu Ylönen at the Helsinki University of Technology [7] in Finland.

An SSH enabled server listens for TCP/IP connection on port 22, which is the official port number assigned to it by IANA [YM03]. Figure 5.5 presents an overview of protocol execution.

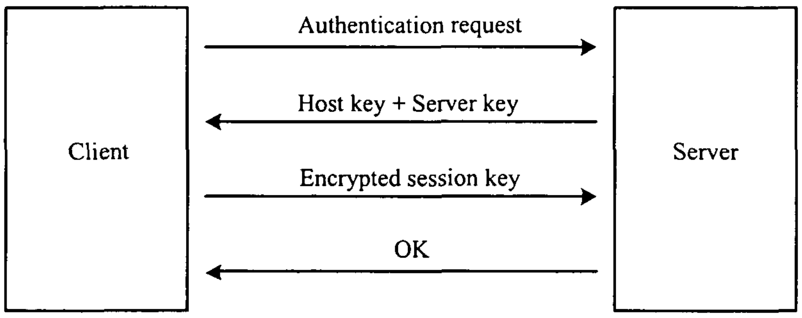


Figure 5.5: A secure shell protocol execution [Opp02]

The session is initiated by the client sending an authentication request to the server. The server, in response, sends the host public key to the client and the server's public key. If the client accepts the host public key, it generates a 256-bit random number that serves as a session key and chooses a symmetric encryption method supported by the server (such as: Blowfish, double DES etc., see Section 2.2.1). In the third step, as is depicted in Figure 5.5, the client makes a double encryption: first it encrypts the session key with the host public key and secondly, the result of this encryption, encrypts with with server's public key. The server decrypts it in reverse order, thus establishing a secure channel based on the session key [Opp02, PLC+01].

There are two versions of SSH: SSH1 and SSH2. The protocols are in fact different and are not compatible with each other. Both protocols are defined as IETF drafts [NN01].

[7] http://www.ssh.fi

Secure Socket Layer (SSL) – is a transport protocol intended to provide security for any network client/server application [WS96].

The protocol is composed of two layers, as presented in Figure 5.6. The SSL Record Protocol is on top of TCP (a reliable transport protocol) and encapsulates all higher level protocols. The client/server authentication protocol is encapsulated by the SSL Handshake Protocol. Negotiating the encryption algorithm and key is done before any plain data are exchanged between two communication parties. SSL is designed to be application independent [FKK96].

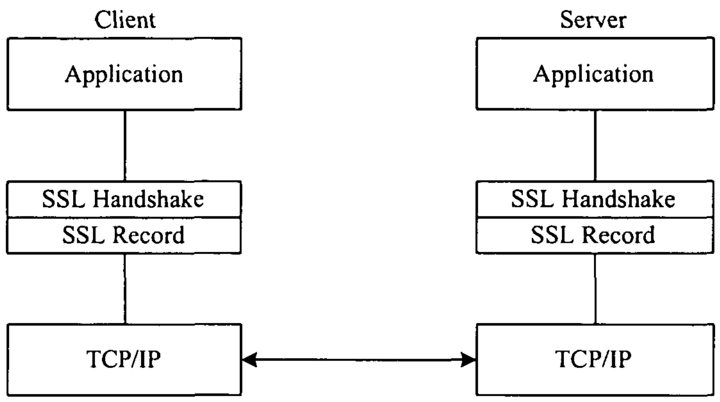


Figure 5.6: The architectural placement of the SSL

The SSL protocol provides TCP/IP connection security that has three basic properties [Opp02]:

- privacy, which is achieved through symmetric encryption (DES, double DES, RC4) after the initial handshake is established,
- communicating parties can authenticate each other using public key cryptography, and
- message integrity is achieved using MAC. Secure hash functions, such as SHA1, MD5, etc. are used for MAC calculation.

The SSL protocol does not provide any mechanism to protect against traffic analysis attacks. Examining the source and destination IP address as well as TCP port number of data packets [8], a traffic analyst can eventually determine which parties are communicating (based on IP address), what types of services they are using (based on TCP port numbers) and in some cases partly recover information about business and personal relationships [WS96].

In order to use SSL, the server must have a specific port number open on which it listens for incoming connections. The port numbers that have been officially assigned by the IANA are presented in Table 5.1.

SSL is widely implemented in HTTP servers, such as Apache and IIS Server. At the client side, SSL is implemented in Internet browsers such as Internet

[8]The source and destinations IP address as well as the TCP port numbers are not encrypted, because these parameters are added to the payload at the network (IP) and transport (TCP) layer respectively. Therefore to every network layer sniffer these parameters are seen in clear text.

Keyword	Port	Description
https	443	Hyper Text Transfer Protocol (HTTP) with SSL support
ssmtp	465	Simple Mail Transfer Protocol (SMTP) with SSL support
snntp	563	Network News Transfer Protocol (NNTP) with SSL support
sldap	636	Light Directory Access Protocol (LDAP) with SSL support
spop3	995	Post Office Protocol version 3 (POP3) with SSL support

Table 5.1: IANA TCP assigned port numbers with SSL support

Explorer, Mozilla [9] and Netscape Communicator. SSL key negotiation is based on asymmetric encryption (see Section 2.2.2) therefore it can only secure the communication channel between two communication parties.

Transport Layer Security (TLS) – is a communication protocol with the primary goal of providing data integrity and privacy between any two communicating participants. TLS version v1.0 is specified by IETF WG as RFC 2246 and is almost identical to SSL v3.0. Although the differences between SSL v3.0 and TLS v1.0 are minor they are not compatible, and TLS v1.0 does not support a mechanism to roll back down to to SSL v3.0. TLS v1.0 does not protect against traffic analysis [DA99].

5.2.3 Security in the network layer

Implementing security in the network (IP) layer enables all services and applications above the IP layer (see Figure 5.4) to exchange securely data with each other. The work on network layer security is supervised by the IETF IP Security Protocol (IPSec) Working Group [10], whose main goal is to standardize the IP Security Protocol and Internet Key Management Protocol (IKMP). The result of the IPSec Working Group was the specification of a comprehensive security architecture known as the IPSec specification (RFCs 2401–2412) [11]. The IPSec specification defines different IPSec architectures, key management and base protocols [Opp00, Til00].

IPSec is a standard that provides authentication, data integrity and privacy in the IP layer. IPSec encrypts data packets at the IP layer, which then can be redirected to any IP network. IPSec requires no additional upgrade to existing networks.

Internet Protocol (IP) provides a connection-less service and its task is to route and send a packet to its destination. IP is a best effort algorithm, i.e. IP does not grant any guarantee for the packets (datagrams) it tries to deliver. IP datagrams travel through a series of routers before they reach their final destination [CDI96]. The IP datagram version 4 is presented in Figure 5.7.

A detailed description of each field can be found in [Com00a, Opp02]. None of the IP fields shown in Figure 5.7 are encrypted and there is no authentication mechanism in IPv4. For an intruder it would be an easy job to change the destination or source

[9] http://www.mozilla.org

[10] http://www.ietf.org/html.charters/ipsec-charter.html

[11] All RFCs can be found at IETF web site at: http://www.ietf.org/rfc/rfcN.txt, whereby N denotes the RFC number.

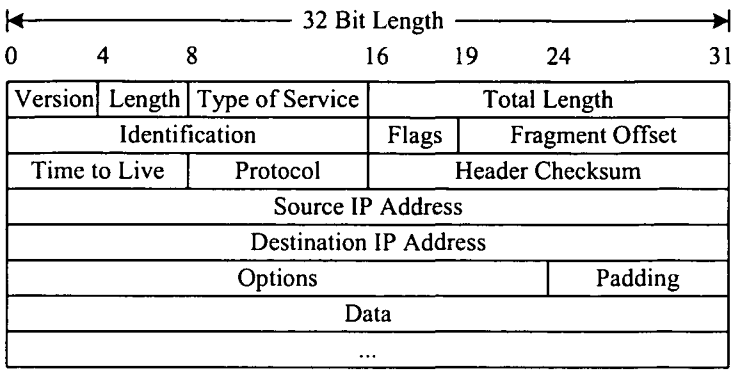


Figure 5.7: Format of an IP datagram, the basic unit of transfer on the Internet [Com00a]

address. The receiver has to assume that the received datagram has originated from the source stated in the IP datagram.

As defined by the IETF, IPSec uses two main elements to protect network communications [KA98c]:

- An authentication header (AH), used for providing source authentication and data integrity. It ensures the final receiver that data has not been modified in transit. But by specification some datagram fields my change in transit, thus AH provides an integrity check only for immutable fields. By default the mutable fields such as Type of Service, Flags, Fragment Offset, Time to Live, and Header Checksum (see Figure 5.7) are filled with zeros at sender and receiver during the integrity check [KA98a].
- Encapsulated security payload (ESP), used to provide confidentiality, integrity and data origin authentication. ESP may be applied alone or in combination with the IP Authentication Header (AH). The ESP header is inserted after the IP header and before the upper layer protocol header (transport mode) or before an encapsulated IP header (tunnel mode) as presented in Figure 5.8 and Figure 5.9 [KA98b].

Both mechanisms, AH and ESP, are based on the concept of a security association (SA). The SA represents the interface between two communicating participants. It sets the IPSec protocol to use for securing the datagrams, the keys, and the time period for which the keys are valid. Any IPSec implementation always builds an SA database (SADB), which is used in the SA of the IPSec protocol to secure data packets. At the receiver site, when the IP packet arrives it can only be authenticated and decrypted if the receiver can link the packet with the context of appropriate SA. In IPSec this parameter is known as the security parameter index (SPI) [DH99].

IPSec uses two modes [KA98c]:

Transport mode – whereby the AH and ESP protocols are applied to data packets flowing from the transport layer to the network layer to provide the desired security. The transport mode of IPSec can be used only when end to end security is desired [DH99]. Figure 5.8 presents the IPv4 data format after applying IPSec for AH or ESP security mode (gray boxes).

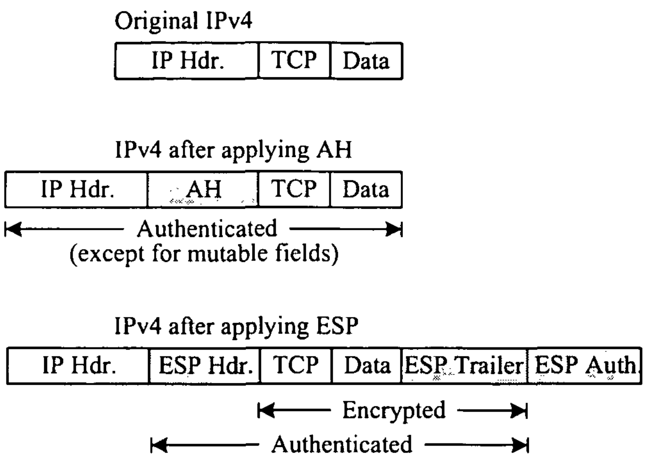


Figure 5.8: IPSec in transport mode [KA98a, KA98b]

AH and ESP can be used together in transport mode. In this case ESP should be applied first, because if the data packets are first protected using AH and then using ESP, data integrity is broken, due to changes to the IP datagram (see Figure 5.8).

Tunnel mode – is used in cases when data packets need to be secured only as far as an intermediary destination. In this case the security is granted by gateways and routers [DH99].

Tunnel mode AH and/or ESP may be employed either in hosts (as end destinations) or in security gateways (as intermediary destinations). When AH and/or ESP is implemented in a security gateway to protect network traffic tunnel mode must be used. In tunnel mode, the "inner" IP header carries the final source and destination addresses, and an "outer" IP header contains intermediary IP addresses, i.e. IP addresses of security gateways, as presented in Figure 5.9. In tunnel mode, AH or ESP protects the entire inner IP packet [KA98a].

IPSec has received criticism for its complexity, "IPsec contains too many options and too much flexibility; there are often several ways of doing the same or similar things" [FS99].

The next generation of IP, IPv6, which has an IP address 128 bits in length supports the IPSec specification from ground up [KA98a, KA98b]. The work and efforts in IPv6 are coordinated by the IETF [12], respectively its IPv6 working group.

[12]http://www.ietf.org/html.charters/ipv6-charter.html

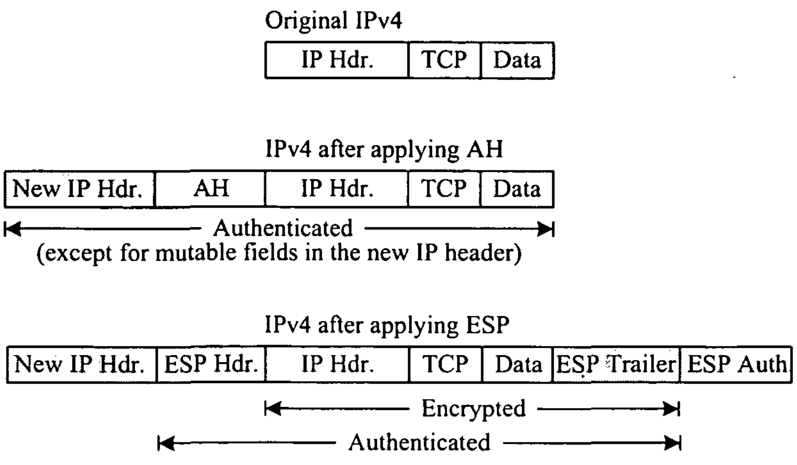


Figure 5.9: IPSec in tunnel mode [KA98a, KA98b]

5.3 Intranet security

An intranet is a computer network based on Internet technology and owned only by one organization. Intranets are usually accessed by organization insiders, whereas outsiders typically have no access or very limited access. Intranets often connect local organization networks at different and distributed sites. In the case of remote sites the communication between different sites is carried out via leased (dedicated) lines or over the Internet. To access an intranet the same browsers are used as to access the Internet [KR00a].

Intranets are becoming more and more attractive targets for external and internal attacks since they hold information designed for organization members only. Intranets are by default protected (configured) against external attacks and internal users have full access to all organization data. Statistics cited at [McC98] state that 47.7% of computer crime comes from inside organizations.

An intranet is a system using physical network components. The basic components are [KR00a]:

- server computers,
- workstations,
- network cables,
- physical interfaces to networks,
- network and transportation protocols, and
- diverse TCP/IP services.

Therefore a detailed background of TCP/IP, Internet protocols and other security related mechanisms are basic preconditions for identifying and avoiding Intranet security threats. Common types of threat include [Win02a]:

- Malicious code attacks – can damage and compromise the security of specific hosts or the entire network. Malicious code attacks are known as viruses, worms and Trojan Horses. They are capable of duplicating themselves and hiding inside operating system files. Once they have infected a host, they can use a list of the user's contacts to spread to other hosts.
- Denial-of-service (DoS) attacks – have the main goal of stopping legitimate users from using network services or to interrupt normal business operations.
- Unauthorized access control – is another method that a malicious user uses to gain access to important internal organization data. Furthermore, he can duplicate, send over email or even destroy data. Therefore, intranet resources should be protected with a carefully designed right management system.
- Mixed threats – are combination of viruses, worms, Trojan Horses, and malicious code that makes intranet servers and hosts more vulnerable to different attacks. These types of attacks can spread very rapidly and cause significant damage.

Protecting intranets against these threats is done by combing different security functions, technologies and products, such as firewalls, intrusion detection, content filtering, and virus protection. The best known model for protection against intranet security threats is model of the Canadian Mounted Police (1980), based on different security levels [KR00a]. The most important of these levels are:

- Communications protection – includes the encryption of the communication channel and is applied in case where different organization units are linked through insecure channel such as the Internet. Here different encryption, firewalls, and filtering technologies could be used.
- Software protection – means establishing the trustworthiness of different scripts and code in intranet applications. All critical Intranet applications must require user authentication in the form of username-password or a X.509 certificate. One important aspect of software protection is virus protection. Virus protection software should be able automatically to retrieve the most recent virus definition files and perform real time scans of hosts.
- Data protection – is tightly linked with user rights and guidelines for resource classification. Special user rights regarding read and delete access must be set when publishing confidential data on an intranet.
- Operation protection – means that security in an intranet should not decrease the efficiency of the system. User rights should be defined based on the role and position that the user has inside organization. Remote access must be carefully designed to avoid any security holes.

5.4 Virtual private networks

A virtual private network (VPN) is a private computer network that exchanges information with other hosts and networks over any public network such as the Internet.

The exchange of data is based on the capability of VPN to emulate the properties of a point-to-point private link. Therefore data packets are encapsulated in a new header that enables routing information and allows data to travel through a public network until it reaches its final destination. For confidentiality data may be encrypted; thus packets that are intercepted on the public network are not readable without the encryption keys [Mic00b, Boe02].

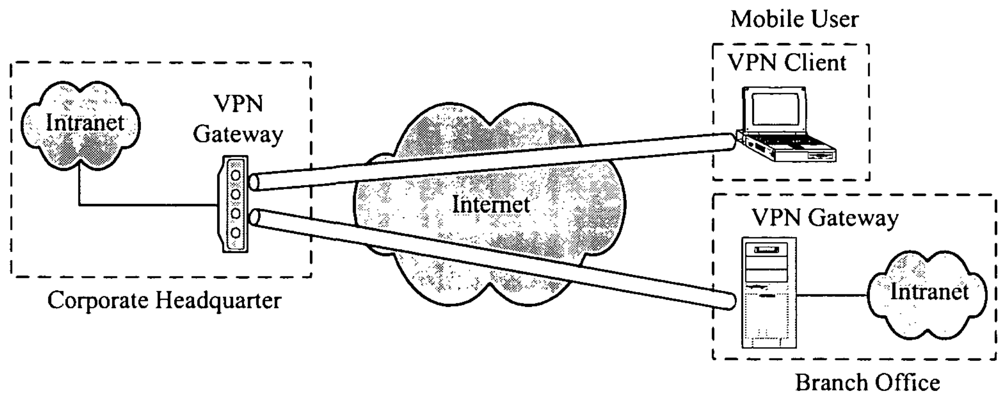


Figure 5.10: VPN Solutions [Bac01]

Figure 5.10 depicts two main cases of applied VPN solutions [Rya01, Bac01]:

- Remote access – is usually used to access email, download files and to execute other transactions. This type of connection is mainly used by small offices that do not have a permanent connection to an organizational intranet. In this case the cost of a VPN are about 60%-80% cheaper than for conventional remote access [Boe02].

- Branch-to-branch or branch-to-headquarters – before the use of VPN these connection were made through dedicated routers and leased lines, at great costs. With VPN the new costs include the deployment and maintenance of VPN gateways at branches and the deployment and maintenance of the VPN server at the headquarters site. In this case VPNs are about 20%-47% cheaper than conventional connections [Boe02]. A detailed analysis of VPN cost advantages is presented in [CK00].

VPNs make use of a tunneling protocol. Sending data packets from one network to another through an intermediate or transit network is known as a tunneling mechanism. Tunneling encapsulates the original data packets inside new temporary header information and sends the encapsulated information for transmission to the intermediate network. The temporary header contains the routing information for sending tunneled data from start to end point. At the tunnel end, the receiver discards the temporary header in order to reconstruct the original data packet. The logical route through which data packets travel from the source network to the final network is known as a *tunnel* [FF01].

There are currently three major tunneling protocols for VPN [Rya01, Mic00b]:

Point to point tunneling protocol (PPTP) – was developed by a vendor consortium consisting of Microsoft, 3COM [13], ECI Telematics [14], Ascend Communication [15], and US Robotics [16] [Boe02]. PPTP is specified by IETF RFC 2637 and is the most widely supported VPN protocol. PPTP does not require a PKI, so it is easy to deploy in cases where high security is not the most require feature. In cases when VPN connections must pass through Network Address Translators (NAT) PPTP will be the best choice, since the IPSec ESP implementation is incompatible with NAT [Mic00b].

Microsoft, as an operating system vendor, has implemented its own algorithm and protocols to support PPTP in the Windows operating system. The first implementation of Microsoft's PPTP suffered from different cryptographic weaknesses, as described in [SM98]. Microsoft updated its PPTP implementation through the addition of support for a challenge authentication protocol known as MS-CHAP v2 and for the Extensible Authentication Protocol (EAP). A security analysis of MS-CHAP v2 is presented in [SMW99]. The latest PPTP improvements provide the capability for using public key certificates and smartcards for user authentication [Mic00b].

Layer 2 tunneling protocol (L2TP) – operates at the data link layer in the OSI reference model (see Figure 5.4) and uses frames as the basic unit for data exchange. L2TP is an extension of PPTP. The L2TP protocol allows tunnel end point participants to establish, maintain and terminate the tunnel with different configuration parameters. L2TP combines the best features of two other tunneling protocols: PPTP from Microsoft and L2F from Cisco Systems [17] [Rya01, FF01].

IP Security (IPSec) – is the security standard for the Internet. IPSec supports authentication and encryption (see Section 5.2.3). The major difference between IPSec ESP tunnel mode and L2TP is that an L2TP tunnel performs at layer 2 of the OSI model (data link layer, see Figure 5.4). This allows L2TP to tunnel protocols such as internetwork packet exchange (IPX) or NetBios [18] extended user interface (NetBEUI) in addition to IP. IPSec's ESP only tunnels IP traffic [Boe02].

IPSec is used in tunnel mode when a gateway provides security services for packets it is forwarding, as is presented in Figure 5.11.

In Figure 5.11 AH and ESP are used in tunnel mode. An IPSec tunneled mode packet has two IP headers: inner and outer (new IP header in Figure 5.11). The inner header is generated by the host and the outer header is added by the VPN gateway. Any packet sniffer that intercepts packets between two VPN gateways in tunnel mode, as is presented in Figure 5.11, cannot understand anything about the IP packets original endpoints. The only readable IP addresses are those from G_1 and G_2.

[13] http://www.3com.com
[14] http://www.ecitele.com
[15] http://www.ascend.com
[16] http://www.usr.com
[17] http://www.cisco.com
[18] Network Basic Input Output System

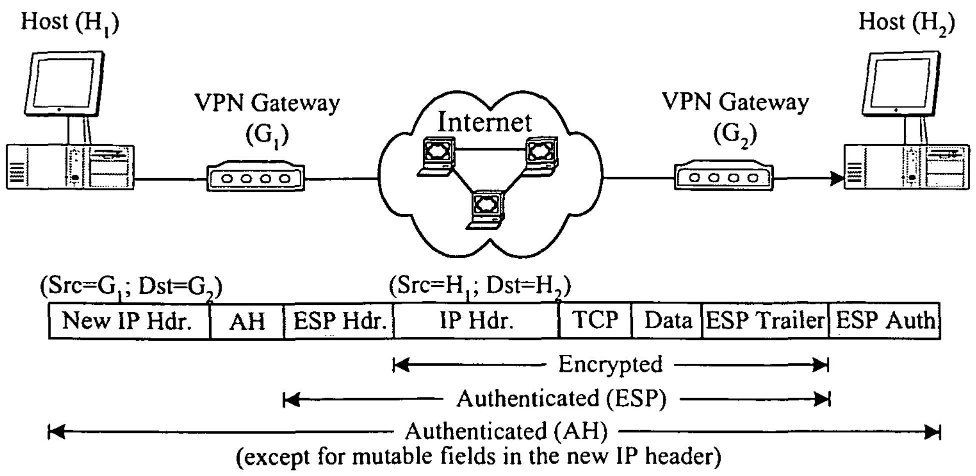


Figure 5.11: VPN IPSec connection in tunnel mode [Boe02]

Enterprises that accept VPN connections, either from remote clients or from branch offices must deploy a VPN server. The position of a VPN server in relation to the firewall could be [FF01]:

- VPN server in front of the firewall – in this case the VPN server has two interfaces: one to the Internet and the other to a private network or firewall. The VPN server must be configured to redirect only authenticated VPN clients to the intranet.
- VPN server behind the firewall – in this case the VPN server is deployed in the DMZ zone along with the other intranet resources like the web server and ftp server. Because the data packets for the VPN server are encrypted (ESP tunneling mode, see Figure 5.11) the firewall cannot determine the final destination address of the data packets. Thus the firewall cannot protect malicious users from accessing unauthorized enterprise resources. Therefore different computer and user level authentication mechanisms should be applied to provide protection against such attacks [FF01].

A VPN server is supported as a service in most operating systems such as different Linux versions and Windows Server.

Chapter 6

Identity Certificates

Chapter 6 presents the first approach in this thesis to prevent violation of user privacy. Privacy violation occurs in current online payments at the time when the user has to enter additional information like credit card number, address etc. while browsing at the merchant web site. The proposed approach is based on asymmetric encryption mechanisms (as described in Section 2.2), X.509 v3 certificate extensions and storing X.509 v3 certificates in a smartcard.

Secure Electronic Transaction (SET), presented in Section 2.5.3, also uses X.509 certificates and private extension [SET97a, SET97b], but in this chapter presents a different approach.

6.1 X.509 certificates

A X.509 certificate is a data structure, that contains information about a specific entity (user, computer, printer etc.) and its corresponding public key, and is digitally signed by issuing authority.

The X.509 specification is an ITU standard for digital certificates. It was first published in 1988 as part of the ITU X.500 directory services standard. A X.500 directory is a database of users, computers, printers and etc. designed for global use, like a telephone book. The latest version of the X.509 specification, version 3, was released in 1996. X.509 certificates are refereed as identity certificates, since they bind identity with the public key stated on the certificate [ITU00, AE00].

A X.509 certificate is digitally signed by a certification authority (CA) to confirm that the identity (name) and other information contained in certificate belongs to the certificate holder (subject) of the corresponding private key (see Figure 2.15).

6.1.1 Structure of the X.509 certificate

The X.509 certificate format is defined in Abstract Syntax Notation One (ASN.1) syntax, a notation for describing abstract types and values. The X.509 certificate is encoded using the ASN.1 distinguished encoding rules (DER) [AFPS99]. ASN.1 DER encoding is a tag, length, and value (TLV) encoding system for each element. If the X.509 syntax had been invented nowadays, it would be defined in XML format.

The X.509 certificate format is specified as follows [ITU00]:

```
Extension ::= SEQUENCE
{
        version                  [0] EXPLICIT Version, DEFAULT v1,
        serialNumber             CertificateSerialNumber,
        signature                AlgorithmIdentifier,
        issuer                   Name,
        validity                 Validity,
        subject                  Name,
        subjectPublicKeyInfo     SubjectPublickeyInfo,
        issuerUniqueID           [1] IMPLICIT UniqueIdentifier OPTIONAL,
        subjectUniqueID          [2] IMPLICIT UniqueIdentifier OPTIONAL,
        extensions               [3] EXPLICIT Extensions OPTIONAL
}
```

Descriptions of the X.509 certificate fields are [AFPS99]:

- **version** – describes the version of the encoded certificate. When extensions are used the X.509 certificate must have version 3 (value is 2, since values are set from 0).

- **serialNumber** – is an integer field and represents the serial number of the issued certificate. The serial number must be unique for the given CA.

- **signature** – identifies the algorithm, such as RSA or Elliptic Curve (see Section 2.2.2), used by a CA to digitally sign the certificate.

- **issuer** – identifies the entity who has signed and issued the certificate. The issuer field must contain a non-empty distinguished name (DN).

- **validity** – represents the time interval during which the certificate is valid. This field is represented as a SEQUENCE of two dates: the start date on which the certificate validity period begins (*notBefore*) and the end date on which the certificate validity period ends (*notAfter*).

- **subject** – identifies the entity whose public key is contained in the subject public key information field. The subject name must be unique for each subject entity certified by a given CA. However a subject may have more than one certificate from any given CA.

- **subjectPublicKeyInfo** – contains the public key and identifies the algorithm with which key is used.

- **issuerUniqueID** – is an optional field supported only in version 3. It allows using alternative names for the issuer.

- **subjectUniqueID** – is an optional field supported only in version 3. It allows using alternative names for the subject.

- **extensions** – provide a way to associate additional information with subjects, public keys, managing certification hierarchy and CRL distribution. The version of the certificate must be 3. These fields enable organizations to define their own extension fields and encode information specific to their needs in a certificate.

The X.509 standard was designed as a general framework. It specifies a solution to a very general problem – authentication in distributed systems – and it is implementation independent [FFW99].

The IETF Working Group (WG) Public Key Infrastructure X.509 (PKIX) established in 1995 develops Internet standards, about 20 Internet drafts and 20 RFCs, necessary for a fully operational Internet PKI [PKI03]. PKIX has specified an On-line Certificate Status Protocol (OCSP) for checking the current status of a X.509 certificate. This protocol eliminates the need for the latest (current) CRL and uses so called delta CRL (see Section 2.3.2) in order to verify certificate status. In OCSP verification is made through *OCSPRequest* and *OCSPResponse* messages [MAM$^+$99].

For our approach the most interesting is the *extension* field. The *extension* field is one that allows capturing any arbitrary data in X.509 certificate [BNP97, AFPS99]. The meaning of the data stored in an *extension* field is interpreted by PKI enabled application. *Thus we are utilizing a X.509 standard for carrying specific data about a certificate holder such as a credit card number.*

6.1.2 SSL – mutual authentication with X.509 certificates

Supporting business-to-consumer applications over the Internet requires the support of distributed authentication. Using X.509 certificate for authentication transforms the authentication problem to a problem of trusting the public key of a particular entity. Which in fact reduces to the problem of trusting the associated certification authority (CA). X.509 certificate serves as a kind of digital passport or credential [Ver02a].

Secure Socket Layer (SSL) operates at the TCP/IP transport layer, just below the application specific protocols such as HTTP, FTP, etc. (see Figure 5.4). The goal of SSL is to authenticate the server and *optionally the client* based on the X.509 certificates and to establish a shared (session) secret key, which is known only to the web server and Internet browser. SSL has become the standard protocol for authenticating web servers and for encrypting exchanged data on Internet [Sch00, Net98].

Figure 6.1 presents an overview of an SSL session. A client sends the *Hello* message, including negotiation parameters like SSL version, key exchange (RSA, Diffie-Hellman etc.), secret key cipher method (DES, TripleDES,), etc. The server in response sends the accepted parameters and its X.509 certificate [1]. If the server has requested a client certificate the clients sends it to the server. During the key agreement process the client generates a 48 byte random number called *pre-master secret.* The client encrypts it with the server's public key and sends it to the server.

[1]And any other certificates that the client needs to build a certificate chain to server's X.509 certificate.

Based on this *pre-master secret*, there are 6 secret keys generated, 3 for each participant [DA99, FHBH+99, Mic03d].

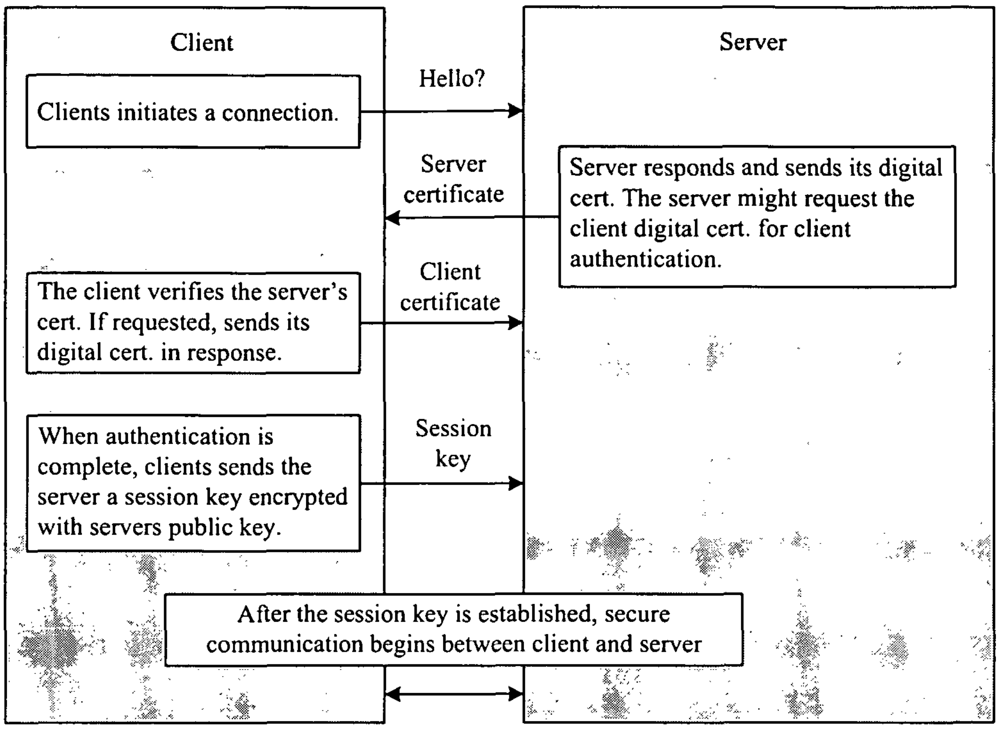

Client
Server
Clients initiates a connection.
Hello?
Server certificate
Server responds and sends its digital cert. The server might request the client digital cert. for client authentication.
Client certificate
The client verifies the server's cert. If requested, sends its digital cert. in response.
Session key
When authentication is complete, clients sends the server a session key encrypted with servers public key.
After the session key is established, secure communication begins between client and server

Figure 6.1: SSL handshake protocol

In real applications the server does not usually authenticate the client, for two major reasons [MB01]:

- The Web (merchant) server verifies only the customer's credit card number through the credit card company. This type of authentication is sufficient for the web server in most cases.
- Most users do not yet have a X.509 certificate, thus enforcing a client certificate means less online customers, and this means less transaction volume. However, merchants are aware of the risk of server only (one side) authentication.

We would like to emphasize that SSL mutual authentication must be enforced since the user payment information is encapsulated inside the X.509 certificate. *Thus a merchant server has the possibility during the authentication process to verify the payment credentials.*

6.2 Approach 1: Using identity certificates

This is the first main contribution of this thesis. This approach is based on X.509 version 3 certificate extensions.

The X.509 certificate holds different information about the issuing CA, the signing algorithms, time validity and personal information about the owner of the certificate. The extension fields, present in X.509 from version 3, are used to add extra information to the certificate. This extra information is specific to issuer or certificate holder and is interpreted by PKI enabled applications and services [BNP97, FFW99].

6.2.1 Private extensions

Version 3 of the X.509 certificate standard (released on 1996) supports the notion of extensions, whereby anyone can define extensions and include them in the certificate. These extensions are called private and they carry specific information relevant to certificate issuer or certificate holder [AFPS99].

Some common private extensions in use today are [BNP97]:

- *KeyUsage* – limits the use of the keys to particular purposes such as "signing-only", "encrypting-only", etc., and
- *AlternativeNames* – allows other identities to be associated with the public key, e.g. domain name service (DNS), email addresses, IP addresses etc.

Certificate extensions posses a flag to indicate that they are application critical and must be processed. In the case that an extension is marked as critical and a PKI enabled application does not know to handle the extension, the application must reject the X.509 certificate as invalid. Usually private extensions are not marked as critical. In this way applications that are aware of the specific extension can check the value. Other applications that are not aware of this extension can continue to process the certificate normally [PS00, Hun01].

In reality users share with different entities (users and institutions) different secrets like: with credit card issuers (American Express [2] for example) user shares the credit card number, with the bank the user shares the bank giro account and with the insurance company the user shares insurance number, and so on, as is presented in Figure 6.2.

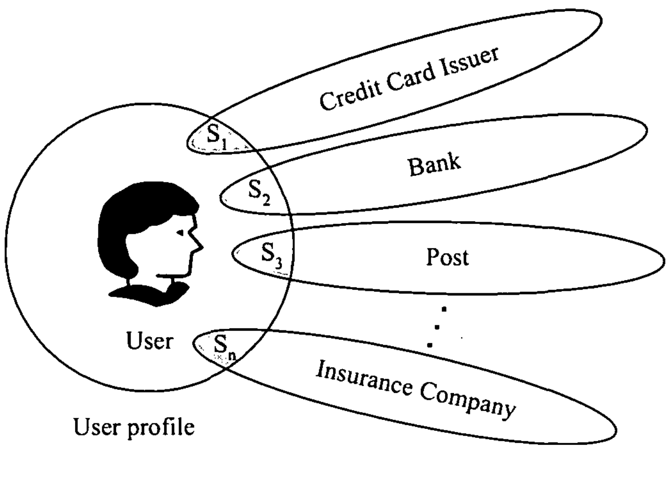
The S_1, S_2, S_3,..., S_n, in Figure 6.2 denotes the secrets that the user shares with other entities. This tuple [3] of secretes, S_1, S_2, S_3,..., S_n, is then stored in an X.509 v3 certificate as private extensions, as presented in Figure 6.2.

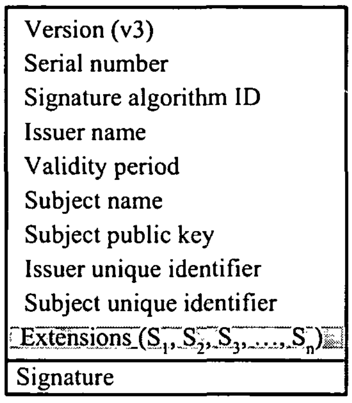
The tuple of secrets, S_1, S_2, S_3,..., S_n is of personal (private) information like:

- credit card numbers,
- bank account number,
- address,
- insurance number, and
- other personal information.

[2] http://www.americanexpress.com

[3] A tuple is an ordered list of elements from a set, usually represented as: a_1, a_2, a_3,..., a_n.




Version (v3)
Serial number
Signature algorithm ID
Issuer name
Validity period
Subject name
Subject public key
Issuer unique identifier
Subject unique identifier
Extensions (S_1, S_2, S_3, ..., S_n)
Signature

X.509 certificate with secretes (S_1, S_2, S_3, ..., S_n)

Figure 6.2: The basic approach, storing the secretes S_1, S_2, S_3,..., S_n in X.509 certificate

The user as the certificate holder is the only one who knows all secrets. A question that arises in this configuration is how the secrets S_1, S_2, S_3,..., S_n must be encrypted so that each entity understands only its respective part. The answer is using public key cryptography, more precisely each secret S_i is encrypted with the public key of the entity i, as is presented in Table 6.1.

Attribute	Encrypted attribute-Extension
VISA	$E(K_{VISAPublicKey}, CreditCardNumber)$
American Express (AMEX)	$E(K_{AMEXPublicKey}, CreditCardNumber)$
MasterCard	$E(K_{MasterCardPublicKey}, CreditCardNumber)$
Bank Account	$E(K_{BankPublicKey}, AccountNumber)$
Address	$E(K_{PostPublicKey}, Address)$
Social Insurance Number	$E(K_{InsurancePublicKey}, InsuranceNumber)$

Table 6.1: New certificate extensions

Encrypting user secret S_i with the public key of entity i and storing the tuple of encrypted secretes, S_1, S_2, S_3,..., S_n in certificate is the basic approach proposed in this thesis.

During the identity certificate issuing process, the issuer must fill in, besides the common fields (name, email address, country etc.), extra fields about the newly extended user profile: credit card number, address, insurance number etc. In this way a X.509 identity certificate with a customer's credit card number as extension can be safely published in a public directory because only the credit card issuer can decrypt it with its private key.

It must be emphasized that the idea of inserting user sensitive information in X.509 certificate extension is not new; a similar approach can be found in [PS00, SET97b]. But lastly, the private extensions are intended to insert private (proprietary) user or issuer information.

6.2.2 Extension structure

Private extensions are supported only in version 3 of X.509 certificates. In [AFPS99] an X.509 certificate extension is defined as follows:

```
Extension ::= SEQUENCE
{
        extnID              OBJECT IDENTIFIER,
        critical            BOOLEAN DEFAULT FALSE,
        extnValue           OCTET STRING
}
```

The X.509 certificate format is defined in Abstract Syntax Notation One (ASN.1). ASN.1 is a language for describing diverse data types and values. The ASN.1 notation provides a number of pre-defined basic types such as [FFW99]:

- integers (INTEGER),
- booleans (BOOLEAN),
- character strings (IA5String, UniversalString),
- bit strings (BIT STRING),

and others. ASN.1 offers the possibility to define constructed types such as: structures (SEQUENCE), lists (SEQUENCE OF), etc. A detailed description of ASN.1 syntax can be found in [FFW99].

A description of each value in an X.509 extension is as follows [ITU00, FFW99]:

- **Object Identifier (OID)** uniquely identifies the extension. In the certificate the OID appears as the extension ID field (*extnID*) and its corresponding ASN.1 structure (*extnValue*) appears as a string.
- **The boolean** field can have either a true or a false value, indicating whether the extension is critical or non-critical. In the case that the extension is set as critical and the application does not understand the extension, the application must reject the X.509 certificate as invalid. In the opposite case, when the extension is set as non-critical and the application does not know to handle the extension, the application can ignore that particular extension and accept the X.509 certificate as valid.
- **Octet string** contains the DER encoding value of the extension. Usually when an application receives the X.509 certificate it checks the extension ID to determine whether the ID belongs to the "well known" IDs. If this is the case, the application will display the corresponding friendly name (like Netscape Comment, Basic Constrains etc.) and the value in its appropriate format.

For integrating (inserting) extensions in a X.509 v3 certificate are distinguished two methods:

1. Using a tree structure, whereby all extensions derived in this form have the same parent OID. Under this parent OID the new extension OIDs are inserted, as is shown in Figure 6.3. Private extensions are organized in hierarchy form, like a tree; it has a root and many sub-leaves and their sub-leaves have many sub-sub-leaves and so on. This is a standard approach for defining new OIDs [NN03d].
2. Not using a tree structure, whereby extensions are appended in different places of the OID tree. For companies that already have their own extension in the IANA [4] list (for example VISA has the OID=15966) it would be uncomplicated to insert a new OID under their main OID.

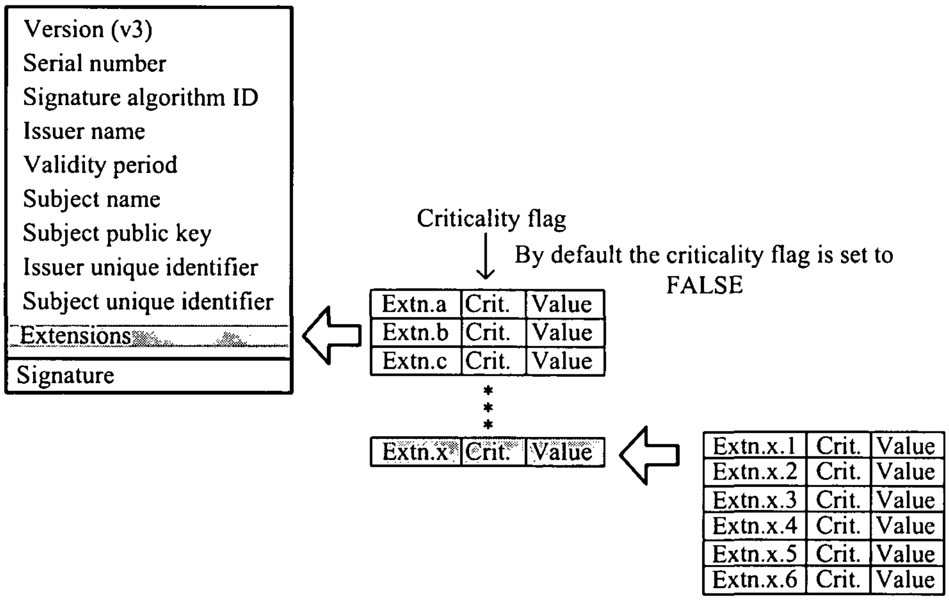


Figure 6.3: X.509 certificate extension structure

Figure 6.3 represents a X.509 v3 certificate with a tree structure of extension OIDs based on the values of Table 6.1.

6.2.3 Object identifier

Object identifiers are a sequence of numbers that are allocated in a hierarchical manner. Object identifiers are used only for certificate extensions. The definition of OIDs is specified in ITU-T recommendation X.660 [ITU92].

The internet private OID-s are identified at node 1.3.6.1.4 and have the following interpretation [NN03d]:

[4] http://www.iana.org

1	ISO assigned OID
1.3	ISO Identified Organization
1.3.6	US Department of Defense
1.3.6.1	Internet OID assignments
1.3.6.1.4	Internet Private
1.3.6.1.4.15600	Last private extension in the IANA list

The value of OID 1.3.6.1.4.15600 was the last OID in the IANA list based on the query made on 02.02.2003, but the list is growing daily. Registering the OID is free of charge for companies. The Vienna University of Technology has no OID registered yet [5].

Let's assume that for the proposed approach we will get an extension 1.3.6.1.4.15601 if request for the new OID were made on 02.02.2003, and in the meantime no other company has requested a OID [6]. Under the extension the 1.3.6.1.4.15601 values described in Table 6.1 must be added. The proposed structure is presented in Table 6.2.

OID	Value
1.3.6.1.4.15601	Root node of new extensions
1.3.6.1.4.15601.1	Encrypted value of VISA credit card number
1.3.6.1.4.15601.2	Encrypted value of American Express credit card number
1.3.6.1.4.15601.3	Encrypted value of MasterCard credit card number
1.3.6.1.4.15601.4	Encrypted value of bank giro account number
1.3.6.1.4.15601.5	Encrypted value of address
1.3.6.1.4.15601.6	Encrypted value of insurance number

Table 6.2: New private OID-s

The values presented in Table 6.2 are used later in Chapter 9 as an example in a case study. Using X.509 certificates for authentication in a client server environment as presented in Section 6.1.2 needs these OIDs in certificates:

- 1.3.6.1.5.5.7.3.1 for server, and
- 1.3.6.1.5.5.7.3.2 for client.

Where the OID 1.3.6.1.5 is the root node of all security related objects [NN03d].

6.2.4 Storing X.509 certificates in smartcards

Smartcards have been proven to be secure, tamper-resistant and very suitable devices for storing sensitive information such as private keys and certificates [Bra00].

Smartcards can process data in intelligent form by taking different actions based on authentication level. Memory access to smartcards for security relevant data is always protected by PIN and only after successful PIN presentation will the smartcard allow memory access. In the case of several false PIN presentations, the chip

[5]Based on the query made on 02.02.2003

[6]The actual list of OIDs can obtained from http://www.iana.org/assignments/enterprise-numbers

(microcontroller) is blocked from further memory access, protecting the information it holds from unauthorized access. *Therefore, in this thesis we propose storing X.509 certificate and its counterpart, the private key, in smartcard.*

Integration of smartcards into operating systems is managed by operating system services (see Section 2.4 and Figure 2.21), and their cryptographic functionality is provided at the application layer usually by two types of libraries:

- Cryptographic Service Provider (CSP) – is used by all Microsoft products such as Internet Explorer, Outlook etc. The operating system manages different CSPs for different hardware and software security tokens. All their functionality is encapsulated by the operating system CryptoAPI layer, thus the application communicates directly with CryptoAPI to access CSP functionality. A detailed description of Microsoft's CSP can be found at [MSD03a].
- Public Key Cryptographic Standard 11 (PKCS#11) – is used in Netscape, Mozilla and other non-Microsoft products. PKCS#11 represents a standard interface to perform different cryptographic functions independent of security token. A detailed description of the PKCS#11 interface can be found in [RSA03a].

The advantage of this layer model is that new smartcards can be added into the system at any time. The new smartcards just need to have a CSP or PKCS#11 library. The application functionality is not affected by these changes.

The Infineon SICRYPT [7] Secure Token Platform (STP) is suitable for different user-driven secure token applications such as smartcards, Universal Serial Bus (USB) dongles, etc. SICRYPT [8] combines the high performance of the Infineon Security Controller of the SLE66Cxx family with the Smartcard Operating System (SCOS) [Inf03c].

SICRYPT smartcards are therefore suitable for storing private keys and certificates. In no way is the proposal restricted only to Infineon smartcards; there are just a set of security features that must be fulfilled by smartcards such as:

- PIN memory protection,
- separate elementary files (EF) for private key and public certificate, and
- smooth integration in operating systems (existence of CSP or PKCS#11 library).

The SICRYPT Secure Token Platform offers two levels of memory access. The user level is protected with so called "User PIN" and the second level is protected with "Administrator PIN" which acts as an unblock PIN [Inf03a]. Figure 6.4 presents the file structure of the dedicated file DF_{CSP} of the SICRYPT smartcard. A complete view of the SICRYPT file structure can be found in [Inf03c].

Storing a X.509 certificate and its counterpart, the private key, in a smartcard has the following advantages:

[7]SICRYPT is an acronym for SIemens (Infineon was formerly part of the Siemens group) and CRYPTography.

[8]http://www.sicrypt.com

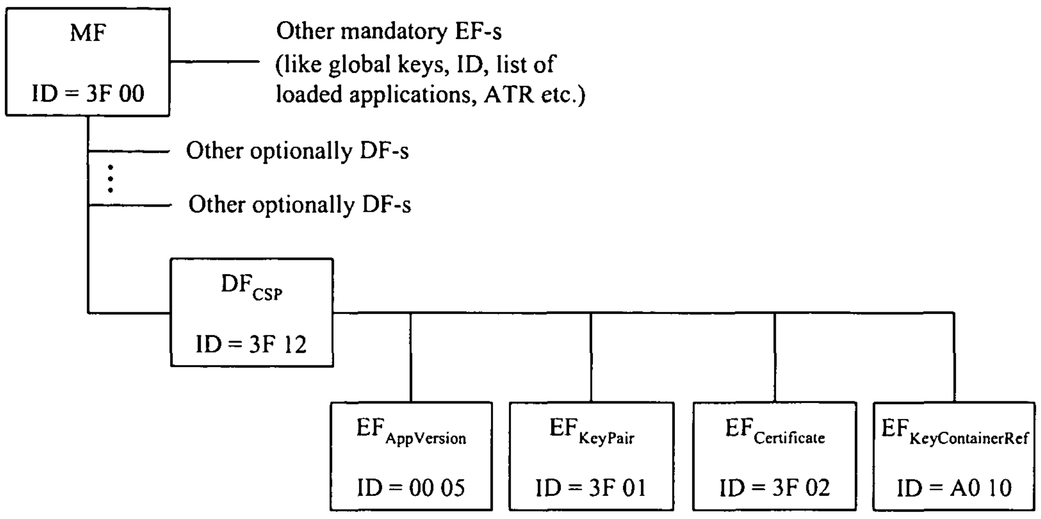


Figure 6.4: Infineon DF_{CSP} file structure [Inf03c]

- Security – The X.509 certificate and corresponding private key are stored in two different elementary files (EF), see Figure 6.4. The write access to the dedicated file DF_{CSP} is PIN protected. The elementary file $EF_{KeyPair}$ is also PIN protected, i.e. the user controls each access to the private key. Each application or service that needs access to the private key (for signing or verification purposes) needs to get the read PIN from the user. The $EF_{Certificate}$ always has read access, i.e. it is not PIN protected. Introducing the certificate into the system means only copying the certificate from $EF_{Certificate}$ to the system or the user certificate store. The key in $EF_{KeyPair}$ cannot be copied; it never leaves the smartcard.

- Mobility – Smartcards are portable devices and users can carry them in their wallet. Besides work and home PCs users can use also public terminals (like Internet Cafes etc.) without fearing that copies of their private key would be left in the system. Even in the case when the user loses the smartcard, without PIN knowledge it will be worthless to a malicious user.

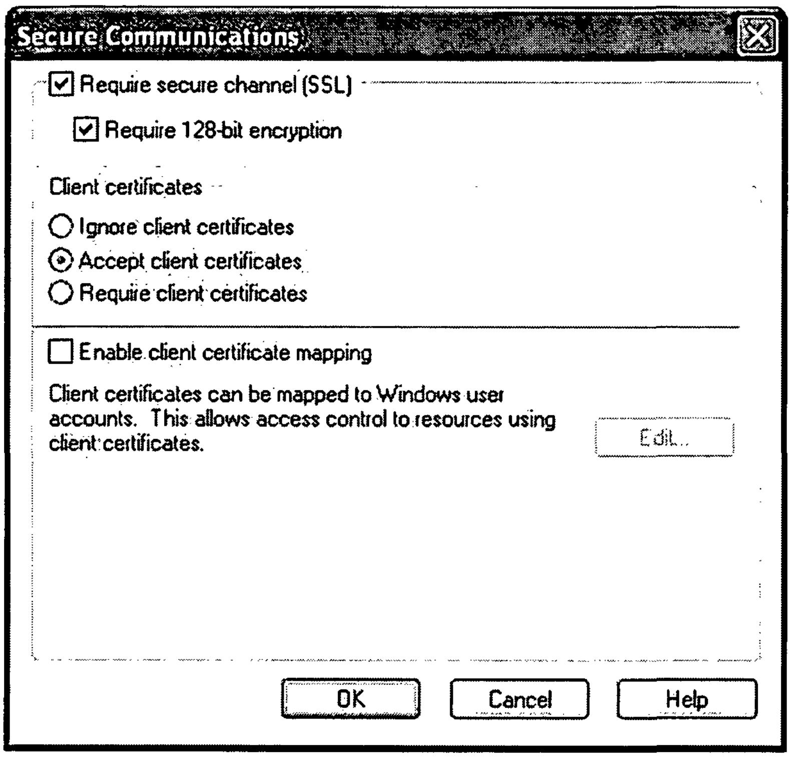
- Compactness – Through use of extensions it is possible to have different payment means (all credit card numbers and all giro accounts) in one certificate, or one card. Other user properties (address, insurance number, etc.) can also be integrated, thus making the user profile more compact.

6.3 Web Services and X.509 certificates

In [ZBL03, See02, Mic02c] approaches are presented that are used to realize and implement a Web Service over SSL. Securing Web Services with SSL means securing the channel between the client and a web server.

Configuring a Web Service with SSL returns to configuring Internet Information Server (IIS) in Windows or Apache Server in Linux, to require certificates. Configuring an IIS to require SSL and to accept client certificates is just a matter of few folder settings as described in [See02].

We propose to set the client certificate (see Figure 6.5) of the folder setting where the Web Service resides as "*Accept client certificates*". Thus the changes in the web server are minimal, but Web Services can benefit from the information in client certificate. The advantage of this setting against the setting "*Require client certificates*" is that for users who do not posses any X.509 certificate the authentication process is not cancelled. In this case only server side authentication is accomplished.


Secure Communications

[✓] Require secure channel (SSL)

[✓] Require 128-bit encryption

Client certificates

○ Ignore client certificates

[✓ Accept client certificates]

○ Require client certificates

[ ] Enable client certificate mapping

Client certificates can be mapped to Windows user accounts. This allows access control to resources using client certificates.

Figure 6.5: Folder settings in Windows IIS for requiring SSL and accepting client certificates

After the mutual authentication is successfully completed, the Web Service based on the private extension in the X.509 certificate can prepare the payment possibilities (VISA, American Express etc.) in a background process, while the client continues shopping at the online store. Furthermore the Web Service can forward the client certificate to credit card issuer (or to other intermediaries) for verification purposes and all this *without violating the user privacy.* The general part of X.509 certificate is understood by anyone, but encrypted extensions (where user private information are stored) are understood only from one instance. Each instance understands only its respective part.

Using Web Services with SSL also requires changing the routing port of the SOAP protocol from HTTP (port 80) to HTTPS (port 443). This information is stored in WSDL file [ZBL03].

Chapter 7

Attribute Certificates

Chapter 7 presents the second approach in this thesis for increasing user privacy in online transactions using attribute certificates.

Secure electronic transactions require both authentication and authorization. A public key certificate proves only the identity of certificate holder. However, an identity certificate does not state what a holder can or cannot do. Attribute certificates (AC) were developed to provide this type of access control. An attribute certificate does not contain a public key, but instead a pointer or link to the public key certificate to which the attributes apply. The main advantage of attribute certificates is that they usually have a shorter life time than identity certificates. We distinguished two types of attribute certificates: long life and single use.

7.1 Structure of attribute certificates

Attribute certificates are standardized in the ITU-T X.509 standard. The structure of an attribute certificate is the same as an identity certificate but the attribute certificate does not contain a public key. Instead it contains the attributes (access roles) for the certificate holder. An identity certificate is issued by a certificate authority (CA) and an attribute certificate is issued by an attribute authority (AA). Usually the CA and AA are different entities [ITU00].

The ITU-T defines the attribute certificate structure as:

```
AttributeCertificateInfo ::= SEQUENCE
{
        version                      Version DEFAULT v1,
        holder                       Holder
        issuer                       GeneralNames,
        signature                    AlgorithmIdentifier,
        serialNumber                 CertificateSerialNumber,
        attrCertValidityPeriod       AttrCertValidityPeriod,
        attributes                   SEQUENCE OF Attribute,
        issuerUniqueID               UniqueIdentifier OPTIONAL,
        extensions                   Extensions OPTIONAL
}
```

The link to the public key is made by the *holder* field (see Section 7.2). The *attribute* field contains the attributes associated with the *holder* of the identity certificate. The other fields of attribute certificate have the same meaning as in the identity certificate (see Section 6.1.1).

A user may have many attribute certificates associated with one identity certificate. The client presents his attribute certificate in one of two models [FH02]:

- *Push* – the client pushes his attribute certificate to the server (see Figure 7.1). This approach has two advantages: no other connections are needed between client and server, and the server does not need to make an extra search for the client AC (this improves performance).
- *Pull* – the client authenticates against the server, and the server pulls the client's attribute certificate from the issuer or some public directory (see Figure 7.1). The *pull* model has an advantage in that it can be implemented without changing the client-server protocol.

The exchange of attribute certificates between client, server and public repository in the push and pull model is presented in Figure 7.1.

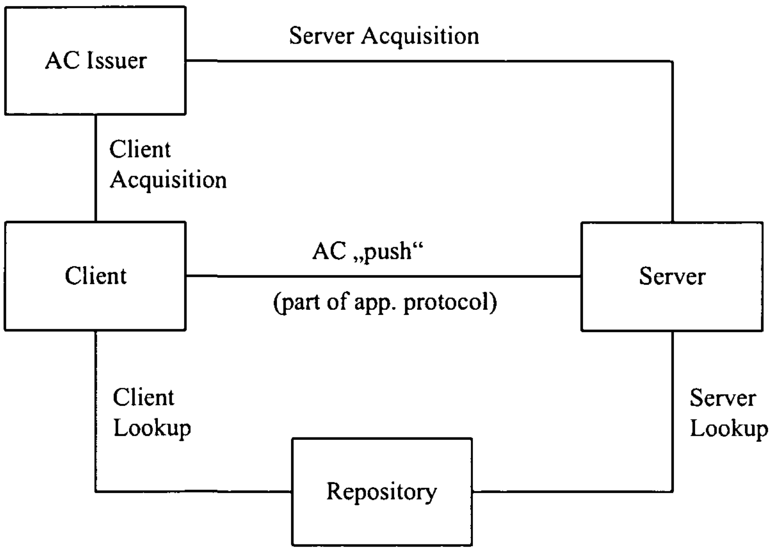


Figure 7.1: Exchanging attribute certificate [FH02]

The attribute certificate framework provides a basis for building a Privilege Management Infrastructure (PMI). This infrastructure support access control applications. Attributes in a public key certificate can also be inserted through the extensions [ITU00]. [COHL02] describes a generic and flexible policy based on Privilege Management Infrastructure that stores user privileges as roles inside X.509 attribute certificates.

An attribute certificate in the form of XML card, called a "Capability Card" is proposed in [OSM98]. The capability card is used to give its holder access to specific resources like web sites, instant messaging, IP telephony etc. The capability card can carry additional information about the issuer authority and holder access privileges (or capabilities). The capability card supports the delegation model, whereby an entity can delegate his capability card to others. This is achieved using three XML delegation control parameters: depth, propagate and width. The XML card supports digitally signed and unsigned cards, whereby the unsigned XML cards can be used in any environment where security is not an concern. Other security details regarding the XML capability card can be found in [OSM98].

An attribute certificate in ASCII format is described in [TJM+99], whereby the AC consists of key words and value tuples which are signed by the issuer. These values include a validity period and a unique ID for the AC.

Authentication with attribute certificates is not possible since they do not contain the public key (and thus also do not contain the private key). Therefore the attribute certificates holders must have an identity (public key) certificate that is referenced in the attribute certificate. So attribute certificates are valid only if they have a valid link to identity certificates. Thus it does not make sense to stretch the validity of the attribute certificate beyond the identity certificate.

If attribute certificates have a relative short lifetime (one day, or even one hour), there is no need to revoke them. Instead, just simply wait for them to expire, and refuse to renew them. This gives more detailed control over users and eliminates the need for complex Certificate Revocation List (CRL) distribution and management procedures [Bal03, ITU00].

7.2 Binding identities and attributes

Attribute authority through its digital signature assures the correctness of attribute certificates. Attribute certificates are linked to identity certificates thorough the *holder* field. The *holder* field has the following structure [ITU00]:

```
Holder ::= SEQUENCE
{
      baseCertificateID      [0] IssuerSerial  OPTIONAL,
      entityName             [1] GeneralNames  OPTIONAL,
      objectDigestInfo       [2] ObjectDigestInfo  OPTIONAL
      --at least one of the  values must be present
}
```

The attribute certificate should use the value of *baseCertificateID*, which is also recommended in [FH02], to link to the public key certificate.

Three methods of binding identity and attribute certificates are described in [PS00]:

- Monolithic Signature – is suitable if only one authority exists that controls identity and attributes. In this type of binding both the ID and the attribute certificate exist in a single certificate. This approach has a disadvantage, because once the certificate is issued, the certificate information cannot be changed. The approach is similar to a private extension. This method is used to store the ID certificate in smartcards [SH02].

- Autonomic Signature – supports multiple CAs and different lifetimes of identities and attribute certificates. Binding information is not fixed. Applications are responsible for determining the binding type. It can be the subject name, subject public key, or some other ID depending on application policy, as depicted in Figure 7.2. If for example the "Name" (see Figure 7.2) is used as the binding information in the attribute certificate, then the attribute certificate holder can use different ID certificates (signed by different CAs) to prove the correctness of attribute certificate.

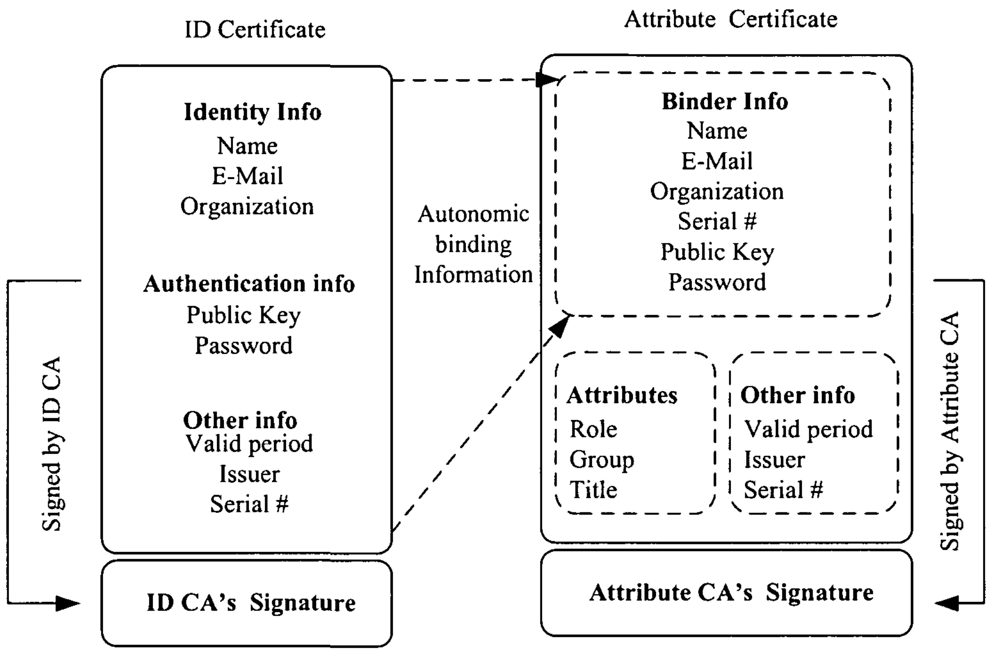


Figure 7.2: Binding ID certificate and attribute certificate [PS00]

- Chained Signature – this is tightly binding to the ID certificate signature. This type of binding does not allows multiple binding of attribute certificate to identity certificates because of uniqueness of the digital signature on an identity certificate.

Entities may acquire privileges in two ways [ITU00]:

- An AA may assign privileges to an entity through the creation of an attribute certificate. This certificate is stored in a publicly accessible directory and is processed by one or more privilege verifiers to make an authorization decision (see Figure 7.1).

- Alternatively, an entity may request a privilege from some AA. The attribute authority may return the attribute certificate (only) to the requesting entity, which explicitly shows it when requesting access to some protected resource.

In RFC 2904 an analogy is made between digital identity certificates, attribute certificate and paper based certificates like passports. A passport identifies the owner (inside and outside his country) and tends to be valid for a relatively long period. A digital certificate represents a user in Internet and in intranet applications. An attribute certificate is more like an entry visa for a country. It is typically issued by an authority different to the passport issuing authority. It does not have long validity period as a passport; it may even be valid for only one entry (single use). Obtaining an entry visa only requires a valid passport (in order to authenticate the owner's identity). The entry visa specifies the conditions under which the passport owner can enter the country [VCF$^+$00].

7.3 Approach 2: Using attribute certificates

In this section the second main contribution of this thesis is presented. User extended attributes like credit card numbers, bank accounts, address, insurance number etc., are encrypted and stored in X.509 attribute certificates. In electronic transactions, a bank is a good candidate to be a attribute certificate issuing authority. It authorizes (privileges) a user to make electronic transactions. The process of issuing and verifying the attribute certificate is presented in Figure 7.1. Users that utilize this approach have to face more complexity. They have to manage several certificates, one identity certificate and for each property one attribute certificate.

7.3.1 Attribute and extensions fields

The attribute certificate as defined by the ITU-T standard has two fields, attribute and extension, where the certificate issuer, holder or some third entity can insert the attributes regarding the attribute certificate holder [ITU00].

The attribute field of the attribute certificate can contain any data. Standard types of attributes are the following [Nyk00]: service authentication information, access identity, charging identity, group, role, clearance etc.

Let us denote user secrets S_1, S_2, S_3,..., S_n as in Figure 7.3. This tuple of secrets, S_1, S_2, S_3,..., S_n, which the user shares with other entities, represents personal (private) information like:

- credit card numbers,
- bank account number,
- address,
- insurance number, and
- other personal information.

Each secret S_i is encrypted with the public key of the entity i, as is presented in Table 6.1 in Section 6.2.1. Encrypting the secrets in this way (with different public keys) prevents any kind of privacy violation, since it prevents the secrets from being shared among the entities. Each entity understands the public information on the attribute certificate and one secret, decrypting it with its private key.

This tuple of secretes, S_1, S_2, S_3,..., S_n, is then stored in an X.509 attribute certificate either in the:

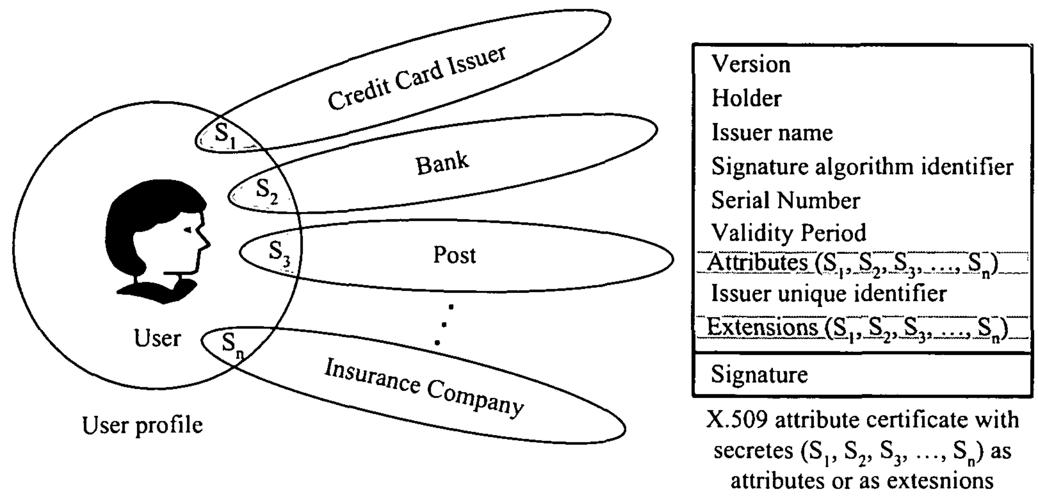

Credit Card Issuer
Bank
Post
Insurance Company
S_1
S_2
S_3
S_n
User
User profile
Version
Holder
Issuer name
Signature algorithm identifier
Serial Number
Validity Period
Attributes (S_1, S_2, S_3, ..., S_n)
Issuer unique identifier
Extensions (S_1, S_2, S_3, ..., S_n)
Signature
X.509 attribute certificate with secretes (S_1, S_2, S_3, ..., S_n) as attributes or as extesnions

Figure 7.3: Inserting secretes S_1, S_2, S_3,..., S_n in attribute certificate

- attribute field, or
- extension field,

as presented in the gray boxes in Figure 7.3. In both cases the structure of the field would be the same as presented in Section 6.2.2, having a pair of unique OIDs and encrypted values.

Generally, attribute certificates tend to have shorter lifetimes than identity certificates. *Based on the lifetime of the attribute certificates we distinguish two approaches: using long life or single use attribute certificates.*

7.3.2 Long life attribute certificates

The attribute certificate that is valid for several hours to the expiration of the identity certificate is denoted as *a long life attribute certificate.* Extending the time validity of an attribute certificate beyond the validity of the identity certificate breaks the authentication chain and the attribute certificate will be marked as invalid by verifying application.

American Express (AMEX) offers its users an online service called *Private Payment*to increase the security of Internet shopping [Ame01]. Private Payment enables American Express card holders to use an temporary *transaction number* instead of an actual credit card number to make online purchases. Only the American Express as the issuing and verifying authority is able to link the temporary transaction number to an existing credit card number. Thus all purchases with Private Payment all directly billed to the corresponding credit card account. Users can (but must not) generate a new transaction number for each online purchase. The Private Payment transaction number is time limited, expiring within a minimum of 30 days and a maximum of 67 days from issue date. During this time merchants have enough time to process (verify) customer orders. The frequent changes of the transaction number reduce the possibility of fraud. Further details about Private Payment can be found in [Ame01].

The temporary transaction number in the American Express case acts as an attribute certificate, i.e. it authorizes (privileges) a user to make online transaction and is valid for some time (but not beyond the validity of the credit card). The format of the transaction number is proprietary to American Express. It is neither an ITU-T recommendation nor in XML format, since the American Express has in this case a double role as:

- issuer of transaction number, and
- verifier of transaction number.

Issuing attribute certificates in proprietary format in cases where $issuer = verifier$ only simplifies the online transaction process, since most companies had their own formats before the ITU-T specification become a standard.

7.3.3 Single-use attribute certificates

By a single-use attribute certificate are denoted those attribute certificates that *are valid for short period of time (several hours) and only for one online transaction.* This means that the entity that makes an online transaction has to acquire an attribute certificate before each transaction.

In the American Express case [Ame01] (see Section 7.3.2) implies acquiring a new *Private Payment* transaction number before each transaction. This provides increased insurance against fraud.

The main advantage of the use of the single-use attribute certificates is that they do not need to be revoked, since their lifetime is limited to a few hours and therefore there is no need for certificate revocation lists (CRL).

In general, using single-use certificates, both identity and attribute, demands that the user enroll and download his certificate and the private key to the software in different environments (Windows, Linux, etc.). If by any case the software leaves behind any copies of the user private key, this does not compromise the system, since the certificate is automatically invalidated after its first use. A detailed description of single-use certificates is available in [HS98].

Chapter 8

Non-repudiation

X.509 certificates with encrypted credit card number as an extension, as presented in Sections 6.2 and 7.3, are usually saved in a public repository and are distributed to many users. The merchant and credit card issuer must be assured that the transaction was originated by the legitimate credit card holder.

Non-repudiation is a mechanism by which the sender of message cannot claim afterwards that he did not send the message, as described in Section 2.2. We make use of the protocols described in Section 2.5 for achieving non-repudiation. These protocols assure the user's privacy.

We discuss also in Chapter 8 the advantages and disadvantages of the approaches proposed in Sections 6.2 and 7.3.

8.1 Assuring privacy and non-repudiation

Privacy is achieved only through good encryption methods [Sch96]. Using the encryption methods described in Section 2.2, the merchant should not know about payment information and the bank should not know about order information (goods).

This section discusses protocols for achieving non-repudiation, data authenticity and integrity over an open channel like HTTP or HTTPS. The message flow is similar to SET, as described in Section 2.5.3, with the difference that the consumer's credit card number is now stored encrypted in a X.509 certificate.

8.1.1 Achieving non-repudiation over Internet

The message flow, as shown in Figure 8.1 describes the structure and flow of the exchange messages between client, merchant and bank for achieving non-repudiation and user privacy. This protocol is similar to the physical store protocol presented in Section 4.1.

Let assume that the product selection and price negotiation between consumer and merchant have already happened, and so these parts are not included in the protocol. Furthermore the protocol assumes that client (consumer, user), merchant (service provider) and bank (credit card issuer) are always online, i.e. are connected to the

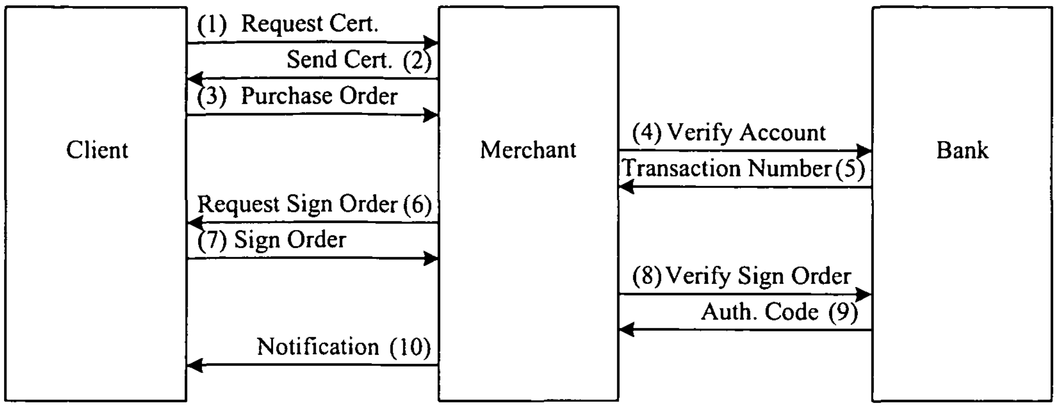


Figure 8.1: Exchange message protocol set over Internet

Internet during a transaction. Offline (disconnected) web servers are worthless, since they are not accessible.

Steps 1 to 10 are executed in a sequential manner as is in Figure 8.1. It is assumed that the X.509 certificate exchange between the merchant and bank has already happened. The message flow is as follows:

Step 1. The client requests a public key certificate from the merchant (if it does not have yet).

Step 2. The merchant sends its certificate to the client.

Step 3. After client has validated the merchant's certificate (i.e. client checks if certificate is issued from a trustworthy authority, time validity and is not in a CRL) he encrypts his request with the merchant's public key. The format of the user purchase order (information order) is presented in Figure 8.2.

Encrypted Goods	X.509 Certificate

Figure 8.2: User purchase order format

The client's X.509 certificate is not encrypted, since it can be available from any public repository. The goods are encrypted with merchant's public key, $E(Merchant_{PublicKey}, Goods)$.

Step 4. The merchant decrypts the message with its private key. It checks the validity of the client certificate with following criteria:

- if the certificate is issued by a trustworthy authority,
- if the certificate lifetime has not expired, and
- if the certificate is not in a CRL.

If the certificate does not fulfill all the above criteria, the merchant marks it as invalid and terminates the session with the client. Comparing this procedure with a physical store purchase, it would be equivalent to a store security officer recognizing some thief and preventing him from entering the store. But if the client certificate is valid the merchant sends the client's X.509 certificate to the bank (credit card issuer) for *credit card number verification.* The client's encrypted credit card number is stored in a X.509 certificate as a private extension, as described in Section 6.2.1. Indeed this step could be thought of as the merchant asking the bank: Is the account specified in the X.509 certificate private extension chargeable?

Step 5. The bank checks the X.509 certificate received from the client. The validity check includes:

- if the certificate comes from a trusted certificate authority,
- if the certificate has not expired,
- if the certificate is not in a CRL, and
- if the certificate has the extended private extensions described in Section 6.2.1.

If the certificate is valid in this context, the bank now checks the account specified in the X.509 extension. If the account is blocked or overdrawn the bank sends negative response to the merchant. A predefined set of returns codes could be assigned to each possible state of the client account in order to communicate the consumer's account state. If client X.509 certificate has passed the second check (the accounts exist and is chargeable), the bank sends a special string which is denoted as the *transaction number* (TN), which is a random number. To each transaction number the bank associates another two bit flags denoted as *requested* and *used.* All these three values, *TN, requested and used,* are then associated with the user account. When the bank sends a TN to the merchant, it sets its associated *request* flag to true (1 binary). In this way the bank eliminates replay attacks, if by any case a dishonest merchant wants to replay the client charge process. The bank encrypts the TN with the merchant's public key and sends it over the Internet.

Step 6. The merchant evaluates the response from the bank, decrypting it with its private key. If the response is negative the merchant terminates the session with the client. If the response from the bank is positive it must contain the *transaction number* (TN) generated by the bank. The merchant formats the response to the user purchase order made in step 3 as in Figure 8.3.

Time	Name	Amount	Client Name	Account Encrypted	Hash (Goods)	TN

Figure 8.3: Merchant response format

In Figure 8.3 *Time* represents the time at the merchant server, *Name* represents the merchant official name (as usually represented in credit card transactions). *Amount* represents the value that the merchant wishes to charge the user for the goods. *Client's name* and *Account* are taken from the client certificate. In order to achieve user privacy (the merchant should not know the credit card number and the bank should not know about the goods) in the transaction the hash value calculated over goods $H = h(Goods)$ is inserted, where SHA-1 could be used as the hash algorithm (see Section 2.2.3 about hash algorithms). This approach presented here, with hash values, is the same as in the SET protocol [Has01, SET97b]. All these data are then encrypted with the user's public key and sent over the Internet to the client. The merchant saves the order for the later shipment process.

Step 7. The client decrypts the message with his private key and then digitally signs the response received from merchant (see Figure 8.4). With the help of the hash value over goods the client has the possibility to check if he is charged for goods he requested.

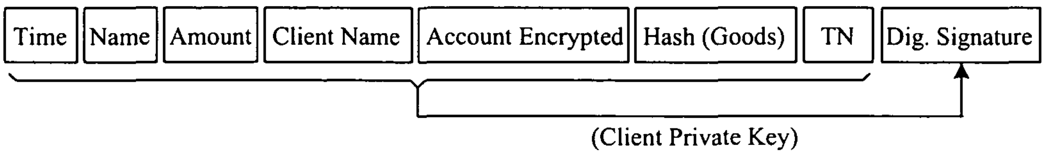


Figure 8.4: User signs digitally the request

In this case the digital signature has a double role. In general (see Section 2.2.4 about digital signatures) it first ensures the receiver that data has been not changed in transit, and second it guarantees the receiver (merchant and bank) about the ownership of a private key, whose counterpart (public key) is stated in the X.509 certificate. It ensures the bank that the user has triggered (initiated) the transaction rather than some malicious user. The user should have the possibility to save (or email) the signed request. This would be equivalent to receiving a copy of the hand-signed coupon in a physical store (as described in Section 4.1). The client encrypts the message with the merchants public key and sends it over the Internet.

Step 8. The merchant receives the signed response, decrypts it with its private key and encrypts with the bank's (credit card issuer's) public key. In this step the merchant acts only as router. The message format is the same as in Step 7.

Step 9. The bank decrypts the message received from the merchant with its private key. It verifies the signature of the user request. The transaction number in the message must have the *request* flag set (1 binary) at the bank database, otherwise the merchant is trying to duplicate the transaction data. The bank now sets the *used* flag of the transaction number and from the bank's point of view the transaction is completed. The bank now generates an *authorization code* and formats the data as in Figure 8.5.

Time in Figure 8.5 represents the current time at the merchant. The authorization data includes also the *transaction number (TN)*. For non-repudiation reasons the bank digitally signs the response with the authorization code. All data are then encrypted with the merchant's public key and sent over Internet.

Step 10. Based on a positive authorization code, the merchant makes the goods or services available for shipment or download and receives the charged amount

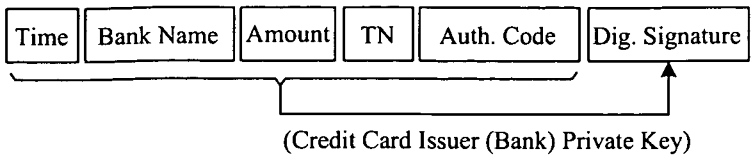


Figure 8.5: Credit card issuer digitally signs the authorization data

from the credit card issuer. The merchant informs the user of a successful completion of the transaction. The message is encrypted with client's public key.

8.1.2 Achieving non-repudiation over SSL

The approach presented in this section assumes that SSL communication is established between each of the two-participant groups, i.e consumer and merchant (group one) and merchant and bank (group two). Furthermore it is assumed that mutual authentication based on X.509 certificates is performed between participants.

Steps 1 to 8 are executed in a sequential manner. The format of data messages is the same as in Section 8.1.1. In this case there is no need for data encryption since encryption is performed by the SSL (see Section 2.6.1 about SSL), therefore two steps fewer are required than in Section 8.1.1. Figure 8.6 represents an overview of exchanged messages over an SSL session between client, merchant and bank.

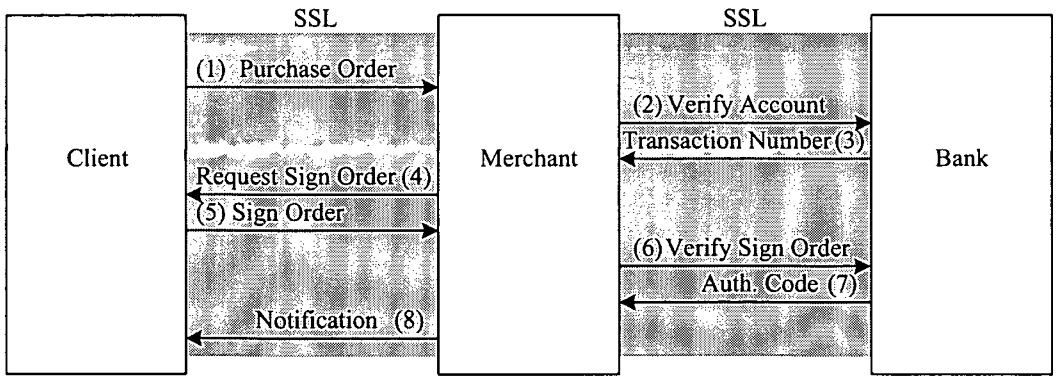


Figure 8.6: Message exchange protocol over SSL

The description of each step in Figure 8.6 is the same as in Section 8.1.1. Step 3 in Section 8.1.1 is equivalent to Step 1 here, Step 4 is equivalent to Step 2 here and so on. The gray boxes in Figure 8.6 indicate that communication between each two partners is SSL encrypted.

8.1.3 Minimum exchange messages

The approach presented in this paragraph presents a minimum of exchange messages between client, merchant and bank. In both previous approaches, the client completes the transaction in two steps (like in a physical store, see Section 4.1): ordering and then signing.

The approach presented in this step unifies this two steps in one. This approach also eliminates the need for a transaction number (TN) from the bank. The transaction number in this case is generated by the client, thus enabling the signing of transaction data from the first step, as in Figure 8.6. Signing the transaction data in the first step enables the bank and merchant to verify the origin and correctness of the transaction data in the second step, whereas in Section 8.1.1 and 8.1.2 this is done in steps 8 and 6 respectively (see Figure 8.1 and 8.6).

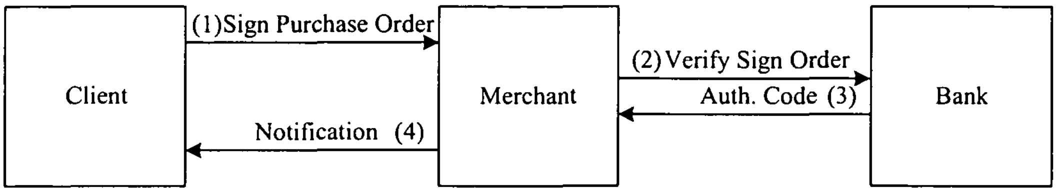


Figure 8.7: Minimum exchange messages

Figure 8.7 represents a flow overview of the minimally exchanged message sets between client, merchant and bank.

The flow and format of messages are as follows:

Step 1. The user prepares the purchase order; he generates a transaction number (which is a random number used against replay attacks). The purchase format is presented in Figure 8.8.

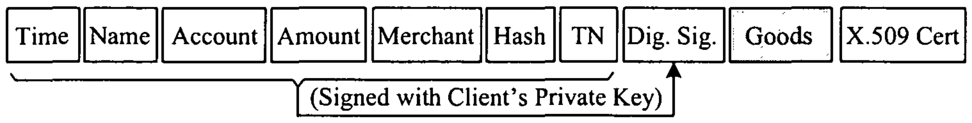


Figure 8.8: Purchase order format

The *Time* in Figure 8.8 represents the transaction time at the client. *Name* and *Account* are values taken from the client's X.509 certificate. *Amount* represents the charge amount of the transaction. *Merchant* is the merchant name or ID (as is usual in credit card transactions). The *Hash* is the hash value over the user selected goods $Hash = h(Goods)$, where as the hash algorithm SHA-1 could be used (see Section 2.2.3 about hash algorithms). The *TN* represents the transaction number generated by the client. *Dig.Sig.* represents the digital signature over the previous data. The digital signature ensures the merchant and bank that the client has initiated the transaction and that he is the owner of the corresponding private key. *Goods* represent the user selected goods, which must be readable by the merchant in order to complete the purchase. The client attaches his X.509 certificate, with encrypted credit card number in a private extension, to the message. If the message exchange is done via an open channel, like the Internet, the client should encrypt the message with the merchant's public key.

Step 2. The merchant checks the client certificate for the following conditions:

- if the certificate is issued by a trustworthy authority,
- if the certificate life time has not expired, and
- if the certificate is not in a CRL.

If the certificate check fails, the merchant marks it as invalid and terminates the session with the client. The merchant also has the possibility to check the digital signature, i.e. verifying that the client owns the corresponding private key. The merchant extracts the list of *Goods* from the received message in order to assure user privacy and routes the message (without list of goods) to the bank for verification. The same approach is applied in the SET protocol, where the merchant forwards all elements except the order information (goods) [Has01].

Step 3. The bank checks the client's X.509 certificate against the following criteria:

- if the certificate comes from a trusted certificate authority,
- if the certificate has not expired,
- the certificate is not in CRL, and
- if the certificate has the extended private extensions, as described in Section 6.2.1.

If the certificate satisfies all these criteria, the bank verifies the digital signature, assuring that the transaction is originated by client. Afterwards the bank checks the account specified in the X.509 extension. If the account is blocked or overdrawn, the bank sends a negative response to the merchant. If the client account (specified in the message and X.509 certificate) is chargeable the bank formats the response as presented in Figure 8.9.

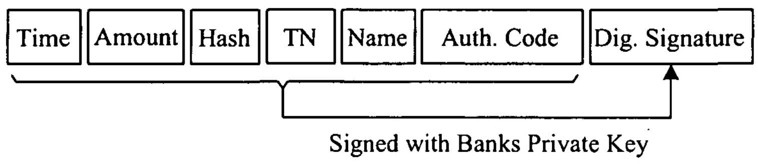


Figure 8.9: Authorization code format from credit card issuer

Name in Figure 8.9 denotes the bank name. The response to the merchant is signed by bank to achieve non-repudiation.

Step 4. Based on a positive authorization code, the merchant makes the goods or services available for shipment or download and receives the charged amount from the credit card issuer. The merchant informs the user of a successfully completed transaction.

The protocol presented in this section can be applied over HTTP or HTTPS. In the case of HTTP, messages are then encrypted with the respective sender's public key. In the case where a secure network exists (banking network, virtual private network (VPN)) between the merchant and the bank, the message flow in Figure 8.7 can be used without any change to the message format.

8.2 SET vs. proposed approaches

At first glance the proposed approaches in this thesis have several similarities with SET, such as:

- using X.509 certificates for credit card holders, merchant, and credit card issuer (bank or payment gateway),i.e. all participants possess an X.509 certificate,
- using digital signature for credit card holder verification,
- using extra X.509 private extensions, and
- encrypting credit card number.

Approaches proposed in this thesis differ from SET in the following respects:

- SET requires the same root certification authority (CA) for all participants (as described in Section 2.5.3). SET participants (users, merchants and banks) get their X.509 certificates from a local (geo-political) registration authority, which is authorized by the SET Brand Authority, which is in turn authorized by the SET root CA, thus making a 3-4 level hierarchy of trust. Each certificate is linked to the authority that has issued it, up to the root CA [SET97a]. We do not require that X.509 certificates be linked to the same root CA. The issued X.509 certificates must be issued by a trustworthy CA. It is the responsibility of the issuer, merchant and bank to accept or reject the root certificate from the CA of the other participants. This means that the user's root certificate from the issuing CA must be accepted by the merchant and bank as trustworthy. Comparing this approach with the passport example (see Section 2.3.2), SET requires that all "passports"(certificates) are issued by the same authority. In our our approach "passports"(certificates) by different authorities.
- The encrypted credit card number is stored in a certificate as a private extension with specific OID. SET uses 6 extra private extensions [SET97b] (so called SET extensions) and none of them is used to store credit card information. In our approach the number of extensions depends on the user profile (i.e on the number of credit cards the user possesses, giro accounts etc.).
- SET marks two extensions as *critical*, which means that applications that do not know how to process the X.509 certificate must mark certificate as invalid. In our approach all extensions are marked as *not critical*.
- In our approach we use a different OID for each credit card issuer.
- In our approach an X.509 certificate can hold all possible user credit card numbers (and other properties also), thus *one certificate - all cards*.
- SET predefines a protocol on how participants should obtain their certificates from a CA. We do not specify a special message set that participants in a transaction should use to obtain their certificates from a CA. They should use their standard flow (online or offline). The difference is that with our approach the CA must fill in extra fields (credit card number, giro accounts, insurance

number etc.) and include these fields in the certificate. It is obvious that at the time of X.509 certificate issuing the CA must have the corresponding public keys of other entities (i.e. public key of: credit card issuer, bank, insurance company, post etc.).

- We propose to use smartcards for certificate storage. In this form it would be possible to have all the user's bank information (credit card and giro account numbers) in one smartcard. Storing certificates in smartcard enhances user mobility.

We want to emphasize once more that due to use of hash algorithms the bank does not know about order information (goods) and due to encryption of the credit card number in the certificate the merchant does not know the credit card number. This increases the user's privacy.

The hash values in the message flows proposed in Section 8.1 are used for user traceability, to link charges and order information. The hash value, transaction time (time stamp) and transaction number protect the transaction flow against replay attacks.

8.3 Identity certificates vs. attribute certificates

As already mentioned in Section 2.3.2 an identity certificate (public key certificate) binds the public key with the name (identity) of certificate holder. Public key certificates are generated, distributed, and potentially revoked by CAs.

Attribute certificates bind the characteristics (attributes) of an entity with the holder of an identity certificate (as stated in the *holder* field of the attribute certificate) [Bal03]. Thus the certification of an attribute is dependent on both a certification authority (CA) and attribute authority (AA). In practice attribute certificates are rarely used. The reasons why attribute certificates are not yet widely used today are [Pin03]:

- attribute certificate (AC) definition is not appropriate,
- no management rules exists for ACs, and
- because ACs are defined in ASN.1.

Identity and attribute certificates have similar data structures, as presented in Figure 8.10, in which different fields are denoted with gray color. Both data structures are digitally signed by an issuer authority.

The main differences between identity and attribute certificates are [FH02]:

- attribute certificates do not hold a public key,
- attribute certificates describe what the holder of the certificate is or is not allowed to do,

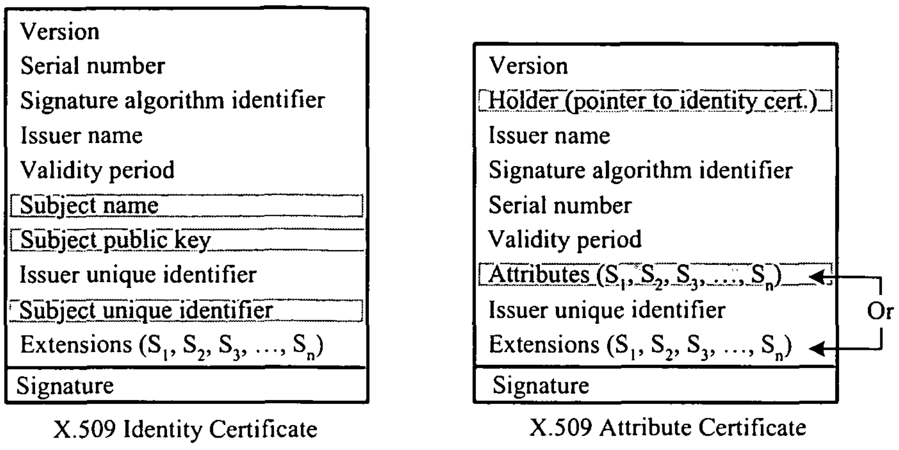

Version
Serial number
Signature algorithm identifier
Issuer name
Validity period
Subject name
Subject public key
Issuer unique identifier
Subject unique identifier
Extensions (S_1, S_2, S_3, ..., S_n)
Signature
X.509 Identity Certificate
Version
Holder (pointer to identity cert.)
Issuer name
Signature algorithm identifier
Serial number
Validity period
Attributes (S_1, S_2, S_3, ..., S_n)
Issuer unique identifier
Extensions (S_1, S_2, S_3, ..., S_n)
Signature
Or
X.509 Attribute Certificate

Figure 8.10: Comparing identity certificates with attribute certificates

- identity certificates depend only on CA and attribute certificates depend on both CA and AA, and
- attribute certificates have usually shorter lifetimes than identity certificates (user role and rights are changed frequently and therefore have short lifetime).

In general, a X.509 v3 public key certificate can also carry authorization information about its owner. This information can be encoded in one of the X.509 v3 extension fields. Defining attributes as a private extension is a good decision in the case where [Ali00, Pin03]:

- the CA entity has knowledge of the attributes,
- the life of the attribute matches the life of identity certificate, and
- the application understands the extension. Private extensions must be marked as not critical, so applications that do not understand then can ignore the extension.

Comparing *Approach 1* presented in Section 6.2 and *Approach 2* presented in Section 7.3 reduces to comparing the advantages of identity certificates over attribute certificates.

Using identity certificates has the following advantages over attribute certificates:

- compactness, i.e. all user attributes (properties) are in one certificate (in one smartcard), and
- less complexity in the system, since the user has to take care of only one certificate.

The advantages of using attribute certificates over identity certificates are [Ali00]:

- Application specific information is removed from the identity certificate.

- The attribute certificate can be issued by the organization that controls the "attributes". In payment protocols this is the bank that authorizes the user to make an online payment.
- Using short lived attribute certificates or single-use certificates as proposed in Section 7.3.3 eliminates the need for a CRL, because there is no need to revoke expired certificates. This reduces the complexity of the PKI system.
- The user could have many attribute certificates linked to a single identity certificate. In this sense attribute certificates allow more flexibility.

We recommend identity certificates for storing extended user properties, as presented in Approach 1 in Section 6.2, because of the compactness i.e. all user properties (different credit card numbers, giro account number, address, insurance number etc.) are stored in one identity certificate, thus *many properties (cards) one certificate.*

Chapter 9

Case Study – Virtual Web Drive

The approach presented in Section 6.2 has been implemented in a test application with Web Services. The application is called *Virtual Web Drive*, where a user can buy online a virtual space. The payment for the service is made via a X.509 certificate stored in a smartcard. Afterwards the user uses the virtual drive for storing securely different files in a encrypted form. The login to the Virtual Web Drive is also made with X.509 certificate, which is stored in the smartcard.

The *Virtual Web Drive* application uses the subscriber principle, in which users must first subscribe to the Web Service provider for the desired service for some defined time period (6 months, 12 months, etc.) before using that service.

The basic idea of a *Virtual Web Drive* is taken from [MJ03] [1]. We have extended the application presented in [MJ03] by adding:

- a payment process (subscribing to the Web Service) with X.509 certificate,
- logging into the virtual web drive with a X.509 certificate,
- smartcard support, and
- new features were discussed through this chapter.

9.1 Application overview

The Virtual Web Drive application allows a user to upload and retrieve documents securely from a specified server. The documents on the server are saved in a encrypted form. For performance reasons we have applied the same techniques as in PGP (see Section 2.3.3). Before uploading, user files are encrypted with symmetric methods (3DES, Rijndael, etc.) as is presented in Figure 9.1. The key used for file encryption is a random session key which is stored together with the file. The session key is encrypted with the user's public key. The client's asymmetric keys, public and private, are stored in a smartcard and his public key is also available in X.509 certificate.

[1]This is an comprehensive book about .NET cryptography.

In Figure 9.1 the system architecture of Virtual Web Drive is presented. During the download process the client re-constructs the session key by decrypting it with his own private key. With the session key the client decrypts the file to its original state, i.e. in plain text. Hash values are added to the file for ensuring data integrity while the file resides at the server.

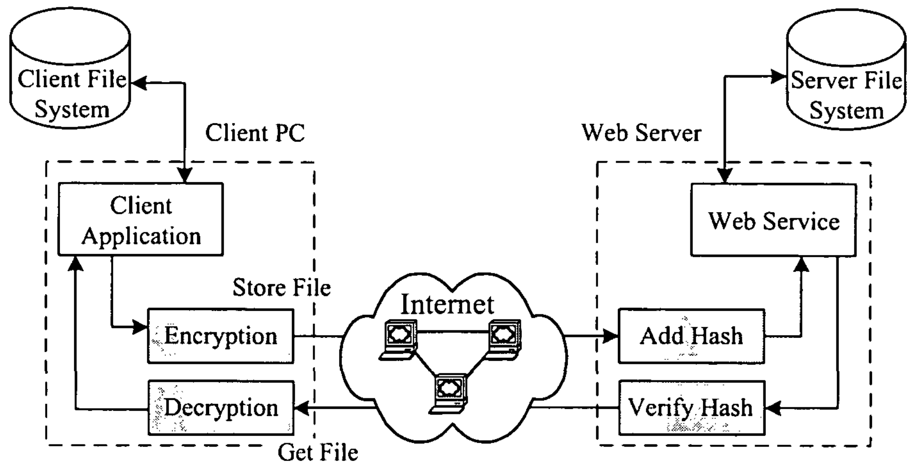


Figure 9.1: Virtual Web Drive architecture

User information is stored in a back-end database and uploaded user files are stored on the server's hard drives. For security reasons, original file names are substituted with random values in the form of globally unique identifiers (GUID) [2].

The main goal of implementing this application was to demonstrate the usability of the new private (encrypted) extensions in X.509 certificates. The private extensions are used for the payment process. The X.509 certificate and the private key, its counterpart, are stored in a smartcard. Different encryption techniques used in the application serve to make it more "realistic" and representative of commercial security applications.

The Virtual Web Drive has the following security properties:

- By requesting virtual space on the server (merchant) the payment process is made with X.509 certificates with encrypted extensions.
- Before signing the request, the client has the possibility to view the transaction data in a so called "*secure viewer*" and after signing to save a copy of his signed request.
- To upload or download files from the server clients must login with a X.509 certificate.
- Communication between the client and Web Service provider is not encrypted, but the files that are uploaded are encrypted with a randomly generated session key.

[2]A GUID is a random 16 byte integer that can be used over different computers and networks as a unique identifier [MSD01c].

- Due to the login feature, clients only have access to their uploaded files.
- Additionally "*a view mode*" is provided for the Web Service in which clients can login with username and password. In this mode clients can just view their uploaded files [3].

9.2 Application components

The Virtual Web Drive components are presented in Figure 9.2. The merchant bank is not directly involved in the payment transaction, and is drawn in gray in Figure 9.2. In real applications the credit card issuer (bank) could be substituted with any payment gateway processor.

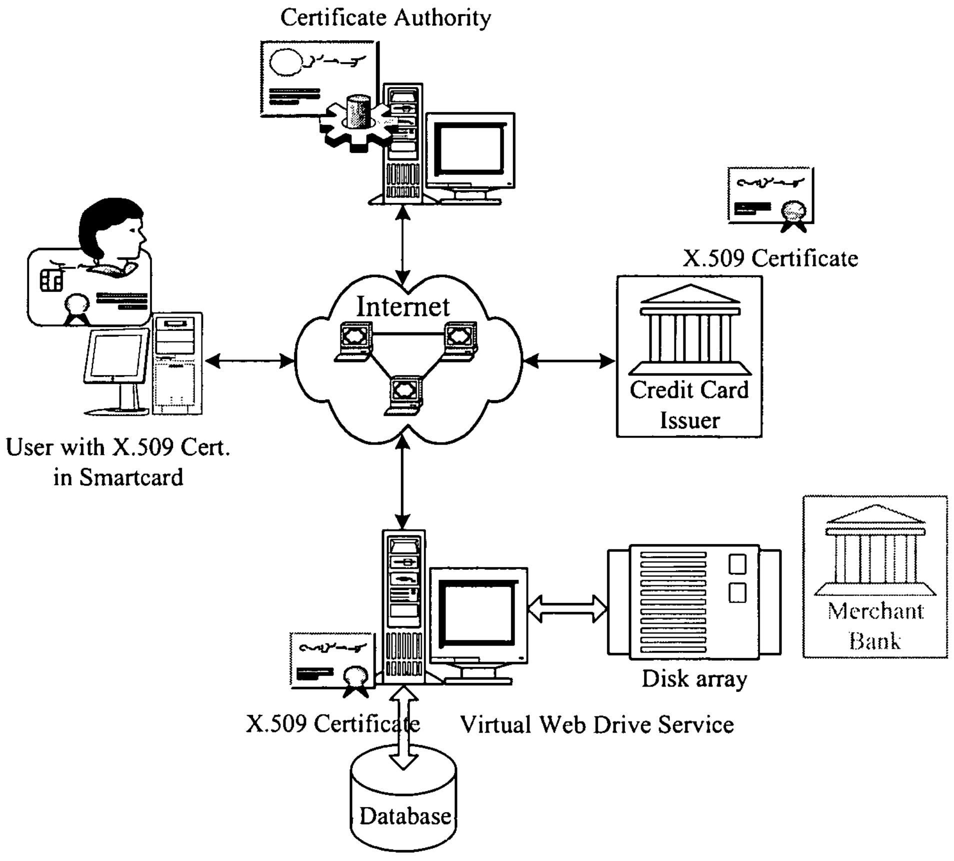


Figure 9.2: Virtual Web Drive application components

The Virtual Web Drive service (merchant) and credit card issuer are implemented as Web Services. The client does not communicate directly with the credit card service. It is the merchant that uses the services of the credit card issuer bank (like checking the validation of the clients credit card number). The client application only uses the Web Services offered by Virtual Web Drive for storing and retrieving files.

[3]Since in this mode the client has no access to his public and private key, he can neither make an upload (encrypting with his public key) nor download (decrypting with his private key) his files.

9.2.1 .NET and Java technology

The technology used in the Virtual Web Drive application is based exclusively on Microsoft products, with the exception of the application used for generating X.509 certificates, OpenSSL [4].

For building the client application and Web Services we have used the .NET technology and its C# language, although the same goal could be achieved with the Java language.

A comparison of .NET and Java architectures is presented in Figure 9.3. Java and .NET have very common technical features such as: both support true object oriented (OO) languages with single-inheritance and garbage collection. Both have huge libraries for supporting different functions for the user interface (UI), file access, security, web access, remoting, database access etc. The main difference is that Java is a single language with many deployment choices and .NET supports many programming languages with a single choice of deployment, i.e. only Windows platforms [Cab02] (see Figure 9.3). Microsoft states that in the near future they will support .NET in platforms other than Windows [RAC+02].

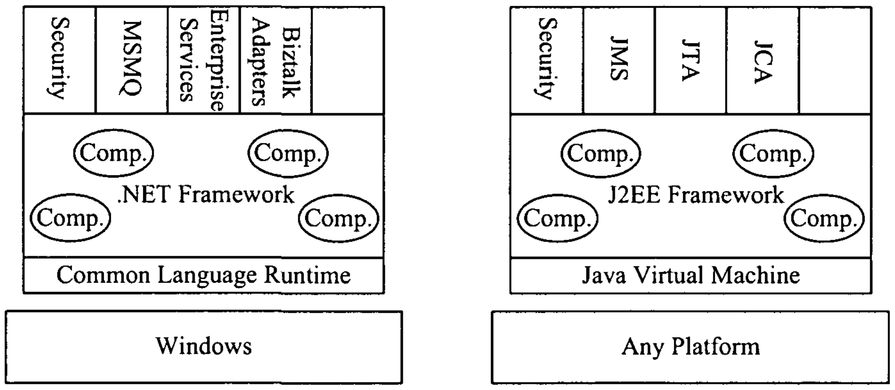


Figure 9.3: .NET and Java technical architectures [LS03]

Microsoft and Java solutions are estimated to underpin 80% or more of new application development projects by 2005, as is presented in Figure 9.4. Java will dominate in the cases where the application will be deployed on different hardware and software platforms. .NET will dominate in Windows systems due to its rapid application development (RAD) support. Neither .NET nor Java will dominate alone the market, both they will likely both remain in the market and there will be no clear winner with enterprises continuing to use both technologies. Figure 9.4 presents the market development and a forecast of Microsoft's .NET and Java platforms over the next few years. For companies and people that are not already involved with any development programming models, Figure 9.4 clearly puts the Java technology as preferred one [Dri02].

Another comparison, between Microsoft's .NET and IBM Web Sphere 4.0 is presented in [Mic01a] based on line of codes (usability) and deploying costs. This claims .NET is preferable, due to its RAD properties.

[4]http://www.openssl.org

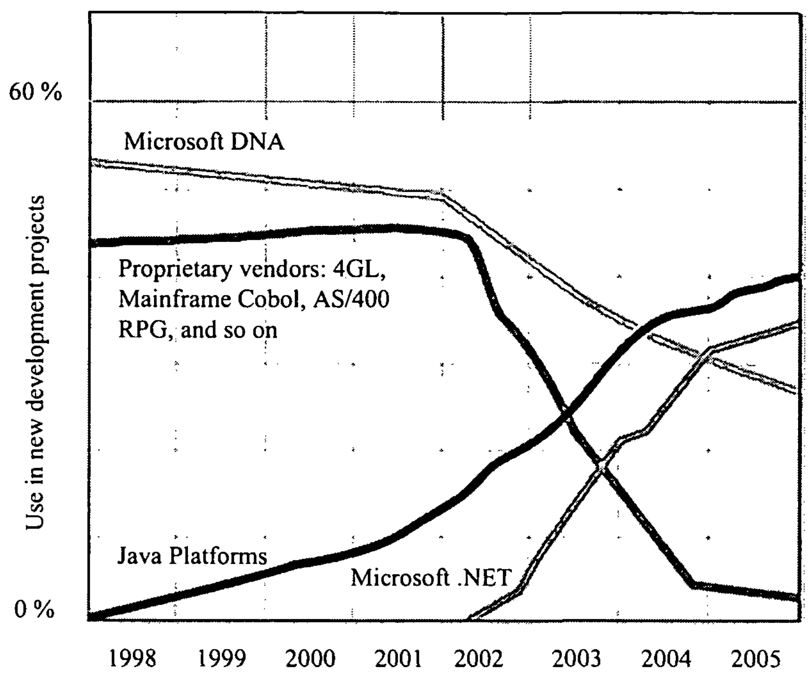


Figure 9.4: E-business application development programming models [Dri02]

Microsoft's SQL Server is used as the database server in the Virtual Web Drive application. Alternatively another database server could be used like Oracle, MS Access etc. There are two databases in the system. The merchant uses a database for storing online transactions with its clients and user-file relationship. The second database is located at the credit card issuer (bank), as presented in Figure 9.1. The credit card issuer's database stores clients personal information, credit card numbers, and clients transactions with different merchants.

Although the approach used in this test application seems to be Microsoft centric, the main reason why we decided to use Microsoft technology is that we have experience with Microsoft's products and resources and licences from Microsoft.

9.2.2 Certificate authority

A certification authority (CA) is responsible for issuing X.509 certificates to participants involved in an online transaction. Each participant in an online transaction must have an X.509 certificate. In order to insert the new private extensions into X.509 certificate the CA must have the corresponding public keys, respectively the X.509 certificates.

In the Virtual Web Drive application we have used only one private extension, namely the value of an American Express Credit Card [5] which has an object identifier (see Section 6.2.3 and Table 6.2) OID = 1.3.6.1.4.15601.2. Thus, each client's X.509 certificate must contain the above OID, otherwise the Virtual Web Drive application

[5]The value of the OID used in this example is just a fictitious, invented value. The real value must be registered at IANA in order to be world wide unique.

will reject the X.509 certificate as invalid for the payment process. Under this OID the client's American Express (AMEX) Credit Card Number is stored encrypted with the public key of AMEX. Mathematically, the OID value stored in the X.509 certificate is:

$$OID = E_{RSA}(AMEX_{PublicKey}, CreditCardNumber) \tag{9.1}$$

where E_{RSA} denotes the RSA encryption. Some test users were created in the credit card issuer (AMEX) database with dummy credit card numbers.

The challenge in this work was to generate a new X.509 certificate. Microsoft, with its Windows 2000 Advanced Server, provides a certificate authority which is very easy to configure and use. But there are only standard templates provided, there are no places where CA administrator can give information about extra private extensions. The next attempt was to write software that creates a PKSC#10 request [6] and sends this request to a CA server. For this purpose we used the Microsoft ActiveX control *CEnroll* through its exposed interfaces. The Certificate Enrollment Control, known as the CEnroll object, is a very powerful object with a rich reach collection of methods and properties regarding the certificate enrollment process. It can be accessed through four interfaces. The CEnroll object is used mostly in automation languages [MSD01b]. Unfortunately this did not help either, because CEnroll do not offer any interface for inserting private extensions into certificate request.

After we had tried all Microsoft solutions, we decided to use the OpenSSL tool. OpenSSL is a project driven by volunteer programmers to develop an open source toolkit for implementing the SSL, TLS and general purpose cryptographic libraries. OpenSSL is based on the SSLeay library developed by Eric A. Young and Tim J. Hudson. OpenSSL toolkit is free to use for commercial and non-commercial purposes [Ope03a]. OpenSSL is a command line driven tool, therefore the challenge was changing the configuration file and writing a script [7] for generating a X.509 certificate.

Figure 9.5 describes a part of the OpenSSL configuration file, which has the format of a standard INI file, with predefined sections, user defined sections, and values. The configuration file contains the new private extensions. In this case it contains:

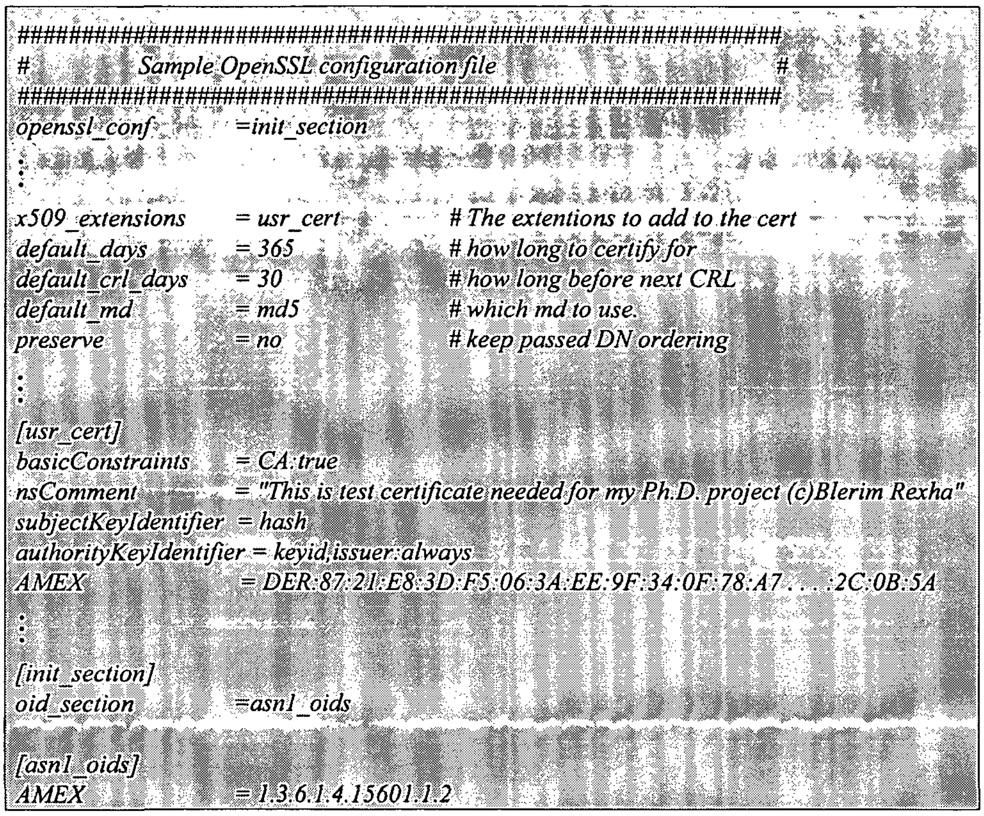
- nsComment extension – which is a Netscape comment and can contain any text. In this case it is used only to denote that this is a test certificate.
- AMEX extension – which has the OID = 1.3.6.1.4.15601.1.2 and a value that contains the client's credit card number in encrypted form and presented in hexadecimal notation, see Figure 9.6. This value (in this form) is seen by everyone who receives the clients X.509 certificate. This extension value is different for each client, since each client has a different credit card number.

[6]Public Key Cryptography Standard (PKCS)#10 describes syntax for a request for certification of a public key, a name, and possibly a set of attributes.

[7]Indeed we found a script example on the Internet, which was apparently written by "Artur Maj". We write "apparently", because we have no means of verifying the authorship through any X.509 certificate, digital signature or public key. And this thesis is about data security on the Internet. But in any case the example was very helpful, and the modified script worked fine.

The output of OpenSSL scripts is:

- a *pem* file, which contains the private key (1024-bit RSA) and certificate encoded in Base64 format. This file is encrypted with the password which is entered during the generation process, and
- the client's X.509 certificate with private extensions.



```
##################################################################
#             Sample OpenSSL configuration file                  #
##################################################################
openssl_conf          =init_section
:
x509_extensions       = usr_cert          # The extentions to add to the cert
default_days          = 365               # how long to certify for
default_crl_days      = 30                # how long before next CRL
default_md            = md5               # which md to use.
preserve              = no                # keep passed DN ordering
:
[usr_cert]
basicConstraints      = CA:true
nsComment             = "This is test certificate needed for my Ph.D. project (c)Blerim Rexha"
subjectKeyIdentifier  = hash
authorityKeyIdentifier = keyid,issuer:always
AMEX                  = DER:87:21:E8:3D:F5:06:3A:EE:9F:34:0F:78:A7. . . :2C:0B:5A
:
[init_section]
oid_section           =asn1_oids

[asn1_oids]
AMEX                  = 1.3.6.1.4.15601.1.2
```

Figure 9.5: OpenSSL configuration file

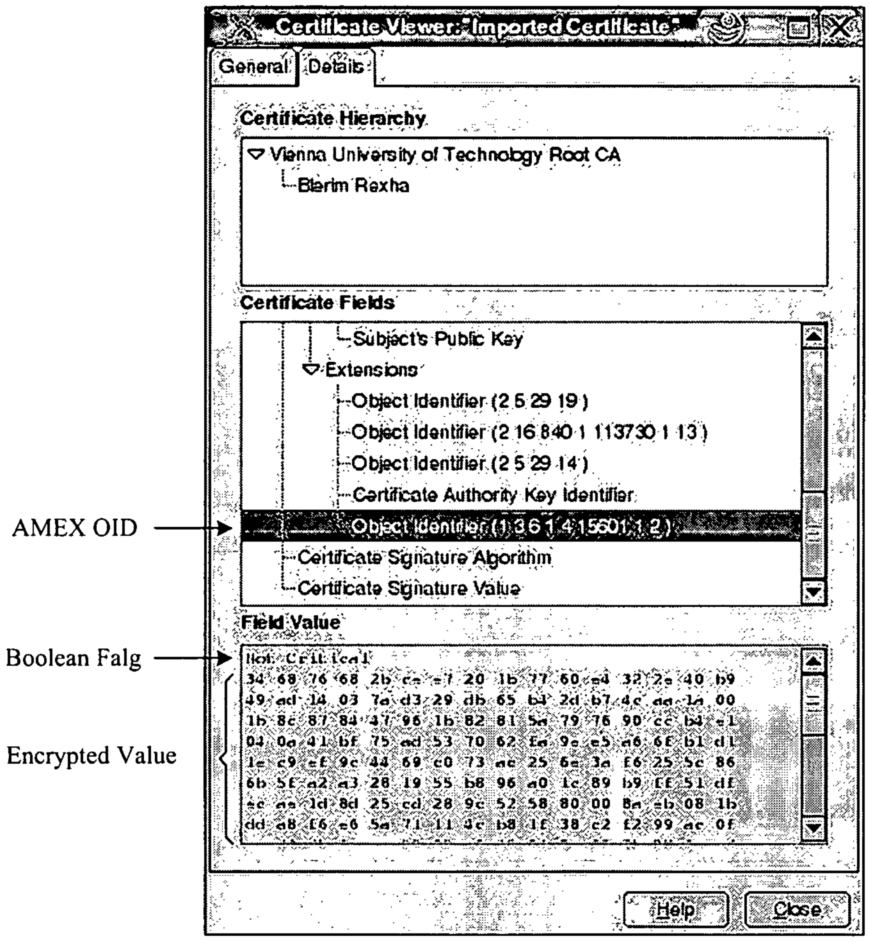
The *pem* file is very sensitive since it contains the client's private key. It must be saved in secure storage and should not be accessible to any one. *We have decided to store client's X.509 certificate and its corresponding private key in a smartcard.* The "PfxToSicrypt" tool [8] is used to import the X.509 certificate and private key into the smartcard, it supports only files of "*p12*" type, i.e. which have the PKSC#12 format. PKCS#12 is a standard for storing private keys and certificates and is used in Netscape and Microsoft Internet Explorer with their import and export options. To export certificate and private key from a *pem* file into PKCS#12 format the *pkcs12* function of the OpenSSL toolkit is used.

In the Virtual Web Drive application we only used one CA for the sole reason of not making the application more complicated. In the case that more CAs are used, each

[8] This tool is Siemens AG internal and supports importing a PKSC#12 file into Infineon's SICRYPT smartcards.

participant in the online transaction must trust the other CA. This means that each participant must install other CAs self-signed root certificates in its "*trusted root*" certificate store.

In Figure 9.6 is presented a X.509 v3 certificate with AMEX extension, viewed with Mozilla under the Linux operating system. Thus any receiver of a X.509 certificate can view the encrypted value of the credit card number, but only the credit card issuer (in this case the AMEX) can link it with the real credit card number, just decrypting it with its private key.



Certificate Hierarchy

```
▽ Vienna University of Technology Root CA
  └─Blerim Rexha
```

Certificate Fields

```
  │  └─Subject's Public Key
  ▽ Extensions
     ├─Object Identifier (2 5 29 19 )
     ├─Object Identifier (2 16 840 1 113730 1 13 )
     ├─Object Identifier (2 5 29 14 )
     ├─Certificate Authority Key Identifier
     └─Object Identifier (1 3 6 1 4 1 5601 1 2 )
  ├─Certificate Signature Algorithm
  └─Certificate Signature Value
```

Field Value

```
Not Critical
34 68 76 68 2b ca e7 20 1b 77 60 e4 32 2e 40 b9
49 ad 14 03 7a d3 29 db 65 b4 2d b7 4c aa 1a 00
1b 8c 87 84 47 96 1b 82 81 5a 79 76 90 cc b4 e1
04 0a 41 bf 75 ad 53 70 62 fa 9e e5 a6 6f b1 d1
1e c9 ef 9c 44 69 c0 73 ac 25 6e 3a f6 25 5c 86
6b 5f a2 a3 28 19 55 b8 96 a0 1c 89 b9 ff 51 df
ec ae 1d 8d 25 cd 28 9c 52 58 80 00 8a eb 08 1b
dd a8 f6 e6 5a 71 11 4c b8 1f 38 c2 f2 99 ac 0f
```

Figure 9.6: A X.509 certificate with AMEX extension

9.2.3 Merchant

The merchant or Web Service provider provides (sells) web space to clients. The merchant has its X.509 certificate issued by a trustworthy CA in order to establish a secure channel with its clients.

In this application the merchant is implemented as a Web Service, offering a public interface to its consumers; it also uses Web Services from the credit card issuer. The interface that the merchant offers to its clients could be divided into three categories:

- Always accessible – the main functions in this group are always accessible. The candidates for this group are functions which return the merchants X.509 certificate and enable login operations.

- Payment function – is used by the client only once (per month or year, on a subscriber basis), for requesting web space. This function requires that the client has a X.509 certificate with credit card number encrypted as a private extension.

- Only after successfully login – are other functions available for storing, retrieving and deleting files at merchants server.

The virtual merchants uses a SQL Server as its database to store online transactions with clients and bank (credit card issuer) and to store client-file relationship. User files are stored in disk arrays (see Figure 9.2).

The merchant uses the Web Services offered by the credit card issuer for credit card number validation; thus the merchant's Web Services internally invoke another Web Service.

When the merchant gets a request for web space, it checks the client's account stated in the X.509 certificate while redirecting it to the credit card issuer (using Web Services exposed by credit card issuer company), i.e asking the credit card issuer: Is the account stated in the X.509 certificate chargeable? In the case that account exists and is chargeable, the merchant gets from the credit card issuer a transaction number, which identifies the current transaction. It is implemented the flow in 10 steps, as presented in Section 8.1.1. The merchant prepares the data that the user must digitally sign in order to prove that he is the owner of public key stated in the X.509 certificate.

The merchant Web Service is programmed in C# and .NET. It uses three cryptographic libraries:

- the standard .NET cryptographic library [9],

- the Web Service Enhancement (WSE) .NET cryptographic library [10], and

- the CAPICOM library [11].

The reason why we used all three libraries is that none of them alone provides full functionality for the application. With functions provided by CAPICOM the validity of the X.509 certificate is checked. For iterating through certificates in user store we have used WSE cryptographic library and for extracting the public key from the X.509 certificate we have used the standard .NET cryptographic library.

[9]This library is located at: System.Security.Cryptography namespace.

[10]This library is located at: Microsoft.Web.Services.Security.X509 namespace.

[11]CAPICOM is an ActiveX control that provides a COM interface to Microsoft's CryptoAPI function library

9.2.4 Credit card issuer

The credit card issuer (bank) issues the credit cards to users and provides a Web Services for credit card number validation for merchants. The credit card issuer has its X.509 certificate issued by a trustworthy CA in order to establish a secure channel [12] with merchants.

The credit card issuer generates 10 transaction numbers (TN) for each client in advance. TN are random system generated GUID-s. Each TN has two flags associated with it:

- Requested flag, which denotes that this transaction number is requested by some merchant and must no longer used.
- Used flag, which denotes that one transaction is completed and the amount of money associated with the TN is allocated to the merchant involved in the transaction.

This TN is generated if the protocol flow between client, Web Service provider (merchant), and credit card issuer (bank) is implemented as the ten steps in Section 8.1.1. If the TN is generated at the client (see Section 8.1.3) it has no flags associated with it. The credit card issuer uses an SQL Server for storing the client's credit card numbers and their related information, such as TN and related transactions.

The credit card issuer Web Service has roughly two interfaces for merchants:

- Credit card number validation, whereby the TN is returned in a positive validation.
- Authorization code, which ensures the merchant that it will get the money from the client's account. The authorization code is signed digitally by the credit card issuer.

The credit card issuer Web Service is programmed in C# and .NET and like merchant's Web Services, uses all three cryptographic libraries. The credit card issuer Web Services checks the client certificate for: if it is issued by a trustworthy CA, if it is valid (expiration and content validity) and if it contains the necessary extensions. If the certificate does not fulfil all of the above criteria it is classified as an invalid X.509 certificate and the transaction between the merchant and credit card issuer is broken. It must be emphasized that client (application) never communicates with the Web Services of the credit card issuer.

9.2.5 Client

The client is implemented as a stand alone application that uses the Web Services exposed by merchant. The client application is implemented in .NET and the C# programming language. For performing cryptographic operations clients use all cryptographic libraries exposed by .NET including CAPICOM.

[12] A secure channel does not necessary to be established over Internet, any existing banking network can be used also.

The client uses a X.509 certificate with encrypted account information for the payment process. The X.509 certificate is stored in a smartcard. As a smartcard is used an Infineon [13] security controller chip of the SLE66Cxx family with a Secure Operating System (SCOS) for smartcard based applications. This type of smartcard is part of the SICRYPT product family of Infineon Technologies. Cryptographic services of the SICRYPT smartcards are supported through a cryptographic library known as cryptographic service provider (CSP) (see Figure 2.21). A CSP is a Microsoft Windows PC based software component responsible for creating keys, storing them in different locations (memory, disks, smartcards), destroying them and using them to perform a variety of cryptographic operations. The CSP functionality is wrapped by Microsoft's CryptoAPI library. The Infineon SICRYPT smartcard CSP is already included in the Windows XP setup [Inf03a].

The SICRYPT file structure is presented in Figure 6.4 and memory access to SICRYPT smartcard is protected with two PINs, a so called global (administrator) and user (application) PIN. Each time a function wants to use the private key, stored in $EF_{KeyPair}$ (see Figure 6.4) through CryptoAPI a user is asked about their PIN. In this way the client can install his certificate on any PC, even on a publicly accessible PC since it is only the X.509 certificate installed in system – the private key never leaves the smartcard. We have used the "Sicrypt Card Admin" [14] tool from Infineon Technologies to install a X.509 certificate from a smartcard to a PC.

In the client application the payment process between client and merchant starts when the client selects his X.509 certificate store. The CAPICOM library supports five X.509 certificate stores [MSD01b]:

- *Memory store*, is used to perform fast certificate operations. Any changes in the contents of the store are not saved for later use.
- *Machine store*, is mostly used for certificates that apply to PC, like IPSec certificates.
- *My store*, represents user's current store. In this store all user X.509 certificates are installed by default. If certificate is installed from smartcard the store "can remember" where the corresponding private key is stored.
- *Active Directory store*, is used to get user certificates from a central repository and any changes made to the store will be not saved.
- *Smartcard store*, in this case all smartcards that are inserted in system are queried for X.509 certificates.

"My store" and the "Smartcard store" are used in the client application as the X.509 certificate store, other stores are not supported. The client application queries the certificates in the selected certificate store and enables for selection only those certificates which contains the new OIDs ($OID = 1.3.6.1.4.15601.2$). Because under this OID client's credit card information is stored.

The client application has the possibility to trace all message exchanges with the Web Service provider (merchant) and before signing the payment request to view the

[13]http://www.infineon.com

[14]This tool is publicly available at http://www.sicrypt.com

payment data in a separate window, like a "*secure viewer*". We have implemented the flow presented in Section 8.1.1 in 10 steps, i.e. the transaction number is generated by the credit card issuer. When the client digitally signs the payment request at the same time he proves that he is the owner of the private key (whose public key counterpart is stated in the X.509 certificate) and that he has started the payment transaction.

The approach used for consuming Web Services is a subscriber basis. The client (consumer, user) first pays for services and afterwards uses them. In general the schemes that could be adopted for charging for Web Services are [Cla02]:

- Freeware – in this case the consumer is not charged for using Web Services. This schema is practicable for the trial period of a Web Service.
- Charge per call – in this case consumer is charged for using Web Services for each call. Usually consumer buys a number of calls to Web Service and during each call the credit counter is decremented.
- Subscription – a consumer pays for unlimited use over a particular period of time: one month, six months, one year etc. Once the period expires the consumer is asked to extend his subscription or it is extended automatically, based on agreement.

After the transaction is successfully completed, the merchant allocates the requested space (goods) and registers the client in its database. The merchant receives from the credit card issuer a signed authorization code about the client payment. After the payment transaction the client is able to login. After the login operation the client mainly uses these three web methods:

- SaveFile() – used to store files on the server. Before using this web method the client encrypts the file with the random session key. The session key is encrypted with the client's public key and is stored at the beginning of file. Thus, each file has a different encryption key. For file encryption we have used CAPICOM *EnvelopedData* object. As a session key we have used a triple DES key. This web method also implements a hash function over encrypted file content, as is presented in Figure 9.7. The hash value ensures the receiver before download that file content has not been changed during permanent storage at server. As the hash algorithm we have sued SHA1. Figure 9.7 presents the structure of the file as is stored on the server.

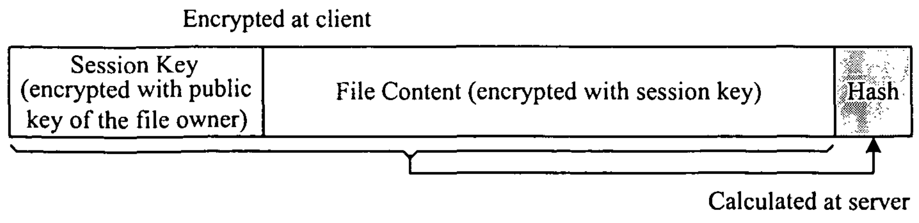

Encrypted at client
Session Key (encrypted with public key of the file owner)
File Content (encrypted with session key)
Hash
Calculated at server

Figure 9.7: Clients file structure on the server

- GetFile() – used to retrieve encrypted files from the server. The client receives a file in encrypted form, so the decryption is performed at the client side. It extracts the session key from the beginning of the file and decrypts it with his private key, see Figure 9.7. The file is afterwards decrypted with session key and stored on the client's local hard drive. This service implements a hash value check for ensuring file integrity.

- DeleteFile() – used for deleting files from the server permanently. In this case, just a delete command with file ID is sent to the server.

The encryption logic is encapsulated in a single assembly [15] that is used by the client and merchant application. The database read, write and update functions are also encapsulated in one assembly and used by the client, merchant and bank application.

Comparing the approach used in the client application with the security principles discussed in Chapter 5, we could assert that the client application uses *application layer security*. We could apply the SSL (transport layer) and IPSec (network layer) security mechanisms between client and merchant server, but it will not increase the security requirements of the client application. The main security requirement of the client application is to store files in encrypted form on the server. This is achieved with little effort while implementing security logic in the application layer.

9.3 Analyzing results

The implemented client, merchant and bank application were tested in two modes:

- Local mode, whereby all applications participants (client, merchant and credit card issuer) and their respective databases were located at local machine.

- Distributed mode, whereby each participant is located on a different machine and are connected only through the Internet. The client was located in the Siemens Intranet [16] and is behind a firewall and router, as in Figure 9.8. The merchant web server is located in DMZ (demilitarized zone), i.e. is protected from the outside world through a web router. As a web router we have used a Netgear [17] Web Safe Router RP614v2, which is suited to SOHO [18] networks and supports the notion of DMZ. The bank web server is directly connected to the Internet (see Figure 9.8).

In both test modes Internet Information Server (IIS) is used as the web server and SQL Server is used as the database server. And in both cases we were able to make payment transactions and upload/download files.

We have perform a test in which a client tries to make a fraudulent payment and those frauds are detected. We have assumed that a malicious user during the X.509

[15]Assembly is the logical unit that contains compiled code targeted at .NET. An assembly is completely self describing logical unit which can contain executable code and library code [RAC+02]

[16]The Siemens Intranet presented in Figure 9.8 is just a simplified version of Siemens AG network infrastructure.

[17]http://www.netgear.com

[18]Small Office and Home Office

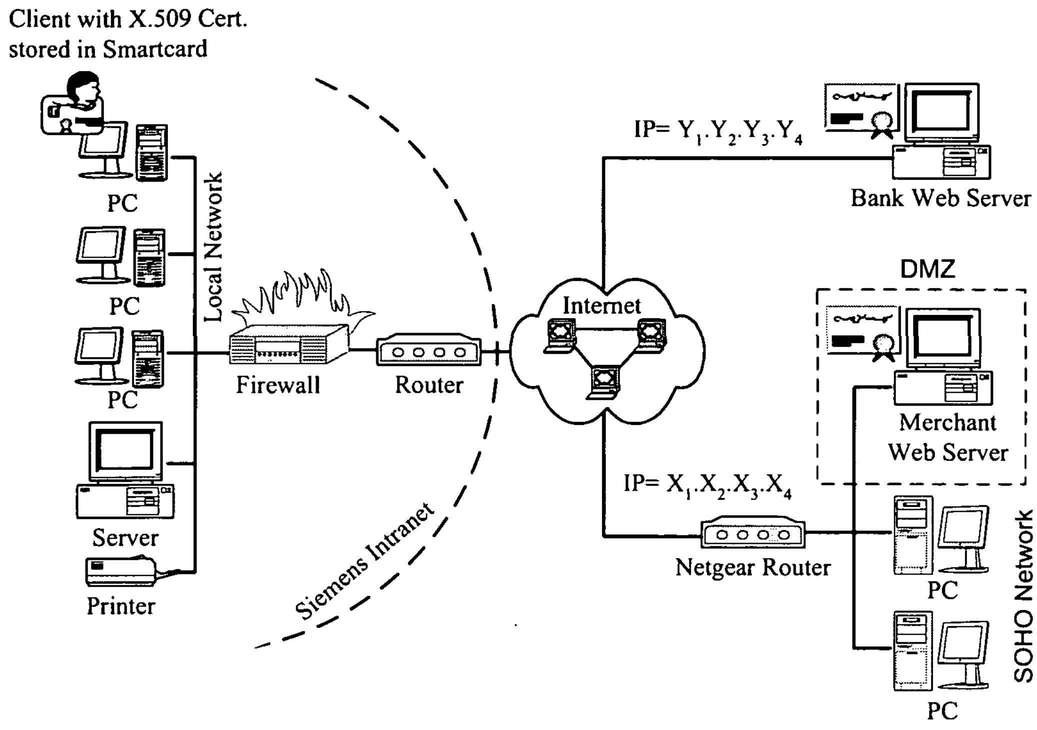


Figure 9.8: Client, merchant and bank in a possible distributed configuration

certificate issuing process has provided a credit card number which is either blocked by credit card issuer company (such is the case where the credit is stolen) or it does not exist at all. In such cases the credit issuer does not issue a transaction number, as in Section 8.1.1, but just terminates the session with the merchant stating that *the provided credit card number is fake.* The merchant then sends appropriate message to client and terminates the payment process. In a real application, these types of transactions (failures, frauds etc.) may be logged for later analysis.

Attempts at masquerading as the X.509 certificate owner are also successfully detected during digital signing of the payment request. The client application in this case throws an exception since the private key used for signing is missing. In cases where the message flow is implemented as described in Section 8.1.1 (in 10 steps) the missing private key is detected at step 7. This may be seen as a denial of service (DoS) attack for the real user, since for each transaction attempt a credit card issuer generates a transaction number. A masquerading user could be detected in step 2 (see Section 8.1.1) if the merchant applies the authentication process in this step. During this authentication process the merchant proves whether the user has a private key, whose public key is in the X.509 certificate. If the merchant uses different payment methods (like payment per check, per delivery etc.) then authentication in this step is excessive. In the case that the merchant uses only the payment method presented in this thesis, i.e. with encrypted account information in an X.509 certificate, then authentication in step 2 would improve security in the payment transaction and avoid the need of the credit issuer to generate transaction numbers.

Masquerading attacks can be detected if the exchange protocol described in Section 8.1.3 is used. In this protocol the payment request is signed in the first step. Thus the user proves in the first step that he is owner of the private key.

Attempts at using mismatched private/public key are detected by the credit card issuer, which proves the correctness of the digital signature in the payment transaction.

We have also observed that any changes that were made to files on the server (where they are permanently stored) were successfully detected. This feature is achieved with hash protection.

Chapter 10

Summary

Web Services act as a connection bridge between different applications in different platforms to perform diverse e-business functions. Through the use of Web Services, applications are not limited to presenting their information on the Internet, but they can make intelligent use of information accessed over the Internet [Wil01]. Using standard protocols such as XML, SOAP, WSDL and UDDI applications can communicate with each other. This enables faster and cheaper integration of different applications and new development models of distributed applications.

The approach used for securing and increasing privacy in Web Services studied in this thesis is based on X.509 certificates and PKI. The novelty of this approach is the encapsulation of user properties like credit card information, address, insurance number etc. in an encrypted form in X.509 private extensions. Thus, the new X.509 certificate carries extra information about user properties. *The approach can be extended to arbitrary properties, and is not limited to Web Services but potentially for any technology that uses X.509 certificates.*

Although X.509 certificates have been criticized by [Bra00] for their unique serial number and unique issuer name, which makes them very easy to trace [AE00], we do not concern ourselves here with making anonymous Web Service calls.

10.1 Achievements

In this thesis are presented different approaches for increasing privacy in online transactions, based on X.509 private extensions, especially in credit card payment transactions.

The X.509 certificate binds a user's name with his public key. We have extended the X.509 certificate with new fields. These new fields contain user properties like credit card information, address, insurance number etc. and are stored like private extensions in the X.509 certificate. Each extension is encrypted with the public key of the respective entity: credit card numbers are encrypted with the credit card issuer company's public key, the insurance number is encrypted with the insurance company's public key and so on. Each extension is thus understood by only one entity, the one that owns the corresponding private key. We have stored the proposed extensions in:

- X.509 identity certificates,
- X.509 long-life attribute certificates, and
- X.509 single-use attribute certificates.

In Section 8.3 we compared identity certificates with attribute certificates and analyzed the advantages and disadvantages of each solution.

The origin of a transaction is traced by using the well known payment protocols as in Section 2.5. Non-repudiation is needed because of the changes we have made to the X.509 structure and the nature of the X.509 certificate. The X.509 certificate is usually stored in a public repository. The non-repudiation protocols are completed in:

- 10 steps, or
- 4 steps.

The protocol realized in 10 steps is similar to the standard credit card payment protocols presented in Section 4.2. The protocol realized in 4 steps makes use of the new X.509 certificate structure for encrypted credit card information.

The proposed approach at first glance seems similar to the Secure Electronic Transaction (SET) protocol presented in Section 2.5.3. We have used a different approach in encapsulating user properties (such as credit card numbers, addresses, insurance numbers etc.) in encrypted form in X.509 certificates. In Section 8.2 we compared the proposed approach with the SET protocol in detail.

A direct consequence of changing the structure of X.509 is that communication partners must use digital signatures for proving the origin of a transaction, i.e. that the transaction is triggered by the user claimed in X.509 certificate.

From the proposed approach the following parties can benefit:

- users – since they are assured that their private information is not shown to every party in the online transaction, and
- merchants – since they are no longer the target of information thieves (like credit card number thieves).

X.509 certificate are proposed to be stored in smartcards. In smartcards an X.509 certificate is always readable but its counterpart private key can be used only after presenting the user's PIN, i.e. after successful authentication. Also the encryption with a private key (digital signature) is calculated in a smartcard. The private key never leaves the smartcard. By integrating smartcards with the overall system architecture is improved:

- overall system security – without PIN knowledge the private key is never accessible and thus no correct digital signatures cannot be generated, and
- user mobility – smartcards have proven as portable and, due to their smooth integration with operating system services, are widely accepted for secure storage of private information such as private keys.

Using a test application we have simulated the client as a stand alone application in which the merchant and bank were implemented as Web Services. We have implemented the new private extensions in X.509 certificates and used them for Internet payment transactions and different encryption functions. For this implementation we have used OpenSSL and Infineon SICRYPT smartcards for secure storage of private keys and X.509 certificates.

A disadvantage of the proposed approach is that it requires changing the structure of X.509 certificates. In many cases it would suffice to change the certificate policy statement [1], but in some cases it would require more effort in order to change the rule paragraphs governing digital signatures.

The following elements of the work presented in this thesis:

- new X.509 identity and attribute certificate structure,
- private extension with encrypted user properties (structure and format),
- online exchange protocol (in 4 and 10 steps) for completing online transactions, and
- storing these new certificates in smartcards

are some of the 10 claims which Siemens AG in Munich has filed as a patent application to the German Patent and Trade Mark Office [2]. The patent authors are Blerim Rexha, as a representative of Siemens AG, with 90% shares and Albert Treytl, as a representative of Vienna University of Technology, with the remaining 10% shares. The patent application was submitted on 14.08.2003 and is titled: "*Increasing privacy with X.509 certificate private extensions in electronic transactions*" with Siemens internal dossier number: 2003P11991 DE.

10.2 Possible improvements

Analyzing the 10 step protocol presented in Section 8.1.1 we have shown that user masquerading (i.e. using the public key X.509 certificate of another user) is detected in step 7. Thus, user masquerading reduces the quality of service (QoS) provided by Web Service provider. Masquerading reduces the QoS because until the step 7 *a masqueraded user* and *a real user* have the same priority, the web server treats both users in same manner. User masquerading is detected in this step since here the user must digitally sign the transaction data, thus proving that he owns the private key whose counterpart public key is stated in the X.509 certificate. A possible improvement in this case includes confirmation earlier in the authentication process that the user owns the private key. It must be emphasized that this kind of user masquerading is not possible when SSL (with the mutual authentication option activated) or the 4 step protocol presented in Section 8.1.3 are used. In both protocols the user must prove in an early step that he owns a private key that matches the public key stated in X.509 certificate.

[1]This certificate policy statement defines the terms and conditions under which a certificate authority issues public key certificates.

[2]http://www.dpma.de

Analyzing the client application, where is used a X.509 certificate for payment transactions and for encrypting and decrypting purposes, leads to the idea of restricting the usage of the X.509 certificate. The X.509 certificate version 3 standard has an private extension named *Key usage*, whose main purpose is to restrict the usage of the public key certificate [ITU00]. Setting the *Key usage* in the X.509 certificate to *digital signature and non-repudiation* would enable the certificate for payment transactions but restrict it for encryption/decryption operations. The disadvantage of this proposal is that system complexity grows, since the user has to administer more certificates.

Web Services can be accessed from any platform that has Internet access. Applications in different platforms with an Internet connection can send and receive SOAP messages [BCG$^+$01]. A challenge would be to port the Windows client application presented in Chapter 9 to a different platform such as Linux or a handheld operating system. We do not expect any incompatibilities since X.509 certificates and Web Services are defined in a platform independent manner. We expect transaction payments in Linux and handheld operating systems environments to have the same security features as in Windows based systems.

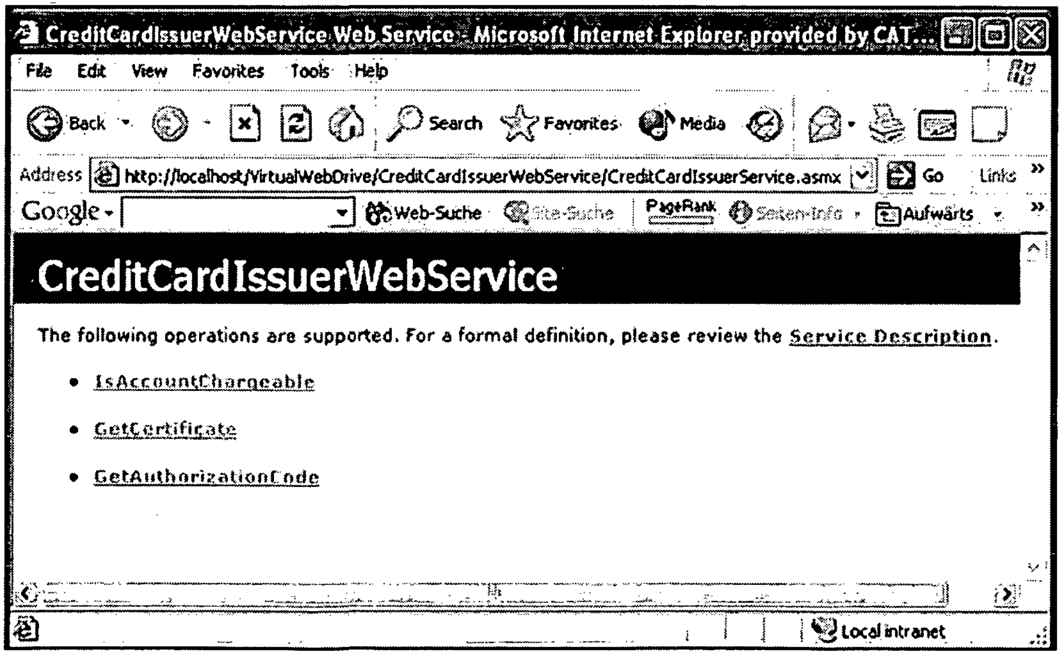
Appendix A

User Interface of Virtual Web Drive Application

Appendix A presents some screen shots of Virtual Web Drive application described in Chapter 9.

A.1 Credit card issuer Web Service

In Figure A.1 is presented the web interface exposed by credit card issuer Web Service. The *GetCertificate* web method is always accessible, other two methods can be accessed only after successful authentication.


CreditCardIssuerWebService Web Service - Microsoft Internet Explorer provided by CAT...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/VirtualWebDrive/CreditCardIssuerWebService/CreditCardIssuerService.asmx
Go Links
Google Web-Suche Site-Suche PageRank Seiten-Info Aufwärts
Local intranet

CreditCardIssuerWebService

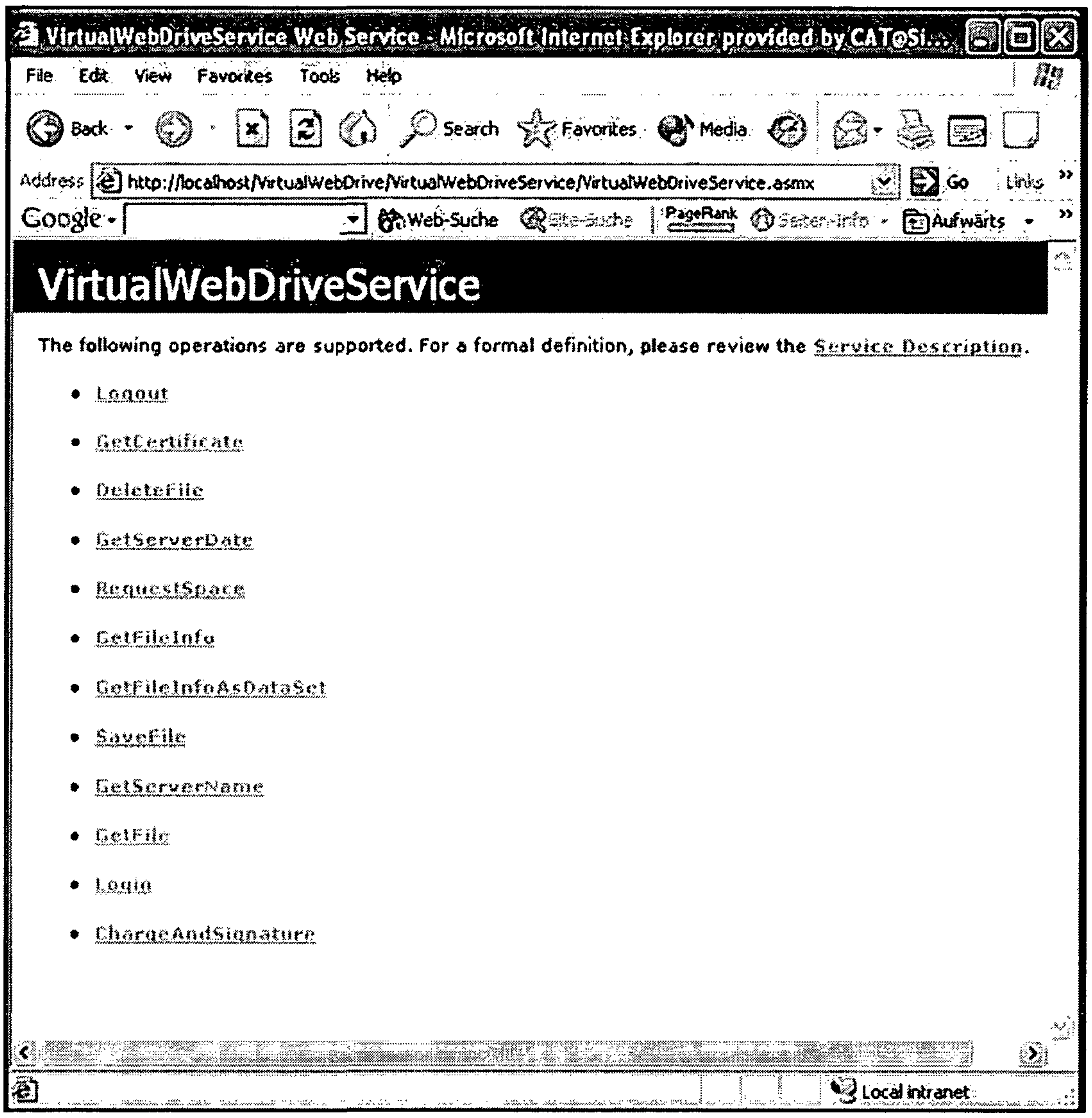
The following operations are supported. For a formal definition, please review the Service Description.

- IsAccountChargeable
- GetCertificate
- GetAuthorizationCode

Figure A.1: Credit card issuer Web Service

A.2 Virtual Web Drive Web Service

In Figure A.2 is presented the web interface exposed by virtual web drive Web Service provider (merchant). Some of the exposed web methods are always accessible like: *GetServerName*, *GetServerDate* and *GetCertificate*. For payment transactions are additionally these web methods accessible: *RequestSpace* and *ChargeAndSignature*. Other web methods are accessible when user is logged in.


VirtualWebDriveService Web Service - Microsoft Internet Explorer provided by CAT@Si...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/VirtualWebDrive/VirtualWebDriveService/VirtualWebDriveService.asmx
Go Links
Google Web-Suche PageRank Aufwärts
Local intranet

VirtualWebDriveService

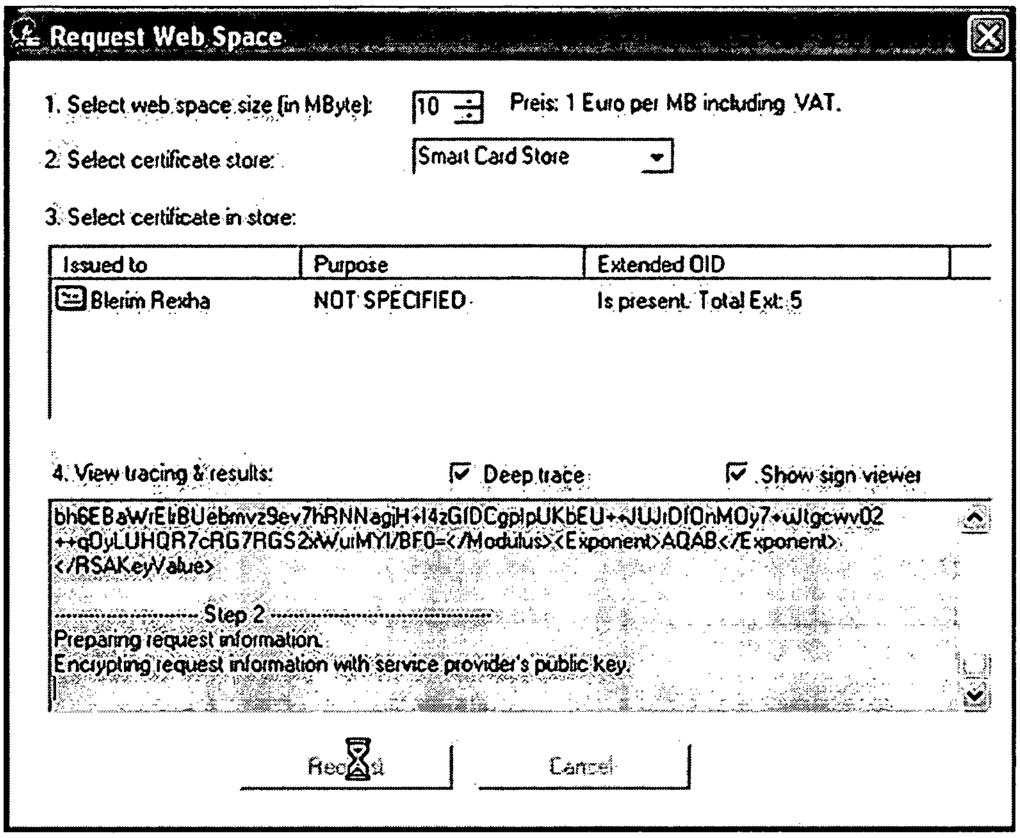
The following operations are supported. For a formal definition, please review the Service Description.

- Logout
- GetCertificate
- DeleteFile
- GetServerDate
- RequestSpace
- GetFileInfo
- GetFileInfoAsDataSet
- SaveFile
- GetServerName
- GetFile
- Login
- ChargeAndSignature

Figure A.2: Virtual Web Drive Web Service

A.3 Client application

Figure A.3 presents the requesting dialog of the client application, where user can request the web space at the Web Service provider(merchant). Through this dialog the user has a possibility to view and trace the communication with merchant and response messages from credit card issuer Web Service.



Request Web Space

1. Select web space size (in MByte): 10 Preis: 1 Euro per MB including VAT.

2. Select certificate store: Smart Card Store

3. Select certificate in store:

Issued to	Purpose	Extended OID
Blerim Rexha	NOT SPECIFIED	Is present. Total Ext: 5

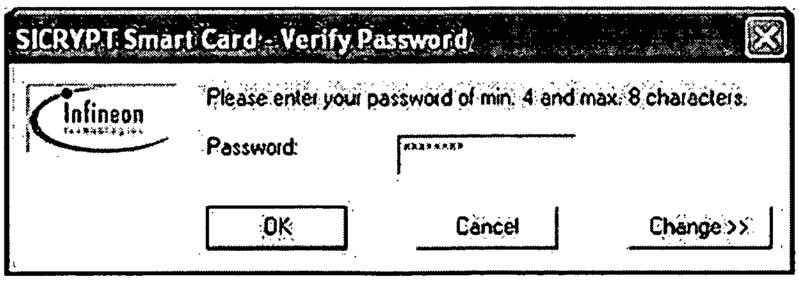
4. View tracing & results: ✓ Deep trace ✓ Show sign viewer

```
bh6EBaWrEliBUebmvz9ev7hRNNagiH+l4zGIDCgpIpUKbEU++JUJrDIOnMOy7+wJtgcwv02
++qOyLUHQR7cRG7RGS2xWuiMYI/BF0=</Modulus><Exponent>AQAB</Exponent>
</RSAKeyValue>

........................ Step 2 ........................................
Preparing request information.
Encrypting request information with service provider's public key.
```

Figure A.3: Requesting web space in client application

Access to private key, which is stored in smartcard, is protected with PIN. Each time an application needs to use the private key for encryption purposes the dialog PIN is prompted by client application, as presented in Figure A.4.



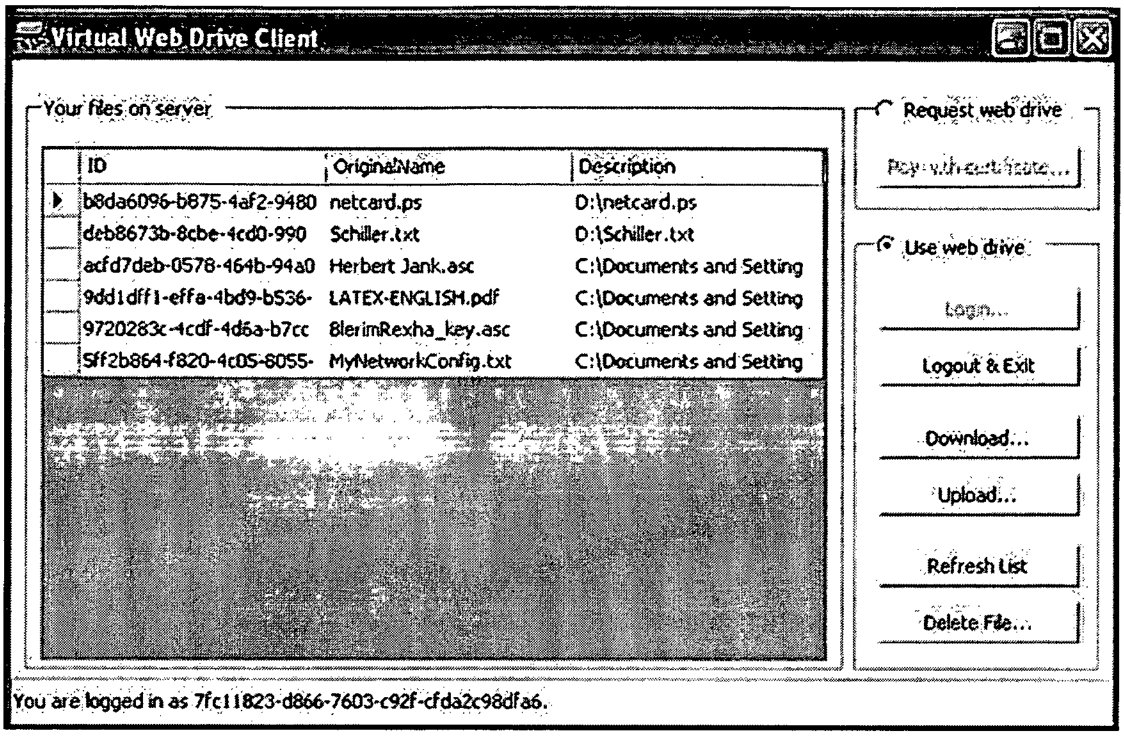
SICRYPT Smart Card - Verify Password

Please enter your password of min. 4 and max. 8 characters.

Password: ********

Figure A.4: SICRYPT smartcard service provider PIN dialog

In Figure A.5 is presented the client user interface after the user is logged in.



ID	OriginalName	Description
b8da6096-b875-4af2-9480	netcard.ps	D:\netcard.ps
deb8673b-8cbe-4cd0-990	Schiller.txt	D:\Schiller.txt
acfd7deb-0578-464b-94a0	Herbert Jank.asc	C:\Documents and Setting
9dd1dff1-effa-4bd9-b536-	LATEX-ENGLISH.pdf	C:\Documents and Setting
9720283c-4cdf-4d6a-b7cc	BlerimRexha_key.asc	C:\Documents and Setting
5ff2b864-f820-4c05-8055-	MyNetworkConfig.txt	C:\Documents and Setting

You are logged in as 7fc11823-d866-7603-c92f-cfda2c98dfa6.

Figure A.5: Client application user interface

List of Figures

List of Tables

List of Abbreviations

AA	Attribute Authority
AC	Attribute Certificate
AH	Authentication Header
AMEX	AMErican eXpress
API	Application Programming Interface
APDU	Application Protocol Data Unit
ASCII	American National Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation 1
ASP	Active Server Pages
ATM	Automatic Teller Machine
CA	Certification Authority
CAPICOM	Cryptographic Application Programming Interface COMponent
CBC	Cipher Block Chaining
CFB	Cipher Feedback
COM	Common Object Model
CIO	Chief Information Officer
CRL	Certificate Revocation List
DCE	Distributed Computing Environment
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DIME	Direct Internet Message Encapsulation
DISCO	DISCOvery
DMZ	DeMilitarized Zone
DNA	Distributed Network Architecture
ECB	Electronic Codebook
EEPROM	Electrically Erasable Programmable Read Only Memory
EJB	Enterprise Java Beans
ESP	Encapsulated Security Payload
FTP	File Transfer Protocol
GXA	Global XML Web Services Architecture
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transport Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IDEA	International Data Encryption Algorithm
IDL	Interface Description Language
IIOP	Internet Inter-ORB Protocol
IIS	Internet Information Services

IP	Internet Protocol
IPSec	Internet Protocol Security
ISP	Internet Service Provider
IT	Information Technology
ITU	International Telecommunications Union
ISO	International Organization for Standardization
J2EE	Java 2 Enterprise Edition
JCE	Java Cryptographic Exetension
JVM	Java Virtual Machine
LAN	Local Area Network
LDAP	Light Directory Access Protocol
MAC	Message Authentication Code
MARS	Multiplication, Addition, Rotation and Substitution
MD-5	Message Digest 5
MF	Master File
MIME	Multipurpose Internet Mail Extensions
MS	Microsoft
MS-CHAP	Microsoft CHallenge Authentication Protocol
NAT	Network Address Translation
.NET	New Enterprise Technology
NCSA	National Center for Supercomputing Applications
OASIS	Organization for the Advancement of Structure Information Standard
OCF	Open Card Framework
OEM	Original Equipment Manufacturer
OFB	Output Feedback
OI	Order Information
OID	Object Identifier
OO	Object Oriented
OSI	Open Systems Interconnection
ORB	Object Request Broker
PC/SC	Personal Computer Smart Card
PEM	Privacy Enhanced Mail
PGP	Pretty Good Privacy
PI	Payment Information
PIN	Personal Identification Number
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
PPTP	Point to Point Tunneling Protocol
QoS	Quality of Service
RA	Registration Authority
RAD	Rapid Application Development
RFC	Request For Comments
RMI	Remote Method Invocation

ROM	Read Only Memory
RPC	Remote Procedure Call
RSA	Rivest Shamir Adleman
SAML	Security Assertions Markup Language
SET	Secure Electronic Transaction
SHA-1	Secure Hash Algorithm 1
SICRYPT	SIemens CRYPTography
SLDAP	Secure Light Directory Access Protocol
SMIME	Secure Multipurpose Internet Mail Exchange
SMTP	Simple Mail Transfer Protocol
SNNTP	Secure Network News Transfer Protocol
SOAP	Simple Object Access Protocol
SOHO	Simple Office and Home Office
SPP	Sequenced Packet Protocol
SRA	Secure RPC Authentication
SSL	Secure Socket Layer
SSH	Secure SHell
SSMTP	Secure Simple Mail Transfer Protocol
SPX	Sequenced Packet eXchange
SPOP3	Secure Post Office Protocol version 3
SW	Status Word
TCP	Transport Control Protocol
TLS	Transport Layer Security
TLV	Tag Length Value
TN	Transaction Number
TTP	Trusted Third Party
UDDI	Universal Discovery Description and Integration
UDP	User Datagram Protocol
UI	User Interface
URI	Unique Resource Identifier
USB	Universal Serial Bus
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WAN	Wide Area Network
WS	Web Services
WSDL	Web Services Description Language
WSE	Web Services Enhancement
WWW	World Wide Web
XACML	XML Access Control Markup Language
XML	eXtended Markup Language
XKMS	XML Key Management Specification
XKISS	XML Key Information Service Specification
XKRSS	XML Key Registration Service Specification

Bibliography

[ABK99] Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A candidate block cipher for the advanced encryption standard. http://www.cl.cam.ac.uk/ rja14/serpent.html, 1999.

[Abr01a] Dennis Abrazhevich. Classification and characteristics of electronic payment systems. *Lecture Notes in Computer Science*, 2115:81–90, 2001.

[Abr01b] Dennis Abrazhevich. Electronic payment systems: Issues of user acceptance. Technical University of Eindhowen, Eindhowen, Nethrlands, also available at: http://citeseer.nj.nec.com/527052.html, 2001.

[Ada97] C Adams. The cast-128 encryption algorithm. Network Working Group; Request for Comments: 2144 at http://www.faqs.org/rfcs/rfc2144.html, May 1997.

[ADLH$^+$02] Bob Atkinson, Giovanni Della-Liibera, Satoshi Hada, Maryann Hondo, Phillip Hallam-Baker, Chris Kaler, Johannes Klein, Brian LaMacchia, Paul Leach, John Manferdelli, Hiroshi Muruyama, Anthony Nadalin, Nataraj Nagaratnam, Hemma Prafullchandra, John Shewchuk, and Dan Simon. Web services security (ws–security), version 1.0. http://www-106.ibm.com/developerworks/library/ws-secure/, April 2002.

[AE00] Toumas Auro and Carl Ellison. Privacy and accountability in certificate systems. Technical report, Helsinki University of Technology, 2000.

[AFPS99] R. Adams, W. Ford, W. Polk, and D. Solo. Internet x.509 public key infrastructure certificate and crl profile. Network Working Group; Request for Comments: 2459 at http://www.faqs.org/rfcs/rfc2459.html, January 1999.

[AG99] C. Adams and J. Gilchrist. The cast-256 encryption algorithm. Network Working Group; Request for Comments: 2612 at http://www.faqs.org/rfcs/rfc2612.html, June 1999.

[AJSW97] N. Asokan, Phillipe Janson, Michael Steiner, and Michael Waidner. The state of the art in electronic payment systems. *IEEE Computer*, 30(9):28–35, 1997.

[Ali00] Spiro Alifrangis. Attribute certificates. Council on Technology Services Digital Signatures Initiative (DSI) Work Group. Also available at: http://www.cots.state.va.us/minutes/ds081000/ACS.ppt, August 2000.

[AM03] AM. Frage des monats: Wie sicher ist passport? (german language). PC-WELT, 7/2003, pp.29, 2003.

[Ame01] American Express. Private payments: Frequently asked questions. http://www26.americanexpress.com/privatepayments/faq.jsp, 2001.

[ANS81] ANSI. The American National Standards Institute; American National Standard for Data Encryption Algorithm (DEA). ANSI X.3.92, 1981.

[Aps02] Kapil Apshankar. Ws-security: Security for web services. http://www.webservicesarchitect.com, 2002.

[Arn00] Andre Arnes. Selecting revocation solutions for pki, paper submitted to norsec 2000. http://www.pvv.ntnu.no/ andrearn/certrev/, September 2000.

[AT02] A. Arsenault and S. Turner. Internet x.509 public key infrastructure: Roadmap. PKIX Working Group; Internet Draft: http://www.ietf.org/internet-drafts/draft-ietf-pkix-roadmap-09.txt, July 2002.

[Bac01] Daniel Bachfeld. Sicheres netz im netz (in german language). ct, Magazin für Computer Technik, Nr.17, http://www.ctmagazin.de, 2001.

[Bal03] Baltimore Technologies. Attribute certificates. http://www.baltimore.com/devzone/pki/attributecertificates.asp, 2003.

[BBF+02] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. Xml-signature syntax and processing. W3C Recommendation, available at: http://www.w3.org/TR/xmldsig-core/, February 2002.

[BCD97] Greg Bossert, Simon Cooper, and Walt Drummond. Considerations for web transaction security. Network Working Group; Request for Comments: 2084 at http://www.ietf.org/rfc/rfc2084.txt, January 1997.

[BCD+99] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas Jr., Luke O'Connor, Mohammad Peyravian, david Safford, and Nevenko Zunic. Mars - a candidate cipher for aes. Technical report, IBM Corporation; http://www.research.ibm.com/security/mars.pdf, September 1999.

[BCG+01] Ashish Banerjee, Aravind Corea, Zach Greenvoss, Andrew Kroczyk, Christian Nagel, Chris Peiris, Thiru Thangarathiam, and Brad Maiani. *C# Web Services - Building Web Services with .NET Remoting and ASP.NET.* Wrox Pres Ltd, ISBN = 1-8610004-39-7, 2001.

[BDNP97] William Burr, Donna Dodson, Noel Nazario, and W. Timothy Polk. Minimum interoperability specification for pki components, version 1 (mipspc). Technical report, NIST, http://csrc.nist.gov/pki/mispc/welcome.html, September 1997.

[BGH$^+$95] Mihir Bellare, Juan Garay, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steiner, Gene Tsudik, and Michael Waidner. iKP – A family of secure electronic payment protocols. In *First USENIX Workshop on Electronic Commerce, New York*, pages 89–106, July 1995.

[BLFF96] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext transfer protocol – http/1.0. Network Working Group; Request for Comments: 1945 at http://www.faqs.org/rfcs/rfc1945.html, May 1996.

[Blo03] Sandra Block. How to protect your credit card from headaches of id theft. http://www.usatoday.com, January 28 2003.

[BNP97] William E. Burr, Noel A. Nazario, and W. Timothy Polk. A proposed federal pki using x.509 v3 certificates. National Institute of Standards and Technology, Gaithersburg MD, USA,, 1997.

[Boe02] Wolgang Boehmer. *VPN Virtual Private Networks Die reale Welt der virtuellen Netze.* Carl Hanser Verlag, München Wien, ISBN = 3-446-21532-8, 2002.

[Bol99] Himabindu Bolisetty. Electronic payment systems: Echas. http://citeseer.nj.nec.com/283725.html, February 1999.

[Bou03] Clint Boulton. Gartner: Web services strong amid sluggish economy. http://www.internetnews.com/ent-news/article.php/2239431, 2003.

[Box00] Don Box. A young person's guide to the simple object access protocol: Soap increases interoperability across platforms and languages. http://msdn.microsoft.com/msdnmag/issues/0300/soap/default.aspx, March 2000.

[Boy01] John Boyer. Canonical xml version 1.0. W3C Recommendation, available at: http://www.w3.org/TR/xml-c14n, March 2001.

[Bra00] Stefan A Brands. *Rethinking Public Key Infrastructure and Digital Certificates, Building in Privacy (Ph.D. thesis updated as book).* The MIT Press, ISBN = 0-262-02491-8, 2000.

[Bro01] Bob Brown. Uddi universal description, discovery and integration. Technical report, http://www.transentia.com.au, 2001.

[BS02a] Jason Bloomberg and Ron Schmelzer. Pro and cons of web services. Technical report, ZapThink Inc, http://www.zapthink.com, USA, May 2002.

[BS02b] Jason Bloomberg and Ron Schmelzer. Securing & managing xml & web services in the enterprise. Technical report, ZapThink Inc, http://www.zapthink.com, USA, 2002.

[BTN00] John J. Barton, , Satish Thatte, and Henrik Frystyk Nielsen. Soap messages with attachments. W3C Note, http://www.w3.org/TR/SOAP-attachments, December 2000.

[Bur98] W. E. Burr. Proposed federal pki architecture. http://csrc.nist.gov/pki/twg/papers/arch2.pdf, May 1998.

[Cab02] Tom Cabanski. Microsoft .net will beat java. http://www.manning.com/dotnetbooks/java_vs_dotnet/java-vs-dotnet.html, 2002.

[Car00] Germano Caronni. Walking the web of trust. Published in the proceedings of the 9th Workshop on Enabling Technologies (WET ICE2000), IEEE Computer Society Press., 2000.

[CCC+02] Felipe Cabrera, George Copeland, Bill Cox, Tom Freund, Johannes Klein, Tony Storey, and Satish Thatte. Web services transaction (ws–transaction). http://msdn.microsoft.com/webservices/understanding/gxa/default.aspx, 2002.

[CCF+02] Felipe Cabrera, George Copeland, Tom Freund, Johannes Klein, Tony Storey, David Langworthy, David Orchard, and John Shewchuk. Web services coordination (ws–coordination). http://www-106.ibm.com/developerworks/library/ws-coor/, August 2002.

[CDI96] Chris Chambers, Justin Dolske, and Jayaramen Iyer. TCP/IP Security. Department of Computer and Information Science, Ohio State University also available at: http://www.linuxsecurity.com/resource_files/documentation/tcpip-security.html, (Year of publication is assumed to be:), 1996.

[Cer00] Certicom. The elliptic curve cryptosystem, an introduction to information security. Technical report, Certicom Inc. http://www.certicom.com, 2000.

[Cer02] Ethan Cerami. Web services essentials - dsitributed applications with xml-rpc, soap, uddi & wsdl. O'Reily Inc., http://www.oreily.com, USA, 2002.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 24(2):84-88, February 1981.

[Cha88] David Chaum. Blinding for unanticipated signatures. In Advances in Cryptology Proc. EUROCRYPT 87, pp.227-233, LNCS 304, Springer Verlag, Berlin, 1988.

[Cha92] David Chaum. Achieving electronic privacy. Scientific American, pp96-101, also available at: http://www.chaum.com/articles/Achieving_Electronic_Privacy.htm, August 1992.

[Cha02] David Chappell. Web services toady - seven maxims. Closing Key Note on Austria .NET Conference, also available at www.davidchappell.com, October 2002.

[Chi02] Mwelwa Chibesakunda. Digital certificates: Study on attribute certificates. University of Cape Town, http://people.cs.uct.ac.za/ mchibesa/report.pdf, 2002.

[CHVV03] Brice Canvel, Alain Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a ssl/tls channel. To appear in Lecture Notes in Computer Science, also available at: http://lasecwww.epfl.ch/pub/lasec/doc/CHVV03.ps, 2003.

[CK00] Reuven Cohen and Gideon Kaempfer. On the cost of virtual private networks. *IEEE/ACM Transactions on Networking*, 8(6), December 2000.

[Cla02] Mike Clark. Selling web services. This article is an extract from Web Services Business Strategies and Architectures. ISBN = 1-90428-413-2, also available at: http://www.webservicesarchitect.com/content/articles/clark03.asp, 2002.

[CNW01] Francisco Curbera, William A Nagy, and Sanjiva Weerawarana. Web services: Why and how. *IBM, T.J. Watson Research Center*, August 2001.

[COHL02] David W. Chadwick, Olexandre Otenko, David Hunter, and Cristiano Leoni. Privilege management for e-construction. ISI, University of Salford, Salford , UK, also available at: http://sec.isi.salford.ac.uk/download/eSMART.pdf, 2002.

[Col03a] Mark Colan. Making web services secure. http://ibm.com/developerworks/speakers/colan, March 2003.

[Col03b] Mark Colan. A technical overview of web services. Technical report, IBM Corporation, March 2003.

[Com00a] Douglas Comer. *Internetworking with TCP/IP: Principles, Protocols and Architecture 4Ed.* Prentice Hall, New Jersey, ISBN = 0-13-0183806-6, 2000.

[Com00b] Commision of the European Communities. Directive of the european parliament and of the council: concerning the processing of personal data and the protection of privacy in the electronic communications sector. http://europa.eu.int/eur-lex/en/com/pdf/2000/en_500PC0385.pdf, July 2000.

[Con03] Internet Software Consortium. Internet domain survey. http://www.isc.org/ds/, January 2003.

[CR00] Matt Curtin and Marcus J. Ranum. Internet Firewalls: Frequently Asked Questions. http://www.interhack.net/pubs/fwfaq/, December 2000.

[CW02] National Fraud Information Center and Internet Fraud Watch. Internet fraud statistics: 2002 top 10 frauds. NFIC, http://www.fraud.org/welcome.htm, 2002.

[DA99] T. Dierks and C. Allen. The tls protocol version 1.0. Network Working Group; Request for Comments: 2246 at http://www.ietf.org/rfc/rfc2246.txt, January 1999.

[Dev01] Don Devis. Defective sign-and-encrypt can you really trust s/mime, pcks#7, pgp and xml? Dr. Bobb's Journal of Software Tools, http://www.ddj.com/, November 2001.

[DH99] Naganand Doraswamy and Dan Harkins. *IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks.* Prentice Hall, USA, ISBN = 0130118082, October 1999.

[Dja02] Ray Djajadinata. Yes, you can secure your web services documents. http://www.javaworld.com/javaworld/jw-08-2002/jw-0823-securexml.html, August 2002.

[DK02] Hans Delf and Helmut Knebl. *Introduction to Cryptography Principles and Applications.* Springer Verlag Berlin Heidelberg, ISBN = 3-540-42278-1, 2002.

[DR99] Joan Daemen and Vincent Rijmen. Aes proposal: Rijndael. http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf, September 1999.

[Dri02] Mark Driver. .net vs. java: No easy answers .net and java are both here to stay. how do you choose which is right for you? Gartner Research, http://www.fawcette.com/dotnetmag/2002_04/magazine/columns/strategy/default_pf.aspx April 2002.

[EFF99] EFF. The Electronic Frontier Foundation; Cracking DES. http://www.eff.org/Privacy/Crypto_misc/DES_Cracking/, 1999.

[Eli03] Ilan Elias. PEM. The Hebrew University - Institute of Computer Science, also available at: http://www.cs.huji.ac.il/course/2002/sans/, 2003.

[Ent01] Entrust. Web services trust and xml security standards. http://www.entrust.com, April 2001.

[EPI02] Electronic Privacy Information Center EPIC. Sign out of passport! http://www.epic.org/privacy/consumer/microsoft/default.html, January 2002.

[EPL02] Robert Elfwing, Ulf Paulsson, and Lars Lundberg. Performance of soap in web service enviroment compared to corba. *IEEE Proceedings of the Ninth Asia-Pacific Software Engineering Conference (APSEC'02)*, August 2002.

[ES00] Carl Ellison and Bruce Schneier. Ten risks of pki: What you're not being told about public key infrastructure. Computer Security Journal, Volume XVI, Number 1, http://www.counterpane.com/pki-risks.html, 2000.

[Evj03] Bill Evjen. *Web Services Enhancements Understanding the WSE for .NET Enterprise Applications.* Wiley Publishing Inc., ISBN = 0-7645-3736-9, 2003.

[FAKB02] Alia Fourati, Hella Kaffel Ben Ayed, Farouk Kamoun, and Abdelmalek Benzekri. A SET Based Approach to Secure the Payment in Mobile Commerce. Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN.02), March 2002.

[FB01] Warwick Ford and Michael Baum. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption.* Prentice Hall, ISBN = 0130272760, 2001.

[Fer02] Edurado B Fernandez. Web services security - current status and future. http://www.webservicearchitect.com, March 2002.

[FF01] Jalal Feghhi and Jalil Feghi. *Secure Networking with Windows 2000 and trust services.* Addison Wesley , ISBN = 0-201-65778-3, 2001.

[FFW99] Jalal Feghhi, Jalil Feghi, and Peter Williams. *Digital Certificates: Applied Internet Security.* Addison Wesley , ISBN = 0-201-30980-7, 1999.

[FH02] S. Farrell and R. Housley. An internet attribute certificate profile for authorization. Network Working Group; Request for Comments: 3281 at http://www.ietf.org/rfc/rfc3281.txt, April 2002.

[FHBH+99] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. Http authentication: Basic and digest access authentication. Network Working Group; Request for Comments: 2617 at http://www.ietf.org/rfc/rfc2617.txt, June 1999.

[Fin00] Klaus Finkenzeller. *RFID Handbuch: Grundlagen und praktische Anwendungen induktiver Funkanlagen, Transponder und kontakloser Chipkarten, 2nd Edition.* Carl Hanser Verlag München Wien, http://www.hanser.de, ISBN = 3-446-21278-7, 2000.

[Fis02] Micheal Fischer. Towards a generalized payment model for internet services. Master's thesis, Vienna University of Technology, Institute of Information Systems, Vienna, September 2002.

[FK00] Michael Frischer and Oliver Kump. Security and productivity improvements – sufficient for the success of secure electronic transaction? Department of Information Systems, Vienna University of Economics and Business Administration, Vienna, August 2000.

[FKK96] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL Protocol Version 3.0. Transport Layer Security Working Group, Internet Draft, also available at: http://wp.netscape.com/eng/ssl3/, November 1996.

[Fla02] David Flanagan. *Java in a Nutshell; A desktop quick refernce, 4-th Edition.* O'Reilly & Associates Inc., ISBN = 0-596-00283-1, 2002.

[Fly03] Peter Flynn. The xml faq. http://www.ucc.ie:8080/cocoon/xmlfaq, 2003.

[Fou03] The Apache Software Foundation. Apache HTTP Server Project. http://www.apache.otg, 2003.

[Fre03] FreeBSD Handbook. The FreeBSD Documentation Project. FreeBSD Org, available at: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/index.html, 2003.

[FS99] Niels Ferguson and Bruce Schneier. A cryptographic evaluation of ipsec. http://www.counterpane.com/ipsec.html, February 1999.

[FW02] Peter Fletcher and Mark Waterhouse. *Web Services Business Strategies and Architectures.* Expert Press Ltd. UK, ISBN = 1-90428-413-2, 2002.

[Gai02] Jeanine Gailey. The challenges for security - report from web service one. Technical report, http://www.webservice.org, 2002.

[Ger00] Ed Gerck. Overview of certification systems: X.509, pkix, ca, pgp & skip. Technical report, The Bell, http://www.thebell.net/papers, USA, July 2000.

[Gho98] Anup Ghosh. *E-Commerce Security Weak Link, Best Defences.* John Wiley Inc. New York, USA, ISBN = 0-471-19223-6, 1998.

[GJ98] Scott Guthery and Timothy Jurgensen. *Smart Card Developer Kit.* Macmillian Technical Publishing, Indianapolis, USA, ISBN = 1-57870-027-2, 1998.

[Gol00] Don Goldhamer. Privacy concerns. Boston Massachusetts, also available at: http://home.uchicago.edu/ dhgo/privacy-intro/, July 2000.

[Gri97] Richard Grimes. *Professional DCOM Programming; A guide to creating practical application with Microsoft's Distributed Component Object Model.* Wrox Press Ltd. Canada, ISBN = 1-861000-60-X, 1997.

[Gri99] Ian Grigg. How digicash blew everything. http://www.shmoo.com/mail/cypherpunks/feb99/msg00113.html, February 1999.

[Gro01] William Grosso. *Java RMI.* O'Reilly & Associates Inc., ISBN = 1-56592-452-5, 2001.

[GS01] Simson Garfinkel and Gene Spafford. *Web Security, Privacy & Commerce 2Ed.* O'Reilly Inc. http://www.oreilly.com, USA, ISBN = 0-596-00045-6, November 2001.

[Gut00] Peter Gutmann. X.509 style guide. University of Auckland, http://www.cs.auckland.ac.nz/ pgut001/pubs/x509guide.txt, October 2000.

[Gut02] Peter Gutmann. Pki: It's not dead , just resting. *IEEE Computer Society, Vol.35 No.8 pp.41-49)*, August 2002.

[Han01] Whitney Hankison. Network security and web services deployment. http://www.webservicesarchitect.com/content/articles/hankison02.asp, November 2001.

[Har00] Elliotte Rusty Harold. *Java Network Programming.* O'Reilly & Associates Inc., ISBN = 1-56592-870-9, 2000.

[Has01] Vesna Hassler. *Security Fundamentals for E-Commerce.* Artech House, ISBN = 1-58053-108-3, 2001.

[Her02] Roland Herbst. Private keys vor zugriff schützen (in german language). ct, Magazin für Computer Technik, Nr.6, www.ctmagazin.de, June 2002.

[HMGM02] Vesna Hassler, Martin Manninger, Mikhail Gordeev, and Christopf Müller. *Java Card for E-Payment Applications.* Artech House, Boston, www.artechhouse.com, ISBN = 1-58053-291-8, 2002.

[HNN02] M. Hondo, N. Nagaratnam, and A. Nadalin. Securing web services. Technical report, IBM Corporation, System Journal, Vol. 41, No.2, 2002.

[HS98] Yung-Kao Hsu and Stephen P. Seymourne. An intranet security framework based on short-lived certificates. IEEE Internet Computing, pp73-79, March 1998.

[Hun01] Ray Hunt. Pki and digital certification infrastructure. *Ninth IEEE International Conference on Networks (ICON'01)*, October 2001.

[HW00] Edward F. Halpin and Steve Wright. The hidden dimensions of global information networks: What price privacy? *Canadian Association for Information Science, Proceedings of the 28th Annual Conference CAIS 2000, also available at: http://www.slis.ualberta.ca/cais2000/halpin.htm*, 2000.

[Iac02] Luigi Lo Iacono. Abhören von ip-telefonaten: Rote telefone (german language). ct, Magazin für Computer Technik, Nr.5, www.ctmagazin.de, May 2002.

[IBM98] IBM. Opencard framework, general information web document. http://www.opencard.org/docs/gim/ocfgim.pdf, October 1998.

[IDS02] Takeshi Imamura, Blair Dillaway, and Ed Simon. Xml encryption syntax and processing. W3C Working Group. available at: http://www.w3.org/TR/xmlenc-core/, December 2002.

[IM02a] IBM and Microsoft. A Joint WS-Security Application Note from IBM Corporation and Microsoft Corporation. http://www.ibm.com, August 2002.

[IM02b] IBM and Microsoft. Security in a web service world: A proposed architecture. http://msdn.microsft.com, April 2002.

[Inf01a] Infineon Technologies. Preliminary short product information: Sle66cux640p. http://www.infineon.com, August 2001.

[Inf01b] Infineon Technologies. Product information: Security & chip card ics sle 55r16 intelligent 2560byte eeprom with contactless interface complying to iso/iec 14443 type a and security logic. http://www.infineon.com, January 2001.

[Inf03a] Infineon Technologies. Faqs and support. http://www.sicrypt.com, 2003.

[Inf03b] Infineon Technologies. Preliminary short product information: Sle88cx720p. http://www.infineon.com, June 2003.

[Inf03c] Infineon Technologies. Security & chip card ics, interface specification sicrypt secure token platform for public key cryptography version 2.1. http://www.sicrypt.com, June 2003.

[ISO95] ISO/IEC7816-4. Information technology - identification cards - integrated circuit(s) cards with contacts part 4:interindustry commandsfor interchange. International Organization for Standardization, September 1995.

[ISO97a] ISO/IEC7816-3. Information technology - identification cards - integrated circuit(s) cards with contacts part 3: Electronic signals and transmission protocls, second edition. International Organization for Standardization, December 1997.

[ISO97b] ISO/IEC7816-4-1. Information technology - identification cards - integrated circuit(s) cards with contacts part 4:interindustry commandsfor interchange; amendment 1: Impact of secure messaging on the structures of apdu messages. International Organization for Standardization, December 1997.

[ITU92] ITU. International Telecommunication Union; Naming, Addressing and Registration; Procedures for the Operation of OSI Registration Authorities: General Procedures. ITU-T Rec. X.660 (1992); ISO/IEC 9834-1:1993, 1992.

[ITU00] ITU. International Telecommunication Union - Telecommunication Standardization Sector ITU-T; The Directory: Public Key and Attribute Certificate Frameworks. http://www.itu.int, 2000.

[IWV03] IWV. Web services im e-government. Technical report, Competence-Center CC eGov des Instituts für Wirtschaft und Verwaltung IWV der Hochschule für Wirtschaft und Verwaltung HSW in Bern, http://www.webservice.iwv.ch/, Schweiz, Jan 2003.

[JZ02] Mario Jeckle and Barabara Zengler. Soap aber sicher. http://www.jeckl.de Germany, February 2002.

[KA98a] Stephen Kent and Randall Atkinson. Ip authentication header. Network Working Group; Request for Comments: 2402 at http://www.ietf.org/rfc/rfc2402.txt, November 1998.

[KA98b] Stephen Kent and Randall Atkinson. Ip encapsulating security payload (esp). Network Working Group; Request for Comments: 2406 at http://www.ietf.org/rfc/rfc2406.txt, November 1998.

[KA98c] Stephen Kent and Randall Atkinson. Security architecture for the internet protocol. Network Working Group; Request for Comments: 2401 at http://www.ietf.org/rfc/rfc2401.txt, November 1998.

[Kal92] B. Kaliski. The md2 message-digest algorithm. Network Working Group; Request for Comments: 1319 at http://www.faqs.org/rfcs/rfc1319.html, April 1992.

[Kar02] Richard Karpinski. Saml demo shows potential of single sign-on. http://www.internetwk.com/security02/INW20020715S0007, July 2002.

[Kei03] Guy Keinan. S/MIME. The Hebrew University - Institute of Computer Science, also available at: http://www.cs.huji.ac.il/course/2002/sans/, 2003.

[Ken93] Stephen Kent. Internet privacy enhanced mail. Communications of the ACM, 36(8), pp.48-60, August 1993.

[Kes03] Gary C Kesseler. An overview of cryptography. http://www.garykessler.net/library/crypto.html, May 2003.

[KG02] A.T. Kearney and The Stencil Group. The emerging web services market. http://www.atkearney.com and http://www.stencilgroup.com, 2002.

[Kho99] Reena Khosla. Critical review of security achieved through itu-t x.509 and pgp certification standards. http://courses.cs.vt.edu/ cs5204/archive/ProjectFiles/ReenaKhosla.pdf, March 1999.

[Kid01] Eric Kidd. Xml-rpc howto. http://xmlrpc-c.sourceforge.net/xmlrpc-howto/xmlrpc-howto.html, 2001.

[Kin99] Gary King. Secure online credit card transactions. Windows and .NET Magazine, available at: http://www.winnetmag.com/Article/ArticleID/4774/4774.html, 1999.

[KN93] J. Kohl and C. Neuman. The kerberos network authentication service (v5). Network Working Group; Request for Comments: 1510 at http://www.ietf.org/rfc/rfc1510.txt, September 1993.

[Kno02] Eric Knorr. Microsoft, ibm offer ws-security spec to oasis. ZDNet, http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2872547,00.html, June 2002.

[Knu98] Jonathan Knudsen. *Java Cryptography.* O'Reilly & Associates Inc., ISBN = 1-56592-402-9, 1998.

[Koe97] Thomas Koenig. Ssh (secure shell) faq - frequently asked questions. http://www.uni-karlsruhe.de/ ig25/ssh-faq/ssh-faq.html, June 1997.

[KR00a] Jorma Kajava and Timo Remes. Intranet security from organizational point of view. Proceedings of IRIS 23. Laboratory for Interaction Technology, University of Trollhättan Uddevalla,, 2000.

[KR00b] David P. Kormann and Aviel D. Rubin. Risks of the passport single signon protocol. Computer Networks, Elsevier Science Press, volume 33, pages 51-58, also available at:http://avirubin.com/passport.html, 2000.

[Kre01] Stefan Krempl. Zeigt her eure finger: Erhebliche schwächen bei biometrischen verfahren. ct, Magazin für Computer Technik, Nr.4, www.ctmagazin.de, 2001.

[LS03] Harald Leitenmueller and Beat Schwegler. Java and .net interop - how to. http://www.dotnetexperts.at, April 2003.

[LV01] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research,* 14(4):255–293, 2001.

[MAM+99] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 internet public key infrastructure: Online certificate status protocol - ocsp. Network Working Group; Request for Comments: 2560 at http://www.faqs.org/rfcs/rfc2560.html, June 1999.

[Man98] Martin Manninger. *Netzwerksicherheit durch Chipkarten.* PhD thesis, Technische Universität Wien, Fakultät für Elektrotechnik, Institut für Computertechnik, Wien, 1998.

[MB01] H. X. Mel and Doris Baker. *Cryptography Decrypted.* Addison - Wesley, ISBN = 0-201-61647-5, 2001.

[McC98] Lind McCarthy. *Intranet Security Stories From the Trenches.* Prentice Hall, USA, ISBN = 0-13-894759-7, January 1998.

[Mic00a] Microsoft. Microsoft windows 2000 server documentation, requesting certificates. http://www.microsoft.com/windows2000/en/server/help/sag_CMreqCerts.htm, February 2000.

[Mic00b] Microsoft. Windows 2000-based virtual private networking: Supporting vpn interoperability. http://www.microsoft.com/windows2000, 2000.

[Mic01a] Microsoft. Building xml-based web services in microsoft .net vs. ibm websphere 4.0. http://www.gotdotnet.com/team/compare/webservicecompare.aspx, December 2001.

[Mic01b] Microsoft. Microsoft internet information services. IIS help on Windows XP, http://localhost/iishelp/iis/misc/default.asp, 2001.

[Mic02a] Microsoft. Microsoft Windows Server 2003, Technical Overview of Application Services. http://www.microsoft.com/windowsserver2003, November 2002.

[Mic02b] Microsoft. Sql server books online (updated sp3), transaction. SQL Server Help, 2002.

[Mic02c] Microsoft Support. Q307267 - how to: Secure xml web service with secure socket layer in windows 2000. Microsoft Support, http://support.microsoft.com, August 2002.

[Mic03a] Microsoft. Microsoft .net passport for businesses. http://www.microsoft.com/net/services/passport/business.asp, March 2003.

[Mic03b] Microsoft. Microsoft office 11. http://www.microsoft.com, 2003.

[Mic03c] Microsoft. Microsoft Windows Server 2003, Technical Overview of Internet Information Services. http://www.microsoft.com/windowsserver2003, April 2003.

[Mic03d] Microsoft Support. Q257591 - description of the secure sockets layer (ssl) handshake. Microsoft Support, http://support.microsoft.com, May 2003.

[Mid03] Stefan Middendorf. Sicherheit von web services: Sicher bedient (german language). ct, Magazin für Computer Technik, Nr.3, www.ctmagazin.de, March 2003.

[MIT03] MIT. Massachusetts Institute of Technology; Kerberos: The Network Authentication Protocol. http://web.mit.edu/kerberos/www/, 2003.

[MJ03] Mattew MacDonald and Erik Johansson. *C# Data Security Practical .NET Cryptography Handbook*. Wrox Press Ltd. UK, ISBN = 1-86100-801-5, 2003.

[MMVD02] J. D. Meier, Alex Mackman, Sriath Vasireddy, and Michael Dunner. *Building Secure ASP.NET Applications, Authentication, Authorization and Secure Communication.* Microsoft Press, Online Book, 2002.

[MSD01a] MSDN. Microsoft Developer Network Library, Platform SDK, Security: About Smart Card. http://msdn.microsft.com, October 2001.

[MSD01b] MSDN. Microsoft Developer Network Library, Platform SDK, Security: Using Certificate Enrollment Control. http://msdn.microsft.com, August 2001.

[MSD01c] MSDN. Microsoft Developer Network Libryry , XML Web Services Basics. http://msdn.microsft.com, December 2001.

[MSD03a] MSDN. Microsoft Developer Network Library, Platform SDK, Security: Architecture of a Cryptographic Service Provider. http://msdn.microsft.com, August 2003.

[MSD03b] MSDN. Microsoft Developer Network Library; XML Web Services Basics. http://msdn.microsft.com/library, June 2003.

[Mye02] Judith M. Myerson. Web service architecture. http://www.webservicesarchitect.com, USA, 2002.

[NBB$^+$00] James Nechvatal, Elaine Barker, Lwarence Bassham, William Burr, Morris Dworkin, James Foti, and Edward Roback. Report on the development of the advanced encryption standard (aes). Technical report, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Technology Administration, U.S. Department of Commerce, October 2000.

[NCF02] Henrik Frystyk Nielsen, Erik Christensen, and Joel Farrell. Ws–attachments specification. Internet draft, available at: http://www-106.ibm.com/developerworks/webservices/library/ws-attach.html, June 2002.

[NCL01] Henrik Frystyk Nielsen, Erik Christensen, and Steve Lucco David Levin. Web services referral protocol (ws-referral). http://msdn.microsoft.com/webservices/understanding/gxa, October 2001.

[Net98] Netscape Communications. Introduction to ssl. http://developer.netscape.com/docs/manuals/security/sslin/index.htm, October 1998.

[Net03] Netcraft. August 2003 web server survey. http://news.netcraft.com/, August 2003.

[NIS00] NIST. The National Institute of Standards and Technology ; Digital Signature Standard (DSS). FIPS PUB 186-2, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf, 2000.

[NN01] NN. The Secure Shell Frequently Asked Questions. http://www.ayahuasca.net/ssh/ssh-faq.html, March 2001.

[NN03a] NN. Digital signature law survey. http://rechten.uvt.nl/simone/ds-new.htm, February 2003.

[NN03b] NN. Europeans spending more online. http://www.nua.com/surveys/index.cgi?f=VS&art_id=905358734&rel=true, March 2003.

[NN03c] NN. .NET Passport Express Purchase Service Has Been Discontinued. Microsoft statement, available at: http://www.passport.net/Consumer/WalletLetter.asp, 2003.

[NN03d] NN. Object Identifiers. http://www.alvestrand.no/objectid/index.html, 2003.

[NN03e] NN. Telnet. http://www.webopedia.com/TERM/T/Telnet.html, 2003.

[NT01] Henrik Frystyk Nielsen and Satish Thatte. Web services routing protocol (ws-routing). http://msdn.microsoft.com/webservices/understanding/gxa, October 2001.

[Nyk00] Toni Nykaenen. Attribute certificates in x.509. Helsinki University of Technology, Department of Computer Science and Engineering, Helsinki, also available at: http://www.hut.fi/ tpnykane/netsec/complete/, November 2000.

[OAS03] OASIS. Organization for the advancement of structured information standards members. http://www.oasis-open.org/about/, August 2003.

[Oet02] Ryan Oettinger. Consumers worry about online privacy. Jupiter Research, http://www.jupiterresearch.com/xp/jmm/press/2002/pr_060302.html, June 2002.

[OMG03] OMG. Object Management Group: CORBA Basics. http://www.omg.org/gettingstarted/corbafaq.htm, 2003.

[ONe03] Mark ONeil. *Web Services Security.* Osborne Publishing, ISBN = 0-0722-2471-1, 2003.

[Ope98] Open Group. Introduction to single sign-on. http://www.opengroup.org/security/sso/sso_intro.htm, March 1998.

[Ope03a] OpenSSL. About the openssl project. http://www.openssl.org/about/, July 2003.

[Ope03b] OpenSSL. Howto certificates, draft version from openssl. http://www.openssl.org/docs/HOWTO/certificates.txt, July 2003.

[Opp00] Rolf Oppliger. *Security Technologies for World Wide Web.* Artech House Publishers. Norwood, MA, USA, ISBN = 1-58053-045-1, 2000.

[Opp02] Rolf Oppliger. *Internet and Intranet Security, Second Edition.* Artech House Publishers. Norwood, MA, USA, ISBN = 1-58053-166-0, 2002.

[OPT97] Donal OMahony, Michael Peirce, and Hitesh Tewari. *Electronic Payment Systems.* Artech House, Boston, www.artechhouse.com, ISBN = 0-89006-925-5, 1997.

[Ort02] Günter Orth. The web services framework: A survey of wsdl, soap and uddi. Master's thesis, Vienna University of Technology, Information Systems Institute, May 2002.

[OSM98] Koji Otani, Hiroyasu Sugano, and Madoka Mitsuoka. Capability card: An attribute certificate in xml. Internet Draft, http://xml.coverpages.org/draft-otani-ccard-00.txt, November 1998.

[Paa00] Juha Paarjarvi. Xml encoding of spki certificates. IETF, http://www.ietf.org/internet-drafts/draft-paajarvi-xml-spki-cert-00.txt, March 2000.

[Pei01] Walter Peissl. Zwischen anonymität und verlust der privatespäre – nutzungsmöglichkeiten neuer medien. Institute of Technology Assessment, Austrian Academy of Sciences, Vienna, http://www.oeaw.ac.at/ita, May 2001.

[PGP03] Pretty Good Privacy PGP. Using an x.509 pki with pgp 8.0: Protecting existing investments. PGP Corporation white paper, www.pgp.com/products/whitepapers/PGP8X509.pdf, May 2003.

[Pie00] Henna Pietiläinen. Elliptic curve cryptography on smart cards. Master's thesis, Helsinki University of Technology, Faculty of Information Technology, Department of Computer Science, 2000.

[Pin03] Denis Pinkas. Why are attribute certificates (acs) not yet in use today? Bull Services. Bull S.A., also available at: http://portal.etsi.org/STFs/documents, 2003.

[PKI03] PKIX. Public Key Infrastructure X.509. http://www.ietf.org/html.charters/pkix-charter.html, September 2003.

[PLC$^+$01] Agung Prasetijo, Mark Looi, Andrew Clark, Gary Gaskell, Paul Ashley, and Joris Claessens. Firewalling a Secure Shell Service. *Proceedings of the 2001 International Conference on Electrical, Electronics, Communication and Information, Jakarta, Indonesia - CECI2001*, March 2001.

[Pow02] Matt Powell. Understanding dime and ws-attachments. http://msdn.microsoft.com/webservices/understanding/gxa, October 2002.

[Pro01] ProComp. Passport to monopoly windows xp, passport, and the emerging world of distributed applications. http://www.procompetition.org, June 2001.

[Pro02] Hitachi Computer Products. Security for web services. http://www.quadrasis.com, August 2002.

[PS00] Joon S. Park and Ravi Sandhu. Binding identities and attributes using digitally signed certificates. Annual Computer Security Applications Conference 2000, USA, 2000.

[RAC$^+$02] Simon Robinson, K. Scott Allen, Ollie Cornes, Jay Glynn, Zach Greenvoss, Burton Harvey, Christian Nagel, Morgan Skinner, and Karli Watson. *Professional C# 2nd Edition.* Wrox Press, USA, ISBN = 1-861007-04-3, May 2002.

[Ras02] Wayne Rash. Web services are insecure. http://techupdate.zdnet.com, March 2002.

[RE99] Wolgang Rankl and Wolfgang Efing. *Handbuch der Chipkarten, Aufbau - Funktionweise Einsatz von Smart Cards.* Carl Hanser Verlag München Wien., ISBN = 3-446-21115-2, 1999.

[Rih98] Zdenek Riha. Certification. Technical report, Faculty of Informatics, Masaryk University, Brno, December 1998.

[Rin02] Martin Rinard. Operating system lecture notes: Networking. Massachusetts Institute of Technology, also available at: http://www.cag.lcs.mit.edu/ rinard/osnotes/h17.html, 2002.

[Riv92a] R. Rivest. The md4 message-digest algorithm. Network Working Group; Request for Comments: 1320 at http://www.faqs.org/rfcs/rfc1320.html, April 1992.

[Riv92b] R. Rivest. The md5 message-digest algorithm. Network Working Group; Request for Comments: 1321 at http://www.faqs.org/rfcs/rfc1321.html, April 1992.

[Riv94] Ronald L Rivest. *Cryptography, Chapter 13 of Handbook of Theoretical Computer Science, (ed. J. Van Leeuwen) vol. 1.* MIT Press., ISBN = 0262720140, 1994.

[Riv98] Ronald L. Rivest. Can we eliminate certificate revocations lists? In *Financial Cryptography*, pages 178–183, 1998.

[RMK$^+$94] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address allocation for private internets. Network Working Group; Request for Comments: 1597 at http://www.ietf.org/rfc/rfc11597.txt, March 1994.

[Ron02] Stephen A. Ronaghan. Benchmarking e-government: A global perspective; assessing the progress of the un member states. United Nations Division for Public Economics and Public Administration, also available at: http://www.unpan.org/egovernment2.asp#survey, May 2002.

[RRSY98] Ron Rivest, Matt Robshaw, Ray Sidney, and Yiqun Lisa Yin. Rc6 block cipher. http://www.rsasecurity.com/rsalabs/rc6/, August 1998.

[RSA03a] RSA Laboratories. PKCS#11: Cryptographic Token Interface Standard. http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/index.html, July 2003.

[RSA03b] RSA Security. S/MIME Frequently Asked Questions. http://www.rsasecurity.com/standards/smime/faq.html, 2003.

[Rya01] Jerry Ryan. A practical guide to the right vpn solution. http://www.techguide.com, 2001.

[San03] Aleksey Sanin. How to use xml security standards in real world. O'Reilly Open Source Convention, http://www.aleksey.com/xmlsec/, July 2003.

[Sch96] Bruce Schneier. *Applied cryptography: protocols, algorithm, and source code in C.* John Willey & Sons, Inc, ISBN = 0-471–12845-7, 1996.

[Sch99] Bruce Schneier. Twofish: A new block cipher. http://www.counterpane.com/twofish.html, 1999.

[Sch00] Jeff Schmidt. *Microsoft Windows 2000: Security Handbook.* QUE, A Division of Macmillian, Indianapolis, USA, ISBN = 0-7897-1999-1, August 2000.

[Sch02] Torsten Schmale. Web services between demand and reality. Technical report, http://www.netobjectdays.org/pdf/02/papers/industry/1492.pdf, 2002.

[Sch03a] W3 Schools. Soap tutorial. http://www.w3schools.com/soap, 2003.

[Sch03b] W3 Schools. Wsdl and uddi. http://www.w3schools.com/wsdl, 2003.

[Sch03c] W3 Schools. Xml tutorial. http://www.w3schools.com/xml, 2003.

[See02] Scott Seely. Http security and asp.net web services. Microsoft Developer Network (MSDN) Architectural Samples Team, August 2002.

[Sei99] Frank Seiliger. Opencard and pc/sc - two new industry initiatives for smartcards. http://www.opencard.org/docs/ocfpcsc.pdf, 1999.

[SET97a] SET. Secure electronic transaction specification, book 1: Business description version 1.0. SET Secure Electronic Transaction LLC , also available at: http://www.setco.org/download/set_bk1.pdf, May 1997.

[SET97b] SET. Secure electronic transaction specification, book 2: Programmers guide version 1.0. SET Secure Electronic Transaction LLC , also available at: http://www.setco.org/download/set_bk2.pdf, May 1997.

[SH02] Lutz Suhrbier and Thomas Hildmann. Pki based access control with attribute certificates for data held on smartcards. Technical University Berlin, published on HP-OVUA (http://www.hpovua.org), May 2002.

[Sho95] Adam Shostack. An overview of shttp. http://www.homeport.org/ adam/shttp.html, May 1995.

[Sid02a] Bilal Siddiqui. Exploring xml encryption. http://www-106.ibm.com/developerworks/xml/library/x-encrypt/, March 2002.

[Sid02b] Bilal Siddiqui. Uddi-based electronic marketplace. http://www.webservicesarchitect.com, 2002.

[Sie02] Siemens. Signator- das terminal für signature-chipkarten (in german language). http://www.siemens.at/signator/, May 2002.

[Sim97] Ian Simpson. Modeling the risks and costs of digitally signed certificates in electronic commerce. Carnegie Mellon University, Pittsburgh, http://www.ini.cmu.edu/NETBILL/pubs/certlife/certlife.html, 1997.

[Sko00] Aaron Skonnard. Soap: The simple object access protocol. http://www.microsoft.com/mind/0100/soap/soap.asp, January 2000.

[Sko03] Aaron Skonnard. Routing soap messages with web services enhancements 1.0. http://msdn.microsoft.com/library, January 2003.

[Sle01] Marc Slemko. Microsoft passport to trouble. http://alive.znep.com/ marcs/passport/, November 2001.

[SM98] Bruce Schneier and Mudge. Cryptanalysis of microsoft's point-to-point tunneling protocol. Proceedings of the 5th ACM Conference on Communications and Computer Security, also available at: http://www.schneier.com/paper-pptp.html, November 1998.

[SMW99] Bruce Schneier, Mudge, and David Wagner. Cryptanalysis of microsoft's pptp authentication extensions (ms-chapv2). http://www.counterpane.com/pptpv2-paper.html, 1999.

[SR03] Brent Sleeper and Bill Robins. The emerging web service market. http://www.webservicespro.com/2003/0826.html, August 2003.

[SS99] Bruce Schneier and Adam Shostack. Breaking up is hard to do: Modelling security threats for smart cards. *USENIX Workshop on Smartcard Technology*, May 1999.

[SS02] Gunjan Samtani and Dimple Sadhwani. Return on investment (roi) and web services. Technical report, Web Service Architect, www.webservicesarchitect.com, 2002.

[Ste02] Stencil Group. Web services: A new paradigm for business benefit. http://www.cotelligent.com, 2002.

[Str01] Bruno Struif. Use of biometrics for user verification in electronic signature smartcards. Proceedings Smart Card Programming and Security, E-smart 2001, Cannes, Springer-Verlag, September 2001.

[Sty02] Stylusinc Net. Soap vs. dcom & rmi/iiop. http://www.stylusinc.net/technology/soap/soap2.shtml, September 2002.

[Sun03] Sun. Staroffice 6.0. http://www.sun.com, 2003.

[Swa00] John E Swanke. *COM Programming by Example; Using MFC, ActiveX ATL, ADO and COM+*. R&D Books, Kansas USA, ISBN = 1-929629-03-6, 2000.

[Swe03] Sean Sweeney. Privacy fears on the increase. Data Protection Commissioner, http://www.dataprivacy.ie/7nr130103.htm, January 2003.

[Til00] Jim Tiller. IPSec Virtual Private Networks: A Technical Review. Lucent Technologies NetworkCare, http://www.lucent-networkcare.com, 2000.

[TJM$^+$99] Mary Thompson, William Johnston, Srilekha Mudumbai, Gary Hoo, Keith Jackson, and Abdelilah Essiari. Certificate based access control for widely distributed resources. *Proceedings of the 8th USENIX Security Symposium, Washington D.C. USA*, August 1999.

[TU02] Toshiro Takase and Naohiko Uramoto. Xml digital signature system independent of existing application. IEEE Proceedings of the 2002 Symposium on Applications and the Internet (SAINT'02), 2002.

[TY98] Chris Le Tocq and Steve Young. SET Comparative Performance Analysis. White paper from GartnerGroup and GartnerConsulting, San Jose CA, USA, also available at http://www.setco.org/download/setco6.pdf, Novenmber 1998.

[UDD00] UDDI. Universal Description Discovery Integration (UDDI) Technical White Paper. Technical report, UDDI Org, September 2000.

[Use03] Userland. Xml-rpc home page. www.xmlrpc.com, 2003.

[Vas01] Clemens Vaster. Why soap doesn't lack security while it does. *newtelligence Group, http://www.newtelligence.com*, 2001.

[VCF$^+$00] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. Aaa authorization framework. Network Working Group; Request for Comments: 2904 at http://www.ietf.org/rfc/rfc2904.txt, August 2000.

[vdPK00] Ton van der Putte and Jeroen Keuning. Biometrical fingerprint recognition: Don't get your fingers burned. Proceedings of: IFIP TC8/WG8.8 Fourth Working Conference on Smart Card Research and Advanced Application pp.289-303, September 2000.

[Ver02a] VeriSign. Building an e-commerce trust infrastructure: Ssl server certificates and online payment services. http://www.verisign.com/resources/gd/buildEcommerce/buildEcommerce.html, 2002.

[Ver02b] VeriSign. Online payment processing – what you need to know. http://www.verisign.com/resources/gd/enablePayment, 2002.

[Ver03] VeriSign. What every merchant should know about internet fraud. http://www.verisign.com/resources/gd/internetFraud/internetFraud.html, 2003.

[Vol01] Ken Vollmer. Uddi's problem: Technology cannot replace relationships. http://www.internetweek.com/columns01/beat062001.htm, June 2001.

[Vor03] Vordel Company. Firewalls and web services - myths and facts. http://www.vordel.com/knowledgebase/vordel_view3.html, 2003.

[W3C00] World Wide Web Consortium W3C. Simple object access protocol (soap) 1.1. http://www.w3.org/TR/SOAP/, May 2000.

[W3C01] World Wide Web Consortium W3C. Xml schema part 2: Datatypes. http://www.w3.org/TR/xmlschema-2/, May 2001.

[W3C02] World Wide Web Consortium W3C. Web services activity statement. http://www.w3.org/2002/ws/Activity.html, June 2002.

[W3C03] World Wide Web Consortium W3C. Soap version 1.2. http://www.w3.org/TR/2003/PR-soap12-part0-20030507/, May 2003.

[Was02] Martin Wasznicky. Using web services instead of dcom. Microsoft Corporation, http://www.microsoft.com, USA, February 2002.

[Wed03a] Wedgetail Communications. Smart card authentication for j2ee applications using jcsi sso – a white paper. http://www.wedgetail.com, October 2003.

[Wed03b] Wedgetail Communications. Web service single sign-on using j2ee and .net – a white paper. http://www.wedgetail.com/whitepapers/web_services/whitepaper_web_services.pdf, July 2003.

[Wes02] Westbridge Technology. Deploying xml web services accelerates security trends. http://www.westbridgetech.com, 2002.

[Wha02] David Whalen. The unofficial cookie faq. http://www.cookiecentral.com/faq/, August 2002.

[Wil01] Lawrence Wilkes. Web services – right here, right now delivering web services today with ibm solutions. CBDi Forum, December 2001.

[Wil02] Joe Wilcox. Credit card theft feared in windows flaw. CNET news.com, USA, September 6 2002.

[Win99] Dave Winer. Xml-rpc specification. http://www.xmlrpc.com, June 1999.

[Win02a] Eric Winsborrow. Integrated security: A new network approach. http://www.technologyevaluation.com, December 2002.

[Win02b] Winter Green Research. Web services markets: Market strategies, opportunities, and forecasts 2002-2007. 6 Raymond Street Lexington, MA 02421 USA, http://www.wintergreenresearch.com, 2002.

[Wol01] Roger Wolter. Xml web services. Microsoft Corporation, http://www.microsft.com, 2001.

[Wor97] PC/SC Workgroup. Interoperability specification for iccs and personal computer systems, part 1. introduction and architecture overview. http://www.pcscworkgroup.com, December 1997.

[WS96] David Wagner and Bruce Schneier. Analysis of the ssl 3.0 protocol. The Second USENIX Workshop on Electronic Commerce Proceedings, USENIX Press, also available at: http://www.counterpane.com/ssl.html, November 1996.

[Xen00] Symeon Xenitellis. The open - source pki book: A guide to pkis and open - source implementations. http://ospkibook.sourceforge.net/, 2000.

[Yeo03] Lisa Yeo. *Personal Firewalls for Administrators and remote Users.* Prentice Hall, New York, USA, ISBN = 0-13-046222-5, 2003.

[YKMY01] Hideo Yamamoto, Testsutaro Kobayashi, Masahiro Morita, and Ryuji Yamada. Public key based high speed payment (electronic money) system using conatct less smart cards. *International Conference on Research in Smart Cards, E-smart 2001*, September 2001.

[YM03] T. Ylonen and D. Moffat. SSH Transport Layer Protocol. Network Working Group, Internet-Draft Expires: March 31, 2004, available at: http://www.ietf.org/internet-drafts/draft-ietf-secsh-transport-17.txt, October 2003.

[ZBL03] Adam Zilinskas, Morris Brown, and Brian Loomis. *Securing B2B XML Web Services with WSE.* Microsoft Pres, online book, also available at: http://www.elearning4gurus.com/only4gurus/techlib/microsoft/wse_biztalk.pdf, January 2003.

[Zim00] Phil Zimmermann. An introduction to cryptography. http://www.pgpi.org, 2000.

Curriculum Vitae

Personal Data:

Name	Blerim
Surname	Rexha
Birthday	1.5.1970
Birthplace	Prishtinë, Kosova
Address	Favoritenstr. 33/2/16 A-1040 Wien

Education:

1976 - 1984	Primary School, Prishtinë
1984 - 1988	Mathematical Secondary School, Prishtinë
1989 - 1994	University of Prishtina, Faculty of Electrical Engineering, Prishtinë
1995 - 1997	German Language and examinations at the Vienna University of Technology, Vienna
1997 -	PhD studies in the Institute of the Computer Technology, Vienna University of Technology, Vienna

Employment:

Feb. 1995 - Oct. 1995	Assistant at the University of Prishtina, Prishtinë
Mar. 1995 - Oct. 1995	Developer engineer at CST Company, Prishtinë
Jun. 1999 - Sep. 2000	Software developer at PDTS GmbH, Vienna
Sep. 2000 -	Technical consultant and software developer at Siemens AG, PSE division, Vienna