

Authentication, Authorization and Resource Reservation for Distributed Laboratories

Diplomarbeit

der Philosophisch-naturwissenschaftlichen Fakultät

der Universität Bern

vorgelegt von

Thomas Jampen

2002

Leiter der Arbeit:

Prof. Dr. Torsten Braun

Institut für Informatik und angewandte Mathematik

Leiter der Arbeit:

Prof. Dr. Torsten Braun

Institut für Informatik und angewandte Mathematik
Rechnernetze und Verteilte Systeme

Betreuer der Arbeit:

Marc A. Steinemann

Institut für Informatik und angewandte Mathematik
Rechnernetze und Verteilte Systeme

Zusammenfassung

Im Rahmen des *Swiss Virtual Campus (SVC)* wird ein Projekt mit dem Namen *Virtual Internet and Telecommunications Laboratory of Switzerland (VITELS)* durchgeführt. An diesem Projekt sind mehrere Universitäten und Ingenieurschulen beteiligt, um gemeinsam ein virtuelles Labor für Praktika im Bereich der Rechnernetze aufzubauen. Im Gegensatz zu herkömmlichen Laborarbeiten bietet dieses virtuelle Labor den Studenten Hochschulen die Möglichkeit, via Internet auch von zu Hause aus darauf zuzugreifen und somit bequem ihre Kenntnisse zum Thema Computernetzwerke zu vertiefen. Die verschiedenen Versuche, die als ganzes das Projekt VITELS bilden, werden im Moment unabhängig von einander an den verschiedenen Hochschulen entwickelt.

In einer ersten Phase dieser Diplomarbeit wurden existierende Sicherheitsarchitekturen (wie z.B. PKI und Kerberos) analysiert und auf ihre Tauglichkeit für das VITELS Projekt getestet. Das wichtigste Ziel war, den Zugriff auf die Laborhardware nur autorisierten Teilnehmern zu gewähren. Da bis jetzt jede Hochschule ihre eigene Studentendatenbank verwaltet hat, musste eine zentralisierte Datenbank aufgebaut oder eine Möglichkeit gefunden werden, bereits existierende Datenbanken auf geeignete Art und Weise zu verbinden. Eine besondere Schwierigkeit stellten die beschränkten Ressourcen (z.B. Router) des Praktikums dar, die nicht von mehreren Studenten gleichzeitig konfiguriert werden können. Deshalb musste in einer zweiten Phase ein Online Terminplanungssystem (Scheduling) implementiert werden, das die Ressourcen verwaltet und registrierten Studenten die Möglichkeit bietet, sich für bestimmte Stunden für ein spezielles Modul einzuschreiben. Während eines solchen Timeslots hat der eingeschriebene Student das alleinige Recht, auf die Hardware zuzugreifen.

Es hat sich herausgestellt, dass eine Public Key Infrastruktur (PKI) die am besten geeignete Architektur für das VITELS Projekt ist. Die hier vorgeschlagene - und zusammen mit der Diplomarbeit "Gateway für entfernte Praktika" in die Praxis umgesetzte - Architektur benutzt nur kryptographisch gesicherte Verbindungen (z.B. mittels IPsec, SSL oder SSH) zwischen den Clients und den Servern, um die persönlichen Daten der Studenten (wie z.B. Benutzernamen und Passwörter) zu schützen. Für den Zugriff auf die Studentendatenbanken wird das Lightweight Directory Access Protocol (LDAP) verwendet. Das implementierte Reservationssystem erlaubt es den Hochschulen, entweder, die benötigten Studentendaten im zentralen VITELS Verzeichnis einzutragen, oder aber, einen eigenen Server zu verwalten und diesen mit dem zentralen Server durch einen gesicherten Tunnel zu verbinden. Als Benutzerschnittstelle zu diesem Reservationssystem dienen in PHP geschriebene Webseiten, welche einerseits den Studenten erlauben, online Timeslots zu reservieren und wieder freizugeben und andererseits den Moduladministratoren ermöglichen, solche Timeslots zur Verfügung zu stellen, Moduleinstellungen zu ändern und das Registrierverhalten der Studenten zu kontrollieren.

Abstract

Within the scope of the *Swiss Virtual Campus (SVC)* a project called *Virtual Internet and Telecommunications Laboratory of Switzerland (VITELS)* is implemented. Several universities and engineering schools are involved in this project in order to build a virtual laboratory where students can improve their skills in the realm of computer networks. The different modules form a common online course but are developed independently by the participating universities. In contrast to conventional laboratories, the modules provided by the VITELS project are meant to be online available and, thus, can be solved from every computer connected to the Internet.

First of all, existing security architectures (e.g. PKI, Kerberos) have been evaluated relating to the needs of the VITELS project. An important goal was to provide access to the laboratory resources only to authorized participants. Each university can maintain its own student database needed for the VITELS authentication. As a consequence, a common student database had to be built or a common interface for accessing each university's student database had to be found. A major problem was the fact that, unfortunately, the hardware resources for such an online laboratory are limited. Therefore, an online timetable and an underlying scheduling script has been implemented in order to provide the means for reserving timeslots during which only one student can access a certain module.

The most appropriate security architecture has emerged to be a Public Key Infrastructure (PKI). The developed architecture only uses cryptographically secured connections (e.g. with IPsec, SSL or SSH). The proposed architecture uses the Lightweight Directory Access Protocol (LDAP) in order to access the student database. The implemented scheduling system allows the universities to deploy and maintain their own student directories that are accessed when authenticating students. Furthermore, it offers an easy to use online user interface that allows to reserve or to free timeslots, and - to the module administrators - it provides the means to add and remove timeslots, to change module settings, and to control the registration behavior of the students.

My diploma thesis is dedicated to Christine.

Acknowledgements

First of all, I would like to thank everybody who contributed to the success of my diploma thesis. Especially, I would like to thank Prof. Dr. T. Braun who gave me the possibility to carry out my diploma thesis in his research group. Furthermore, I would like to express my gratitude to Marc-Alain Steinemann who coordinated my diploma work with the rest of the VITELS project. He gave me lots of hints and valuable recommendations and sacrificed a lot of his spare-time in order to proof-read my documentation. I would also like to thank Matthias Scheidegger whose door has always been open when I ran into trouble, Stefan Zimmerli who helped me a lot to proceed my work with numerous interesting and fruitful discussions and Christian Heim, Peter Geiser, Roland Trummer and Robert Casties from the Informatikdienste (ID) who helped me with the development of the student directory structure and the integration into the productive system. Last but not least, I would like to thank my girl-friend who has always been patient and insightful when I had been working until late in the night or over the weekend.

Contents

1	Introduction	1
2	Basic Technologies	5
2.1	Lightweight Directory Access Protocol (LDAP)	5
2.1.1	Attribute Types and Object Classes	6
2.1.2	Schema Files	6
2.1.3	Searching LDAP Entries	8
2.1.4	Adding / Modifying / Deleting LDAP Entries	9
2.1.5	Filters	9
2.1.6	Aliases	9
2.1.7	Referrals	10
2.2	Authentication, Authorization, Accounting	11
2.3	Kerberos	11
2.4	Public Key Infrastructure (PKI)	12
2.4.1	Secure Sockets Layer (SSL) / Transport Layer Security (TLS)	13
2.4.2	Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)	14
2.4.3	Secure Shell (SSH)	15
2.4.4	Internet Protocol Security (IPsec)	15
3	Architecture	17
3.1	Security Architecture: Kerberos versus Public Key Infrastructure	18
3.1.1	Advantages and Disadvantages of Kerberos	18
3.1.2	Advantages and Disadvantages of a PKI	19
3.2	Proposed Security Architecture	20
3.3	Directory Structure	21
3.3.1	Student Entries	23
3.3.2	Timetable	24
3.3.3	Modules	24
3.3.4	Staff	25
4	Implementation	27
4.1	Scheduling Script	27
4.1.1	LDAP Queries	27
4.1.2	LDAP Locking Mechanism	29

4.1.3	LDAP Configuration	30
4.2	User Interface to the Scheduling Script	31
4.2.1	Slot Reservation and Module Administration	31
4.2.2	User Interface Configuration	32
4.3	Cron Jobs	32
4.3.1	Updating Each Module's Current User	33
4.3.2	Removing Expired Timeslots	33
4.4	VITELS Staff Directory	34
4.5	PHP-based API for Accessing a Module's Current User	35
5	Configuration and Usage	37
5.1	LDAP Server Configuration	37
5.1.1	VITELS Schema File	37
5.1.2	Slapd Configuration File	39
5.2	Web-based User Interface	41
5.2.1	Online Timetable and Administration Interface	41
5.2.2	VITELS Staff Website	44
5.2.3	VITELS Staff Address Book	46
6	Integration of Security Architecture and Laboratory Hardware	49
6.1	Apache Authentication Based on LDAP	49
6.1.1	Installation	50
6.1.2	Apache Configuration	50
6.1.3	.htaccess Files	51
6.1.4	LDAP Directives	51
6.1.5	Usage of auth_ldap for the VITELS Project	52
6.2	Using the API for Accessing a Module's Current User	53
6.3	LDAP-based Shadow-File Modification	55
7	Related Work	57
7.1	Shibboleth	57
7.2	PAPI	57
7.3	GASPAR	58
7.4	FEIDHE	58
8	Conclusions and Outlook	59
8.1	Conclusions	59
8.2	Outlook	60
8.2.1	Reservation Limitations	60
8.2.2	Multiple LDAP Servers	61
8.2.3	Certificate Authorities	62
8.2.4	VITELS for Students all Over the World	62

List of Figures	65
Abbreviations	67
Glossary	69
Bibliography	73

1 Introduction

The medium of the late nineties and the beginning of the new century seems to be the Internet. Almost everyone has Internet access - either at home, at work or at one of many Internet cafes. Computer Science is considered to be important enough for being taught already at elementary school. Our pupils' generation is a generation where almost everyone's parents own a computer. Thus, the need arose of establishing or improving the possibilities of online or *Distance Learning*. Distance Learning, also known as Distance Education, is simply learning from a distance, usually from home, or from any conveniently located off-campus place. Distance Learning generally makes use of the Internet, television, video cassettes or audio tapes and the mailbox, to deliver instructions, theory and exercises. Distance Learning also refers to on-campus classes where the professor is not physically present, but communicating with students at several places simultaneously via television, Internet, or some other electronic means. This work concentrates on online Distance Learning, which is based on the Internet or World Wide Web (WWW).

Nowadays, pupils and students most often use the Internet for their spare-time activities and not for learning, because the online available material is not presented appropriately. As a consequence, a lot of work has to be done in order to make Distance Learning more attractive and efficient. A common problem with new technologies is the fact that, at the beginning, solutions increasing the use of such technologies are not adequate enough. Adopted to the Internet this means that under online Distance Learning one cannot understand reading regular books online neither is it appropriate offering just lecture notes or slide-shows on the Internet. Online Distance Learning is totally different from learning we are used to. New concepts have to be found and evaluated and known didactical methods have to be adapted to fit the needs of online learning.

The Swiss Ministry of Education and Science recognized the problems mentioned above and tried to coordinate the steps by funding a federal program called *Swiss Virtual Campus (SVC)* [5]. The aim of the SVC is to improve the quality of online learning and interactive teaching by providing web-based courses and exercises. Members of the SVC are the two Swiss federal institutes of technology, Swiss universities of applied sciences and Swiss engineering schools. The SVC was funded with three main goals in mind. First of all, the SVC intends to improve the quality of student learning processes and strengthening interactive teaching. The students should be encouraged to use all the information and resources available on the Internet as a part of their real studies. Second, the SVC aims to improve the collaboration between the universities, that is why several institutions are involved in each project. The third goal is to develop high-quality teaching material and adapted methods appropriate for web-based learning.

The SVC initialized several projects for different subjects where each project should develop a course that can be followed via the Internet and which includes teaching material, exercises,

seminars or practical work as well as online help. One of these projects is the *Virtual Internet and Telecommunications Laboratory of Switzerland (VITELS)* [6]. The goal of this project is to make online Distance Learning more attractive by providing a virtual computer network laboratory which can be accessed by the students via the Internet. Instead of working with several computers and routers in a real laboratory, the VITELS project intends to offer the possibility to setup, configure and analyze the necessary hardware from the student's own computer. Such a laboratory is often called *remote laboratory* because its users can operate from remote computers and do not have to be at the same place where the laboratory equipment is located. The lab work developed in this project intends to provide to students that have attended computer network lectures the means to apply their theoretical knowledge in practical exercises. It is also planned to provide supplementary theoretical material in addition to the practical modules. The VITELS project tries to add a new dimension to Distance Learning by stimulating online exercises and on-line laboratory work instead of just providing online available theory [2]. Furthermore, a remote lab has the advantage of offering a much more flexible usage because the access is not restricted to office hours.

"Each partner of the VITELS project - four universities (Bern, Fribourg, Genève and Neuchâtel) and one engineering school (Fribourg) - is currently developing modules based on the own competence and equipment. The seven modules focus on Linux System Installation and Configuration, IP Network Simulation, Configuration and Performance Evaluation of a Real IP Network, Client/Server Programming, Protocol Analysis, IP Security, and Firewall Management" [1]. Based on these seven modules a course of one semester duration can be implemented, in which - at the University of Bern - the students can spend twelve hours for preparing, performing and evaluating a single module. Naturally, this course can be extended by additional modules. This would allow to select certain modules depending on the particular needs of each university.

An important aspect of this university spanning project is the user and module data management. Students from different universities need to be able to access all modules from their own and from other universities' computers. Thus, all involved educational institutes need to agree on a concerted way of storing student data. The here-proposed architecture uses the Lightweight Directory Access Protocol (LDAP) in order to access the student directories. LDAP is a very powerful but still developing standard currently also deployed by SWITCH [7]. Most probably the directories of Swiss universities will be linked into the federal directory tree - developed and maintained by SWITCH - in the near future.

A second aspect to be considered was the security architecture. Already existing architectures (such as PKI and Kerberos) had to be evaluated. The most appropriate security architecture had to be chosen based on the needs for the VITELS project but also relating to the development done by SWITCH which influences the research of the universities, too. The evaluation has shown that a PKI seems to be the most suitable architecture for the VITELS project. There are a lot of protocols (e.g. IPsec, SSL and SSH) that allow to secure network connections in order to protect the student data (e.g. usernames and passwords).

Another problem that had to be solved originated in the fact that some course modules offer the possibility of configuring real hardware. Unfortunately, the amount of available hardware is limited and, usually, the same hardware cannot be accessed and configured by different students

at the same time. This implies that there have to be scheduling functions that allow students from different educational institutes to reserve certain timeslots. The fact that it is a remote laboratory raises the need for an online accessible timetable with an underlying script that controls and grants the access to the different modules. The underlying scheduling mechanism needs to maintain a database linked to every universities' student database. Students need to login with a username and a password stored on their home university's database server. The scheduling script needs to be able to verify the login and mark potential reservations within its own database.

The goal of this diploma thesis was to deal with the above described problems and - in a first phase - to evaluate possible solutions and potential security architectures. In a second phase, the most suitable security architecture had to be partly built up and the scheduling mechanism had to be implemented. In order to make it as convenient as possible for the students and to allow them to reserve certain hardware for a specified duration, an online timetable that communicates with the underlying scheduling script was programmed. After the described architecture and the scheduling system had been developed and tested in a test network, the whole system has been successfully transferred into the productive environment of the University of Bern. Later, a real-world test for students of the University of Bern has been performed successfully using the "IP Security" module as an example.

The structure of this paper is as follows: In the second Chapter the involved technologies will be introduced and explained. The third Chapter presents the elaborated architecture for the VITELS project. The implementation is described in the fourth Chapter. Chapter five points out different LDAP configuration aspects, shows screenshots of the web-based user interface and explains its usage. The sixth Chapter describes the integration of the chosen security architecture and the laboratory hardware with the example of the "IP Security" module of the VITELS course. In Chapter seven related products are mentioned and compared to the here-presented architecture, while Chapter eight concludes this diploma thesis and gives a short outlook.

2 Basic Technologies

2.1 Lightweight Directory Access Protocol (LDAP)

The Lightweight Directory Access Protocol (LDAP) [11] is a lightweight client/server protocol for accessing a directory service. It was initially used as a front-end to X.500, but can also be used with stand-alone directory servers. LDAP is optimized for read access and, thus, is usually deployed in environments where data is read very often but hardly changed. An Open Source [8] project called OpenLDAP [17] offers a free LDAP implementation.

In order to import and export directory information to or from LDAP-based directory servers, or to describe a set of changes which are to be applied to a directory, the LDAP Data Interchange Format (LDIF) is used. LDIF files store their information in an object-oriented hierarchy. Sample LDIF entries could look like this:

```
dn: c=CH
objectclass: top
objectclass: country
c: CH

dn: o=Company,c=CH
objectclass: organization
o: Company

dn: cn=Max Muster,o=Company,c=CH
objectclass: person
cn: Max Muster
sn: Muster
telephonenumber: +41316313312
```

Figure 2.1: Sample LDIF Entries

The first four lines define an entry for the country Switzerland, the second group of lines determines an organization with the name `Company` within the country Switzerland. The third entry specifies one employee called `Max Muster` of this company. Additional information like the telephone number can be saved as well. The most important line of each entry is the line beginning with `dn`, this line represents the hierarchy or - in other words - the position within the hierarchy. For a detailed technical specification of the LDIF format see the Request for Comments (RFC) 2849 [12].

2.1.1 Attribute Types and Object Classes

As Figure 2.1 shows, an LDAP entry is a group of attribute-value pairs. The most important attribute is the distinguished name `dn`.

Definition 1 *A distinguished name (dn) uniquely identifies an entry within the hierarchical directory. The dn consists of the entry's name plus the path (list of entry names) back to the top of the tree.*

Each `dn` must be unique, because it allows you to find the corresponding entry within the tree-like directory structure. From right to left a `dn` contains all the entry's names leading from the top level to the desired entry. Thus, the top level of the example hierarchy is the country, followed by one (or more) organization within this country. Each organization can have subtrees like for example employees. The directory data is represented as attribute-value pairs. Any information that is to be stored is associated with a descriptive attribute (Figure 2.2).

```
dn distinguished name
cn common name
sn surname
o organization
ou organizational unit
dc domain component
```

Figure 2.2: A List of Common Attributes and Abbreviations

When talking about LDAP, there are two important terms: *object class* and *attribute type*.

Definition 2 *An attribute type defines the data representation of an attribute. It defines how the associated values are to be compared and ordered. Furthermore, an attribute type allows you to define, whether an attribute can have just one single value or a set of values.*

Definition 3 *An object class defines the collection of attributes that can be used to define an entry. These attributes are divided into two groups. The first group specifies attributes that must be used in order to define a valid entry, the second group lists additional attributes that can be used to characterize the entry.*

An entry can belong to more than one object class, but must provide all required attributes from all object classes. The entry can assign additional attributes that are listed as optional within at least one of the object classes. All entries require the attribute `objectclass` that specifies to which object class(es) an entry belongs.

2.1.2 Schema Files

Schema files are used in order to store the different object classes and attribute types that are available. There are some existing schema files like for example the `core.schema` and the `cosine.schema` file. The `core.schema` file defines the LDAP schema items specified in

RFC 2251 - 2256 [13]. The `cosine.schema` file attempts to describe RFC 1274 [14]. These are the essential schema files that include the most common object classes (e.g. `country`, `organization`, `person`, `account`, etc.) and attributes (e.g. `c`, `o`, `ou`, `cn`, `sn`, `uid`, `userpassword`, etc.). The attribute type definition for the attribute `name` looks as shown in Figure 2.3.

```
attributetype ( 2.5.4.41 NAME 'name'
                DESC 'The name of a person or any object'
                EQUALITY caseIgnoreMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
```

Figure 2.3: Definition of the Attribute `name`

The attribute is given a globally unique number - so-called Abstract Syntax Notation number One (ASN.1) [16] - followed by a name. For better readability, a description can be defined (DESC parameter). Then, one has to specify how attributes of this type are compared. This is done with the help of the EQUALITY parameter. There are several possibilities for comparing entries such as matching numbers, matching character (case-sensitive or case-insensitive). In this example, the case is ignored when comparing the values. The parameter SUBSTR defines how substrings are compared. There are the same possibilities as for comparing the whole entry. Finally, the parameter SYNTAX defines the type of the value to be saved. This large number specifies a case-ignore string, but it could also be a case-exact string, an integer or binary data (e.g. for saving pictures), etc. After that, between curly brackets, it is possible to define a maximum number of characters to be stored.

An object class definition is shown in Figure 2.4. It represents the definition of the object class `person`.

```
objectclass ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL
              MUST ( sn $ cn )
              MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )
```

Figure 2.4: Definition of the Object Class `person`

Like attributes, every object class has its own unique ASN.1 number followed by the parameter NAME and the name of the object class. It is possible to specify another object class from which the current object class inherits. That is done by using the parameter SUP. An LDAP entry cannot contain more than one object class that is defined to be STRUCTURAL. This is used in order to prevent entries from being for example a `person` and an `organization` at the same time. There are two lists of attributes, the first specifies the attributes an entry must contain in order to be valid and the second list defines attributes an entry is allowed to contain.

2.1.3 Searching LDAP Entries

LDAP searches can be influenced in many ways.

server/port First of all, the LDAP server and - if necessary - a port can be specified. The default LDAP port is 389. If no server is specified, `localhost` is used.

base dn A base dn can be specified in order to limit the search to the specified directory subtree.

scope The scope of the search can be influenced by choosing either `base`, `one` or `sub`. `base` searches just the base object, which is the object specified by `base dn`. The second possibility is to perform a `one level` search, which considers the base object and all objects at the next level. And `sub` is the default and searches the whole subtree of the base object.

sort The sort option allows to specify an attribute that is to be used for sorting the results.

bind dn/bind pwd Sometimes, anonymous binding to the LDAP server is not allowed. This means that a valid dn and the corresponding password must be supplied in order to retrieve data.

time limit It is possible to specify a time period in seconds one is willing to wait for the results. If the period is over, the search is aborted.

size limit The size limit can be used in order to receive at most the specified amount of entries.

attributes only Sometimes, one just needs to know whether an attribute is present in an entry but there is no need the get its value. This option allows to limit the results to the attributes only.

aliases LDAP allows aliases. An alias is a pointer to another entry. This parameter specifies how alias dereferencing is done. The possible values are `never`, `always`, `search` and `find`. This means that aliases can be dereferenced `never`, `always`, when searching the subtree but not when locating the base object or only when locating the base object but not when searching the subtree.

chase referrals This option can be used in order to automatically chase referrals. A referral is an alias to a directory branch on another server (see Section 2.1.7).

filter A valid filter can be specified (see Section 2.1.5). If no filter is provided, the default `"(objectclass=*)"` is used.

attributes It is possible to specify a list of attributes that are returned, if entries are found. It is possible to search for a user Smith, but as a result it is only needed to know his telephone number. As a filter one would use `"(sn=Smith)"` and the attribute list would contain just the attribute `telephonenumber`.

2.1.4 Adding / Modifying / Deleting LDAP Entries

First of all, the appropriate LDAP server and the port the daemon is running on have to be specified. In order to add, modify or delete entries, a bind dn and the corresponding password have to be provided unless the LDAP server is configured to allow anonymous changes which is very unlikely. These parameters are described above (see Section 2.1.3).

The entries to be added or changed have to be provided in standard LDIF format. Entries to be deleted are to be specified simply by their dn.

2.1.5 Filters

After defining a base dn, the search can be further limited by providing a filter that is applied to every entry below the base dn. If the filter matches, the entry is returned as a result. A simple filter could be `"(cn=Thomas Jampen)"`. A search using this filter would result in entries containing the attribute `cn` and the corresponding value `Thomas Jampen`. This can be used in order to search for persons, phone numbers, etc.

Instead of providing the full value of an attribute, it is possible to use the `*` in order to allow any characters. The filter `"(sn=Me*er)"` finds all users called *Meier*, *Meyer*, *Meister*, etc. It is even allowed to specify just an attribute followed by `*` - e.g. `"(mail=*)"` - this returns just entries that contain this attribute.

Combining filters is possible by connecting them either with a logical AND ("`&`") or with a logical OR ("`|`"). Searching every entry for a user *Meyer* that lives in *Bern* can be done with the filter `"(&(sn=Meyer)(location=Bern))"`. If all users having an email address or a fax number have to be found, the appropriate filter is `"(|(mail=*)(fax=*))"`. For further details please consult the RFC 2254 [15].

2.1.6 Aliases

Aliases are used when an already existing directory entry should be present in a different location within the same directory as well. Assuming that an employee works in two different departments of the same company at the same time. He has already got an entry for the department A (`dn: cn=Max Muster,ou=DeptA,o=Company,c=CH`) and needs to have the same entry for department B. It is not necessary to copy his data and, thus, to maintain two identical entries for him. The entry for the department B can be an alias to the the entry of department A (Figure 2.5). There is an object class `alias` that has one mandatory attribute called `aliasedobjectname` which takes the dn of the original entry as its value. Aliases are not dereferenced by the server. It is the client that needs to do the dereferencing (see Section 2.1.3).

```
dn: cn=Max Muster,ou=DeptB,o=Company,c=CH
objectclass: alias
aliasedobjectname: cn=Max Muster,ou=DeptA,o=Company,c=CH
```

Figure 2.5: An Alias Entry

2.1.7 Referrals

It can be necessary to include a directory branch which is stored on another LDAP server into an existing directory tree. The subtree that should be included does not need to be copied. It is possible to place a special entry in the directory of one server linking to a subtree of another server. Such a link is called *referral*. A referral is similar to an alias, just that it links to an entry on another server.

A company, for example, has an LDAP server that stores the data of the employees working at branch A. Another branch B of this company has its own LDAP server with the data of its employees. The structure on the two servers might look as depicted in Figure 2.6. At branch A, the entry shown in Figure 2.7 can be added in order to link the two servers.

```
dn: o=Company,c=CH
objectclass: organization
o: Company

dn: ou=BranchA,o=Company,c=CH
objectclass: organizationalunit
ou: BranchA

dn: cn=Max Muster,ou=BranchA,o=Company,c=CH
objectclass: person
cn: Max Muster
sn: Muster
telephonenumber: +41123456789
```

```
dn: o=Company,c=CH
objectclass: organization
o: Company

dn: ou=BranchA,o=Company,c=CH
objectclass: organizationalunit
ou: BranchA

dn: cn=Anna Feer,ou=BranchA,o=Company,c=CH
objectclass: person
cn: Anna Feer
sn: Feer
telephonenumber: +41987654321
```

Figure 2.6: Two Branches of a Company with Separate LDAP Servers

```
dn: ou=BranchB,o=Company,c=CH
objectclass: referral
objectclass: extensibleobject
ref: ldap://ldap.branchb.company.ch/ou=BranchB,o=Company,c=CH
```

Figure 2.7: A Referral Entry

The referral entry contains the object classes `referral` and `extensibleobject` and, additionally, the attribute `ref` containing an URL specifying the other server and the desired dn. At the moment, the LDAP server implementation does not allow to chase the referrals automatically, this can only be done by the client (e.g. the `ldapsearch` tool). For future LDAP versions it is planned to include this functionality into the server.

2.2 Authentication, Authorization, Accounting

The VITELS project provides a remote laboratory to students from different educational institutes all over Switzerland. In order to control the access to the limited hardware resources, an efficient *authentication*, *authorization* and *accounting* (AAA) architecture needs to be implemented.

Definition 4 *Authentication is the process of determining whether someone is, in fact, who he claims to be.*

Authentication is needed in order to guarantee that only registered students of the participating institutes can access the online courses.

Definition 5 *Authorization is the process of giving someone permission to do something.*

After a successful authentication phase the student is given access to some of the resources based on an online timetable where students can reserve timeslots for accessing certain parts of the lab.

Definition 6 *Accounting is the process which measures the resources a user consumes or plans to consume during his session. Accounting is used for authorization control, billing, trend analysis and capacity planning activities.*

Billing and trend analysis do not have priority at the moment but could be an issue later on when access to the lab is no longer restricted to registered students (see Section 8.2.4). Due to the limited resources, accounting has to be used for authorization control. This means that students cannot access the whole lab all at once but are just given access to a certain module based on their online reservations.

2.3 Kerberos

Kerberos has been developed at the Massachusetts Institute of Technology (MIT). The name Kerberos originates from Greek mythology, it is the three-headed dog that guarded the entrance to Hades. The MIT offers a free implementation of this protocol on its homepage [20].

Definition 7 *Kerberos is a network authentication protocol that is designed to provide strong authentication for client/server applications by using secret key cryptography.*

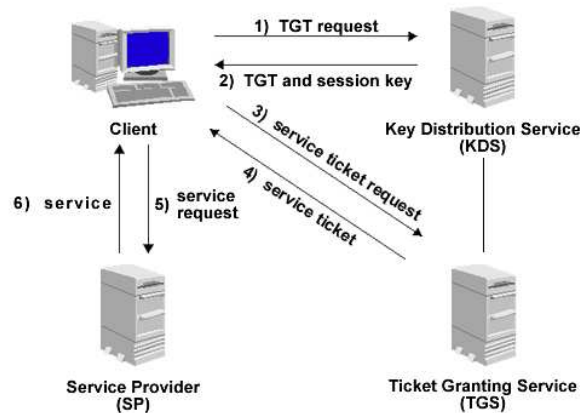


Figure 2.8: Kerberos Overview

Figure 2.8 shows the involved parties in a Kerberos system. A user at a client intends to get a certain service from a service provider. The following happens:

The client sends a Ticket Granting Ticket (TGT) request to the Key Distribution Center (KDC) (connection 1). The KDC returns a TGT and a corresponding session key - that allows the client to use this TGT - encrypted using a key generated from the user's password (connection 2). All user passwords are stored in a central database of the KDC. After that, the user at the client computer is prompted for his password. His password is used to compute the key that is able to decrypt the session key received from the KDC. If the user entered the correct password he obtains the TGT and its associated key. Usually, this entire process already happens when the user logs into the client.

After receiving the TGT, the user can demand a service from a service provider. The client asks the Ticket Granting Service (TGS) for a service ticket by sending the TGT and a service request to the TGS (connection 3). The TGS looks in its master database for an entry for the client and the requested service. If the entry exists, the TGS issues and returns a ticket for this service (connection 4).

The client sends this service ticket to the service provider, that verifies the ticket using its own service key (connection 5). If the ticket is valid, the service provider now knows the identity of the user at the client computer and is able to provide the service (connection 6).

2.4 Public Key Infrastructure (PKI)

In the physical world, face-to-face transactions, photo identification and even written signatures offer some protection against fraud. However, the Internet remains relatively anonymous, making it harder to know who is at the other end of the network.

The challenge is to translate the trust conventions from the physical to the online world. A Public Key Infrastructure (PKI) has become the de facto standard for establishing this trust over electronic networks.

Definition 8 *A Public Key Infrastructure (PKI) is a system of digital certificates and Certificate Authorities that verify and authenticate the validity of each involved party.*

Definition 9 *A Certificate Authority (CA) is an authority in a network that issues and manages security credentials and public keys for message encryption and signature verification.*

Definition 10 *A digital certificate consists of the public key and the identity of an entity, rendered unforgeable by digitally signing the entire information with the private key of the issuing Certificate Authority (CA).*

The name Public Key Infrastructure is used because Certificate Authorities issue digital certificates by signing public keys.

Definition 11 *A public key is a value that can be used to effectively encrypt messages and verify digital signatures.*

The public key can be made publicly available, it does not contain secret information. All secret information is stored within the corresponding private key.

Definition 12 *A private key is a value - known only to one party - that can be used to decrypt encrypted messages, issue digital signatures and compute the corresponding public key.*

The private key must be kept private and must not be made publicly available. Together, a private and a public key form a key pair.

The most important services of a Certificate Authority are:

- **Key Registration** Issuing a new certificate for a public key after verifying the identity of the person demanding the certificate.
- **Certificate Revocation** Canceling a previously issued certificate. This is usually done when a private key is corrupted.
- **Key Selection** Obtaining a party's public key. The CA provides the corresponding public key if someone asks for a specific identity.
- **Trust Evaluation** Determining whether a certificate is valid and what operations it authorizes.

2.4.1 Secure Sockets Layer (SSL) / Transport Layer Security (TLS)

The Secure Sockets Layer Protocol (SSL) is a protocol developed by Netscape [24] designed to provide privacy between two communicating applications (a client and a server) by using public key cryptography. Second, the protocol is designed to authenticate the server and, optionally, the client. SSL requires a reliable transport protocol (e.g. TCP) for data transmission and reception.

An advantage of the SSL protocol is that it is application protocol independent. An application level protocol (e.g. HTTP, FTP, Telnet, etc.) can layer on top of the SSL protocol

transparently. The SSL protocol negotiates an encryption algorithm and a session key as well as authenticates a server before the application protocol transmits or receives its first byte of data. All application protocol data is encrypted before transmission, ensuring privacy. The connection provided by the SSL protocol has three main properties:

1. The connection is *private*. All messages are encrypted using secret key cryptography (e.g. DES, RC4, etc.) with a session key that is defined at the beginning with an initial handshake.
2. The identities can be *authenticated* using public key cryptography (e.g. RSA, DSS, etc.). The server endpoint of the conversation is always authenticated, while client endpoint authentication optionally.
3. The connection is *reliable*. The protocol includes a message integrity check using a Message Authentication Code (MAC) ensuring that package alteration between client and server is detected. The MAC is calculated using secure one-way hash functions (e.g. SHA, MD5, etc.).

Transport Layer Security (TLS) [25] is the latest enhancement of SSL. The TLS protocol is based on the SSL 3.0 protocol specification as published by Netscape. The differences between this protocol and SSL 3.0 are not dramatic, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate. The major changes are cryptographically stronger MAC computation, larger padding (up to 256 instead of 63 bytes) and some protocol cleanup (e.g. ignoring unknown record types, improved alert messages, etc.).

The OpenSSL project [26] offers an Open Source implementation of the SSL/TLS protocols. The project is based on the excellent SSLeay library developed by Eric A. Young and Tim J. Hudson and is managed by a worldwide community of volunteers.

2.4.2 Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)

Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) is used in order to provide cryptographically secure web connections. All common browsers support HTTPS, thus, no additional installation is needed on client side. HTTPS connections are often used for submitting private information (e.g. credit-card numbers, personal details, etc.).

On the server side there are SSL/TLS modules for most of the common web servers (e.g. Apache, Microsoft Internet Information Server, etc.). The architecture presented in this paper uses Apache web servers [9]. There is a project called `mod_ssl` [27] that provides strong cryptography for the Apache web server via the SSL/TLS protocols by the help of the OpenSSL toolkit described above.

2.4.3 Secure Shell (SSH)

In order to work on remote computers, Telnet [30] was formerly used. But today it is outdated due to security problems. Telnet sends the username and the password unencrypted over the network. Nowadays, protocols like SSH (Secure Shell) are used. SSH provides extended Telnet functionality over an encrypted connection. There is an Open Source project called OpenSSH [28] that provides a free implementation of the SSH protocol.

Linux distributions usually come with an SSH client, Windows unfortunately does not. But there is an SSH client called MindTerm from AppGate [29] entirely programmed in Java [33]. MindTerm can be run as a stand-alone application or as an Applet within any common web browser. MindTerm is available for free for personal, non-commercial, non-profit and non-governmental use.

2.4.4 Internet Protocol Security (IPsec)

IPsec stands for Internet Protocol Security. The IPsec protocols were developed by the Internet Engineering Task Force (IETF) [31] and will be a part of IPv6. They are also being widely implemented and used for IPv4.

IPsec uses strong cryptography in order to provide authentication, privacy and data integrity at the protocol level of the network protocol stack. These services allow to build secure tunnels through untrusted public networks. Everything passing through the untrusted network is encrypted by the IPsec gateway machine on one side and decrypted by the gateway machine on the other end of the tunnel. The result is a Virtual Private Network (VPN). IPsec uses three different protocols:

1. **AH** Authentication Header provides a packet-level authentication service.
2. **ESP** Encapsulating Security Payload provides encryption plus authentication.
3. **IKE** Internet Key Exchange negotiates connection parameters, including keys, for the other two protocols.

The here-proposed architecture uses the FreeS/WAN IPsec [32] implementation for Linux in order to secure connections between the different servers.

3 Architecture

In order to provide an online laboratory - with the possibility for remote hardware setup, configuration and exercises - that can be accessed by students from different educational institutes, it is unavoidable to build a rather complex network of different types of servers and connections. Such a Distance Learning architecture is depicted in Figure 3.1 and needs to contain at least the following elements:

- Databases where student data (such as usernames, passwords, matriculation numbers, email addresses, etc.) is stored.
- Course Servers where theory, exercises, online help and the possibility to communicate with other participants is provided.
- Portals (entry points) where users can login in order to access the laboratory hardware.
- Laboratory hardware that can be configured or used for exercises.
- Secured connections in order to guarantee privacy by encrypting network traffic that contains sensitive student data.

Every participating educational institute maintains its own student database, sets up its own Portal Server, possibly a Course Server and provides different hardware to solve the tasks [3]. The here-proposed architecture is designed to stimulate collaboration between the different educational institutes. Nowadays, each institute already maintains a student database that is very often stored on an LDAP server. For those who have not yet used LDAP, it is easy to export their data in LDIF files.

Student data is read very often, but it is hardly changed. As LDAP is optimized for read access, LDAP seems to be a good choice. Another good reason for using LDAP is that LDAP is a widely accepted standard which is still being developed. Furthermore, there is a product called OpenLDAP [17] which is free and Open Source.

Based on LDAP a rather powerful student management system with minimal administrative overhead can be outlined. Student data stored in an LDAP server cannot only be used for VITELS related tasks but allows also an LDAP-based implementation of a university-internal online phone book, LDAP-based web authentication, LDAP-based authentication for the student's email accounts and so on.

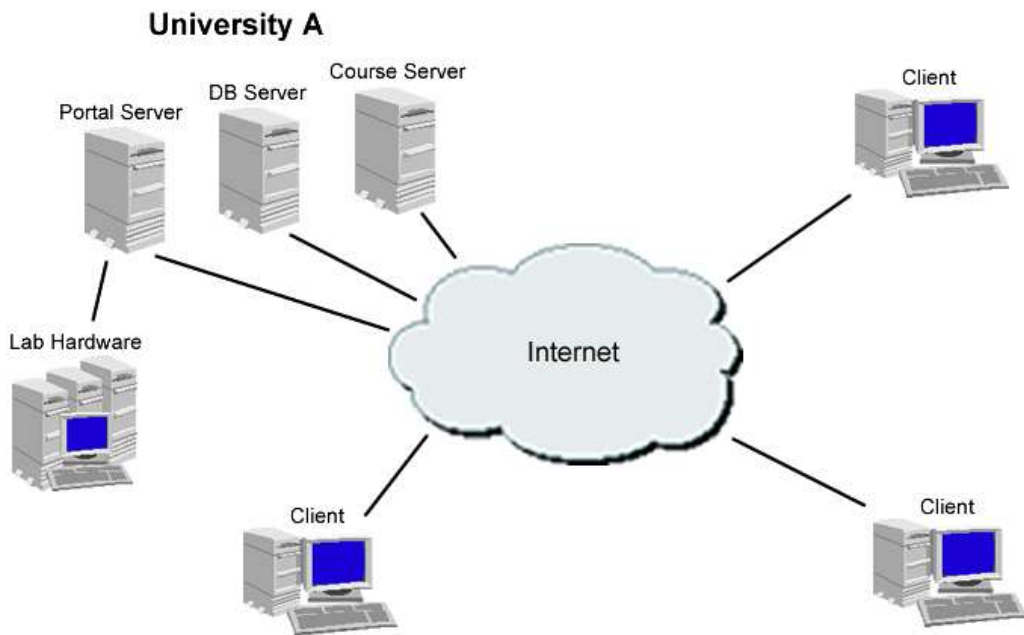


Figure 3.1: General Architecture

3.1 Security Architecture: Kerberos versus Public Key Infrastructure

First of all, the most appropriate type of security architecture had to be evaluated. There are lots of proposed security architectures available on the Internet. Most of them are not yet available in a stable release or do not cover the needs of the VITELS project. It was important to find a solution that could be implemented immediately. Furthermore, the VITELS project consists of several educational institutes and, thus, requires an inter-organizational AAA architecture. The two architectures presented here seemed to be the most elaborated security architectures. On one side there is Kerberos, on the other side there is Public Key Infrastructure (PKI). Both have advantages but there are also significant disadvantages on both sides.

3.1.1 Advantages and Disadvantages of Kerberos

The advantages of Kerberos are the followings:

- Kerberos provides *authentication*, *confidentiality* and *data integrity*.
- Passwords that are not secure enough are rejected.
- Kerberos is designed to be user-friendly, the user has to login with his password only once.
- The password is encrypted before it is transmitted, this prevents network sniffing attacks.

- Finally, Kerberos is Open Source and freely available.

Unfortunately, there are also disadvantages:

- Kerberos needs an especially secured environment with an online database where all the secret keys are stored. This is critical because if the system containing all the secret keys is vulnerable, the entire architecture is corrupted. The centralized aspect could be seen as an advantage, too, because it takes the responsibility of securely storing the key from the user.
- Another, major drawback is that Kerberos relies on kerberized clients. This means that every student needs to have special software installed on his system in order to access the VITELS online courses. This cannot be accepted because students should not be required to install additional software or forced to use a special operating system. And this implies that access should be granted from every available computer, whether it is a MAC, a PC or a Unix system, whether it is at home, at the university or in an Internet cafe.
- The problem described above could have been reduced, if there had been existing Kerberos patches for Apache [22, 23] and plug-ins for Netscape Communicator [21] and Internet Explorer. Some of them existed, but either they are no longer maintained, or have problems (e.g. with Microsoft's Internet Explorer) and have not released new versions for more than a year now.
- Another disadvantage is the use of *secret key (traditional) cryptography*, which - if based on user passwords - cannot be as secure as *public key cryptography*. This favors *password guessing attacks*.
- There are problems with inter-realm authentication (A user from a realm A, e.g. University of Bern, that intends to access another realm B, e.g. University of Geneva).
- Last but not least, Kerberos provides authentication, but *no authorization*.

3.1.2 Advantages and Disadvantages of a PKI

PKIs have lots of advantages as listed below:

- a PKI provides *authentication, confidentiality and data integrity*.
- In contrast to Kerberos, it offers *public key cryptography*. Therefore, the keys are stored on the user's computer, which implies that there is no need for an online accessible centralized key database.
- Because of the use of public key cryptography, a PKI provides *secure key distribution*.
- Another consequence is the *non-repudiation*. This means that a user that has used his private key to commit some action cannot deny this later on, because he is assumed to be the only one that has access to his private key.

- Yet another important advantage is that a PKI does not require an online accessible Trusted Third Party (TTP) that has to be protected against attacks (such as the KDC in Kerberos).
- There are lots of protocols like Secure Sockets Layer (SSL), Transport Layer Security (TLS), Secure Shell (SSH), IPsec, etc.
- And finally, PKIs are LDAP-capable which is very welcome in the VITELS project.

The disadvantages of a PKI are:

- PKI is still developing, there are some aspects that are not entirely resolved, as for example *suspended certificates*. That means certificates that are not revoked but, nevertheless, not valid at the moment, they are suspended for later use.
- Another problem might be the online availability of an up-to-date black-list of *revoked certificates*. Certificates can be revoked when assuming that the corresponding private key has been compromised. Revoked certificates should not be used anymore and, thus, the black-list needs to be accessed every time a certificate is used.
- The advantage described above - no centralized key database - contains a negative part, too. The user alone is responsible for the protection of his private key. At least, a few malicious users can not corrupt the whole system, just their own accounts.
- The private key is secured by a user-defined passphrase. The user cannot yet be forced to use a cryptographically strong passphrase.
- And last but not least, one has to ensure the distribution/protection of the real root certificate. This certificate is the ultimately trusted certificate which has to be accepted by every user, thus, it must be published over and over again in different medias in order to prevent man-in-the-middle-attacks.

3.2 Proposed Security Architecture

For the use within the VITELS project a PKI seems to be more appropriate than Kerberos. Students ought to be able to access the architecture without any installation and configuration effort on their part. Every common browser supports HTTPS which uses SSL/TLS for secure communication. SSH connections can be established through SSH Java Applets which are also supported by most common browsers. Thus, when using a PKI no additional installation is required on the student side. As a consequence, in order to make it as convenient as possible for the students a PKI architecture was chosen. There are existing protocols as described above that fit the needs for the VITELS project. Figure 3.2 shows a possible architecture using these security protocols.

All connections to the LDAP and CA servers are secured with IPsec. The LDAP servers reply to queries coming over the IPsec network interfaces and do not accept incoming requests on other network interfaces. The IPsec tunnels are statically set, which allows every university to easily

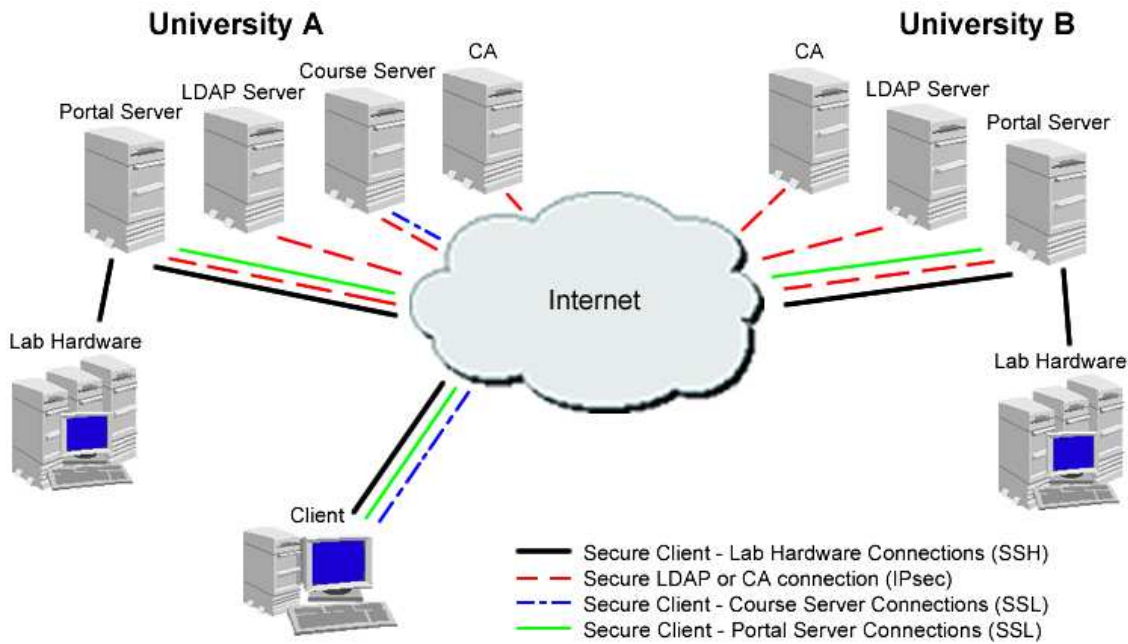


Figure 3.2: Security Architecture

control the database queries to their servers. Such queries can only be made from connected (with IPsec) servers such as the Portal Servers or the Course Servers. This solution does not limit the advantages of an accessible student database (like for example an online phone book) because such applications only need to run on securely connected web servers. But it prevents abuse, the LDAP server cannot be contacted from unwanted or unsecured servers or hosts on the Internet.

The widespread protocols like SSL and TLS allow students and module administrators to access the course and configuration websites, the online exercises and discussion forums over an encrypted connection without being afraid of having an attacker sniffing the connection for private information (such as usernames, passwords, etc).

Where a direct login to laboratory hardware is necessary, the connection - and therewith private information such as usernames and passwords, etc. - can be secured making use of the SSH protocol. The MindTerm SSH client can be started simply by clicking on a link on a website. The Java Applet is then being launched and the secure connection can be established.

3.3 Directory Structure

In order to store the complex data used by the scheduling script a clear and simple directory structure needs to be defined. It is also desired that the LDAP servers can later be easily integrated into the architecture developed by SWITCH [7]. Thus, the VITELS directory structure should

conform to the directory structure developed by SWITCH.

As LDAP is a hierarchical database system, there must be a top level entry, a so-called root entry. Usually, there are two approaches for an appropriate hierarchy: the *domain-based* and the *organization-based* approach. The domain-based approach builds the directory structure based on the Internet domain name of the company. Applied to the University of Bern this would result in the structure shown in Figure 3.3 because the Internet domain name is `www.unibe.ch`.

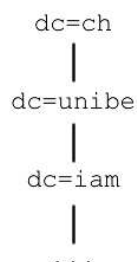


Figure 3.3: Domain-based Structure

The second approach represents a more natural structure. It starts with a country, followed by organizations that are split up in organizational units (`ou`). In our example (see Figure 3.4) an `ou` could be the Institute of Computer Science and Applied Mathematics (IAM).

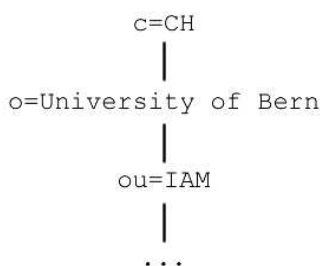


Figure 3.4: Organization-based Structure

For the VITELS architecture the second approach has been chosen because it fits the directions formed by SWITCH. The second approach makes an eventual integration into the tree maintained by SWITCH easier. SWITCH also uses the second approach for its directory and as LDAP is still being developed the assumption can be made that in the future all universities are linked into the directory maintained by SWITCH.

Thus, each university needs to have an organizational subtree with its student data stored inside. This tree is not strictly defined, at the University of Bern it was chosen to be as shown in Figure 3.5. An additional subtree containing VITELS specific data needed to be defined. There are basically three important groups of information that need to be stored in this tree: the timetable, module data and contact and login information for the VITELS staff members (Figure 3.6).

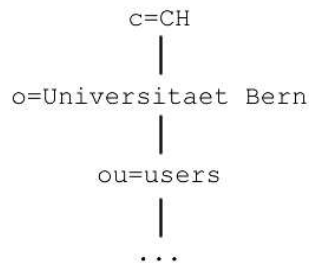


Figure 3.5: Directory Structure for Universities

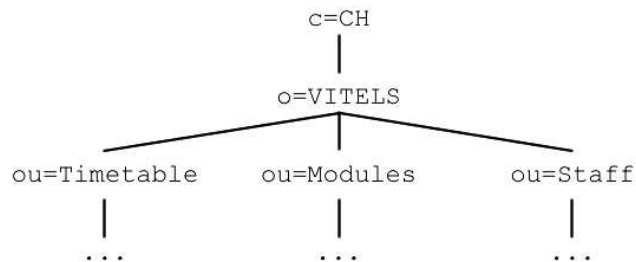


Figure 3.6: Directory Structure for VITELS-specific Entries

3.3.1 Student Entries

The student entries are not stored in the VITELS subtree but within the university subtree and should at least contain the information shown in Figure 3.7.

```

dn: uid=tj97r827,ou=users,o=Universitaet Bern,c=CH
objectclass: person
objectclass: account
uid: tj97r827
userpassword: {crypt}21.K15z1i8JlL
cn: Thomas Jampen
sn: Jampen
  
```

Figure 3.7: A Student Entry

The attribute `userpassword` is usually encrypted with the standard Unix `crypt` function before it is stored. The prefix `{crypt}` indicates that the value is not the password itself but contains an encrypted password. When comparing passwords, the LDAP server first encrypts the password entered by the user and compares the two in encrypted form, because the `crypt`-algorithm is designed not to allow the decryption of encrypted data within a reasonable amount of time.

3.3.2 Timetable

The timetable used for authorizing students to access a certain module is stored within its own directory subtree `ou=Timetable`. There is a separate timetable for each module. As a consequence, the next level of the tree contains entries for each module (Figure 3.8 shows an example for module 1). The entries are just used in order to split the timetable into different sections. This makes sense because each module maintains its own timetable. There is an attribute `mid` that uniquely identifies each module. This attribute is forced by the object class `module`.

```
dn: mid=1,ou=Timetable,o=VITELS,c=CH
objectclass: module
mid: 1
```

Figure 3.8: A Timetable Entry

On the next level, there are the entries for the actual timetable (Figure 3.9).

```
dn: slot=20020415 0800,mid=1,ou=Timetable,o=VITELS,c=CH
objectclass: timetable
objectclass: alias
slot: 20020415 0800
aliasedobjectname: uid=tj97r827,ou=users,o=Universitaet Bern,c=CH
```

Figure 3.9: A Timeslot Entry

The attribute `slot` uniquely defines a timeslot within one module. This attribute is claimed by the object class `timetable` (see Section 5.1.1). The first eight digits specified by `slot` represent a date (the format is `YYYYMMDD`), the last four digits indicate the slots starting time (`HHMM`). The attribute `aliasedobjectname` points to the user that has reserved this very slot.

3.3.3 Modules

The directory subtree `ou=Modules` is built in order to provide a simple interface for other applications that intend to access a module's current user and general module settings (such as the number of slots per day, the slot length and the starting time of a day's first slot). Every VITELS module needs to know its current user and, in order to prepare a new user's session, it needs to access the module settings. A sample module entry is shown in Figure 3.10.

This entry defines a module that allows five slots per day lasting three hours each. The first slot starts at eight o'clock in the morning. The analogue to the user identification `uid` is the module identification `mid` that has to be unique. The `mid` is mandatory because of the use of the object class `module` (see Section 5.1.1). The object class `alias` - and the belonging attribute `aliasedobjectname` - is used in order to provide a link to the user that is currently - at the time of the query - allowed to access the module.

```
dn: mid=1,ou=Modules,o=VITELS,c=CH
objectclass: module
objectClass: alias
mid: 1
firstslot: 0800
slotlen: 3
numslots: 5
aliasedobjectname: uid=tj97r827,ou=users,o=Universitaet Bern,c=CH
```

Figure 3.10: A Module Entry

3.3.4 Staff

The entries stored within the directory subtree `ou=Staff` have got the same structure as the entries stored within the universities' student data subtree explained above (see Section 3.3.1). They contain at least the object classes `person` and `account` or, alternatively, the object class `alias` when the entry is linked to an entry stored within a different subtree.

This subtree contains user entries or aliases to user entries for the VITELS staff. On one side, it is used to authenticate VITELS staff members and, on the other side, it serves as an online address book. This address book can be viewed and changed on the VITELS staff website [6] or it can be imported for example into the address book provided by Netscape's Communicator. In order to make this possible, the entries include another object class called `inetOrgPerson`. Its attributes are recognized by Netscape and allow to correctly display the directory entries in its address book. The attributes specify values that are used in the context of the Internet, such as email, homepage address, etc.

4 Implementation

4.1 Scheduling Script

As a programming language for the scheduling script and the online timetable PHP (PHP Hypertext Preprocessor) [34] has been chosen because it allows to dynamically create web pages on the server and offers a comfortable LDAP API (Application Programming Interface). PHP is a widespread Open Source scripting language that is especially suited for web programming. PHP can be embedded into HTML (Hypertext Markup Language) code, it can be used to create usual CGI (Common Gateway Interface) scripts, but it can do even more: With PHP images, PDF (Portable Document Format) documents and Flash animations can be dynamically generated. PHP supports XML (Extensible Markup Language) and almost every database system. PHP can be used to directly send and receive mails and allows even to instantiate Java objects.

4.1.1 LDAP Queries

All LDAP-based operations and the main scheduling functions have been realized within the PHP class `LDAPQueries`. The class is stored in a file called `ldapqueries.inc.php`. The name of the file contains the extension `.inc` in order to make clear that this file has to be included by other files. It contains functionality that cannot be run on its own and - when directly called in a browser - it does not display anything. Thus, it has to be included in another `.php` file. The extension `.php` is given for security reasons, because files with the extension `.inc` are displayed in plaintext within the browser, and this could be a security problem. As a consequence, the extension `.php` has been added. As we will see later on in Section 4.2.2, the entire configuration is done within a separate file.

The class `LDAPQueries` has to include this configuration file in order to access the configuration variables. There is another file `ldaplock.inc.php` that has to be included in order to be able to use a directory locking mechanism when reserving slots (see Section 4.1.2).

There are just a few global variables in the class `LDAPQueries`. `$conn` saves a link identifier for the current LDAP connection and will be used by most of the functions within this class but should not be accessed from outside. The variable `$timetable` saves the reservations extracted from the LDAP tree. It is a three dimensional array that is used later (within the file `timetable.inc.php`) in order to display the online timetable. The variables `$num_slots`, `$first_slot` and `$slot_len` save the current module's specific information. This information covers the amount of slots per day, the starting time of the first slot and the duration

of each slot of the current module. `$module` contains the current module's id. Furthermore, there is the variable `$num_modules` containing the number of modules currently available, and `$module_ids` is an array that stores all the identifications for these modules. All these variables are made global for convenience in order to reduce the overhead of passing lots of variables back and forth.

The class `LDAPQueries` offers functions that connect and bind to a specified LDAP server: `connect()` and `bind($dn, $pwd)`. This class does not allow anonymous binding, that means, in order to make successful queries you need to specify an existing distinguished name and the corresponding password. Another function, `cleanup()`, allows to properly close the connection to the LDAP server.

The function `get_all_modules()` retrieves all currently available modules from the LDAP tree and builds the global array `module_ids` and assigns the variable `num_modules`. This is done in order to be able to display a list of modules the student can choose from.

There are two functions for accessing and changing module specific settings. The first one - `set_module_settings($user_data, $mid, $first, $slots, $len)` - allows a user whose dn and password are saved in the array `$user_data` to change the specified module's values for the first slot, the number of slots per day and the slot length. The second function - `get_module_settings($user_data, $mid)` - reads these settings from the directory tree. The user data is again used in order to authenticate the user to the LDAP server and the module id determines which module's settings are to be retrieved. This function in fact just binds the user to the directory and then calls the function `read_module_settings($mid)` in order to really read the data from the directory. This functionality has been split into two functions because sometimes the connection to the server has already been established when module settings need to be read.

After connecting and binding to the LDAP server, `build_timetable($user_data, $mid, $date, $show_attr)` retrieves all available modules and the settings of the specified module with the help of the above described functions. This is the module the timetable is built for. Thus, it has to be known how many slots there are per day, how long these slots are and when the first slot begins. After that, a three dimensional array `$timetable` for this module is built. This timetable represents exactly one week, beginning with the Monday specified by `$date`. The first dimension of the array specifies the slot time, the second dimension determines the weekday and the third dimension is used in order to store the attribute given by `$show_attr` (usually the common name) of the student and another shorter value (e.g. the user id) that can be displayed within the timetable. This function returns true, if the timetable could have been built or false, if there are slots missing in the LDAP directory.

`reservation_allowed($user_data, $mid, $slot)` just checks if the specified user is allowed to bind to the server. This has to be verified because the actual reservation is done using an administrator account because ordinary students are not allowed to make changes in the directory. This function can or should be extended if reservation limitations are desired (see Section 8.2.1). This function is called by `make_reservation($user_data, $mid, $slot)` which actually performs the reservation. In order to make sure that no one else is reserving the same slot at the same time, a locking mechanism has to be used (see Section 4.1.2). After successfully locking the directory, it has to be verified that the slot is still

unreserved because the locking procedure may take some time. Slots can be freed with the function `undo_reservation($user_data, $mid, $slot)`. This function first verifies that the slot has really been reserved by the user that is currently trying to free it using the function `is_reserved_by($user_data, $mid, $slot)`. If that is the case, slot is freed using the already mentioned administrator account, if not, the administrator account is not used, but it is tried to free the slot using the user's own account. This procedure will succeed only, if the user is the module administrator and, therefore, has the right to free other user's slots. Unfortunately, this cannot be checked by the program because module administrators are hard-coded in the LDAP configuration file (see Section 5.1.2).

A module administrator is allowed to create and remove timeslot entries belonging to the module(s) he administrates. By calling `create_slots($user_data, $change_mid, $from, $to, $expiry)` slots can be added to the directory. It is possible, but not mandatory, to specify an expiration date for these slots. That means, that these slots are removed automatically when they expire. The function `remove_slots($user_data, $change_mid, $from, $to)` is used when an administrator removes timeslots manually. Before creating or removing any timeslots, the function `verify_date($date)` verifies the starting and ending date. This function can be extended or adapted to special needs, at the moment it just verifies that the date is not more than one year in the past or future. After that, the desired timeslots are added - if they do not already exist - or removed, if they in fact existed, (this is verified using `slot_exists($dn)`). If the modification succeeds, the amount of added or removed slots is returned in order to be displayed on the website, if not, it is because the user is not a module administrator.

The function `is_staff($user_data)` allows to check whether a user is a VITELS staff member. This is used in order to display the administration menu for staff members and module administrators. As a consequence, each module administrator has to be a staff member. This makes sense since module administrators are responsible for their module and, thus, probably need to be contacted.

Last but not least, the function `display_page($page)` is used in order to display an error page when the LDAP server cannot be contacted, or the login page, if the user cannot be bound to the server due to an invalid username or password.

4.1.2 LDAP Locking Mechanism

Given that several students are able to try to reserve the same timeslot simultaneously, the LDAP directory has to be locked in order to prevent that one student overwrites the reservation of another student. Due to the fact that locking is not supported by LDAP, such a locking mechanism had to be implemented.

PHP provides a mechanism to lock files, that means one can grab the lock for exclusively accessing a file. Unless the exclusive lock is released, no one else can access that file. For the VITELS project this means that a lockfile has to exist permanently. Before a reservation is made, the program tries to grab the lock for the special file. If the lock could be acquired the reservation is accomplished, if not, the timetable on the screen is updated and will probably show that the slot is reserved by now. After a successful reservation the lock is released.

The locking procedure is encapsulated within the class `LDAPLock`. This class just provides the functions `lock()` and `release()`. The lock function grabs the lock if possible and times out if the lock could not be acquired for a certain time. This is seen as a comfort to the students because it is unpleasant to sit in front of a computer seeing the sandglass and not knowing how long it will take to finish the reservation.

4.1.3 LDAP Configuration

The file `ldapconfig.inc.php` contains the necessary configuration options for the scheduling system. Every parameter and its meaning is described below:

\$server This specifies the URL of the VITELS' main LDAP server. This is meant to be the server containing the VITELS subtree.

\$port The port used to connect to the server.

\$sizelim A size limit for entries to be returned when searching the directory. 0 is unlimited.

\$timelim A time limit after which a search is aborted. 0 means unlimited.

\$unibe_base The base dn for students at the University of Bern.

\$unifr_base The base dn for students at the University of Fribourg.

\$unige_base The base dn for students at the University of Geneva.

\$unine_base The base dn for students at the University of Neuchatel.

\$ingfr_base The base dn for students at the Engineering School Fribourg.

\$tt_base The base dn for the timetable subtree.

\$mod_base The base dn for the modules subtree.

\$staff_base The base dn for VITELS staff entries.

\$dummy_uid The user id of a dummy user. This user is needed in order to keep unreserved slots available. If an LDAP entry contains attributes with empty values, they are removed. This is not desired, thus, a dummy user has been created.

\$dummy_dn The dn of the dummy user.

\$dummy_data An array containing the dummy user's data.

\$login_page The name of the login page to be displayed when a user enters invalid login information.

\$error_page The name of the error page to be displayed when the LDAP server is not reachable over the Internet.

\$dt_format The format of the date and time of a timeslot. Currently it is `YYYYMMDD HHMM`.

\$bind_dn The dn of an administrator used when reserving and freeing timeslots.

\$bind_pwd The password corresponding to the dn of the administrator used when reserving and freeing timeslots.

\$mail_admin The email address of the LDAP server administrator.

\$lockfile The name of the lockfile. This file needs to have permissions 0644 or 0444, which means, it has to be readable by “other”, e.g. the webserver user.

\$lockdelay The delay in seconds to wait when grabbing the lock for reserving a timeslot.

4.2 User Interface to the Scheduling Script

4.2.1 Slot Reservation and Module Administration

The graphical user interface (GUI) consists of several `.php` files. The main files are: `gui.php`, `login.php` and `ldaperror.php`. All of them include the file `html.inc.php` that contains some basic HTML functions. It provides three functions, the first one for printing a HTML header `print_html_header($title)` (including basic cascading style sheet (CSS) information for a common look and feel of the websites), the second one for printing an HTML footer `print_html_footer()` that correctly terminates HTML documents, and the third one for printing hidden fields (`print_hidden_fields($user_data, $mid, $date, $staff_view, $ldap)`) that are used to store the internal state of the website. This means that the hidden fields contain information about what action the user has performed the last time he submitted a form by clicking on a link.

The file `login.php` displays a login page where students and administrators can enter their login information such as educational institute, username and the corresponding password. The educational institute is very important in order to determine which LDAP server to contact or which directory tree to use for authenticating the user. The data entered in this form is submitted to the file `gui.php` together with a date. This date represents the current week’s Monday that is used in order to know which week to display in the timetable.

The `ldaperror.php` file is displayed when an LDAP server cannot be contacted. It displays the information that the LDAP server might be down and shows an email address where students can inform an administrator.

The main file is `gui.php`. It evaluates every parameter passed by the submitted forms. Based on these parameters it decides, what action to perform and, thus, which part of the GUI to display. This file includes the two files `timetable.inc.php` and `admin.inc.php` that contain the HTML code for the timetable and for the administration menus. Furthermore, it

includes `ldapqueries.inc.php`. With the help of these included files, `gui.php` is able to perform the appropriate LDAP queries in order to display the desired menu or data.

The file `timetable.inc.php` only contains one function that builds and displays the timetable based on the given parameters (`display_timetable($user_data, $ldapq, $staff_view, $query, $done)`). `$ldapq` is an object of the class `LDAPQueries`. The parameter `staff_view` indicates whether symbols - defining reserved and free slots - or the real names are displayed within the timetable. Of course, this option is limited to module administrators. `$query` indicates whether the timetable could have been built or not, and `$done` contains additional information for module administrators. For ordinary students, just the timetable is built and displayed but if a module administrator is logged in, the function `display_admin_menu($isstaff, $staff_view, $done)` defined in the file `admin.inc.php` is called in order to display the administration menu where all possible administrative task can be chosen. These tasks cover displaying the usernames instead of the symbols in order to see who reserved which slot, adding and removing timeslots for certain modules and changing a module's settings.

Last but not least, the file `admin.inc.php` contains three function of which one has already been described above. The two remaining functions display forms to add and remove slots `display_ar_slots($user_data, $mid, $date, $staff_view, $done)` and to change a module's settings (`display_change_mod($user_data, $mid, $date, $staff_view, $ldapq, $done)`).

4.2.2 User Interface Configuration

The configuration file `guiconfig.inc.php` influences the behavior and the look and feel of the websites. Every parameter and its meaning is described below:

show_attr The attribute to be displayed in the timetable for administrators if they decide to view the names instead of the symbols.

weekdays An array of abbreviations for the weekdays to be displayed in the timetable.

num_days How many days to display in the timetable. Usually the length of `weekdays`.

no_slots A sentence to display when no - or not enough - slots are available for the currently selected module and week.

title_pre A title prefix to be displayed in the title bar of the browser for every VITELS page.

4.3 Cron Jobs

A cron job specifies a program and a point of time. At that time the given program is automatically run by the cron daemon. For the VITELS project two cron jobs are needed. These cron jobs have been implemented in Perl [36]. The implementations use the Perl module `Net::LDAP` in order to access the LDAP directory.

4.3.1 Updating Each Module's Current User

In order to make it as simple as possible for other educational institutes - that are implementing their own applications which need to verify the current user of their module - the branch `ou=Modules,o=VITELS,c=CH` has been inserted into the directory tree. Every module has an entry, defining an alias to the current user at the time of the request. These entries have to be always up-to-date and, thus, a program is run every minute on the server containing the VITELS directory tree in order to check whether the current user has changed. Like that, potential changes are straight away detected and the corresponding entry is adapted.

The steps performed by the script `current_user.pl` include contacting the LDAP server, binding as an administrator and getting all available module ids. Based on these ids, each module's settings (such as a day's first slot, the slot length and the number of slots) are extracted. Based on these values, the starting time of the current slot is calculated using the function `get_current_slot_time($firstslot, $slotlen, $numslots)`. If there is an available slot at the moment, its effective current user is queried from the LDAP server. Then, the user marked as current user in the directory tree `ou=Modules,o=VITELS,c=CH` is queried using the function `lookup_aliasedObjectName($ldap, $dn)` and is then compared to the effective current user. If necessary, the corresponding entry is updated. At the end, the connection the LDAP server is closed.

This Perl script is supposed to be run every minute by the cron daemon. In order to install the `current_user.pl` script as a cron job, a so-called `crontab` file needs to be created or edited. Assuming the script is placed in the directory `/usr/local/bin` the appropriate `crontab` looks as follows:

```
_____ file: crontab _____  
SHELL=/bin/sh  
# crontab file for the VITELS project  
#  
# check the current user every minute  
# m h dom mon dow command  
* * * * * /usr/local/bin/current_user.pl
```

In order to be run as a cron job, the file `current_user.pl` needs to be executable. The entries of the `crontab` file specify when the script is run. The first entry specifies the minute (* means every minute), the second entry determines the hour. After that, the day of month (e.g. 13 for the thirteenth), the month itself (e.g. 4 for April) and the day of week (e.g. sun for Sunday) can be specified. The last entry contains the path to the script and the name of the script.

4.3.2 Removing Expired Timeslots

Module administrators are able to add new timeslots to the LDAP directory tree. Generally, timeslots that are past should be removed in order to prevent the directory from growing ad infinitum. For the course statistics and for the behavior of the online timetable it is convenient to keep the timeslots for a certain time after they are past.

When adding new timeslots, administrators are recommended to specify an expiration date. A daily cron job checks every existing slot and removes expired slots from the directory tree. If an administrator omits the expiration date for certain slots, these timeslots will not be removed automatically and will have to be deleted by the administrator himself using the *add/remove timeslots* webpage shown in Section 5.2.1.

After connecting to the appropriate LDAP server and binding as an administrator, all available modules are retrieved. Based on the result, each module's timetable is queried in order to get all available slots. Each slot's *expires* entry has to be accessed in order to determine whether or not to remove the slot from the directory. After removing the expired slots, the connection to the LDAP server is closed and the script terminates.

This script should be run once a day, usually during the night because the system activity is lower as hardly any users are connected. The *crontab* file for this cron job looks as follows. It is run every night at 3:42 am.

```
_____ file: crontab _____  
SHELL=/bin/sh  
# crontab file for the VITELS project  
#  
# daily cleanup of expired timeslots  
# m h dom mon dow command  
42 3 * * * /usr/local/bin/ldap_cleanup.pl
```

4.4 VITELS Staff Directory

As mentioned before, the VITELS website contains a section that lists all staff members and their personal details. These HTML pages are generated by a single PHP file called *staff.php*. It is divided by *if-else*-statements into six parts. The first part is executed when no specific action has to be performed. It queries the *ou=Staff, o=VITELS, c=CH* subtree of the LDAP server and gets all available staff members. The entries are sorted and HTML code is generated in order to display a list of all staff members' names and their educational institute. The second part is executed when a single entry of the staff list has been chosen by clicking on a staff member. The LDAP server is contacted in order to get the selected staff member's personal contact information. Based on this information, a table is built and displayed to the user. The third section gets executed when a staff member intends to change his personal information. The same table as described above is built and displayed but the attribute values are shown within a text field in order to allow modification. An additional text field where the staff member enters his password is appended in order to prevent other users from changing his details. If the staff member submits the changes, the fourth section is called. The changes are sent to the LDAP server and the user is displayed whether the changes could have been performed or not, a failure results due to an invalid password. The fifth section is called when a staff member intends to change its password. A password can only be changed, when the old password is valid and when the new password has been entered two times without any difference. After submitting the new password, it is written to the LDAP directory in the sixth part. In case of errors, appropriate information is displayed on the website.

4.5 PHP-based API for Accessing a Module's Current User

In order to provide an easy to use API to other VITELS module developers, the PHP class `VitelsLDAP` has been implemented. This API provides the functionality to access the subtree `ou=Modules,o=VITELS,c=CH` of the LDAP directory. That is where a module's current user is saved.

This API contains the functions `connect()`, `bind($dn, $pwd)`, and `cleanup()` - that have already been described in Section 4.1.1 - in order to build or terminate connections to the LDAP server. Furthermore, the functions `read_module_settings($mid)` and `display_page($page)` have already been discussed, too. There are new functions, such as `get_current_user($mid)`, which retrieves the current user for the specified module and returns his distinguished name. `get_current_slot_info($mid)` reads the specified module's settings and uses them in order to calculate the starting and ending time of the current slot. The resulting times are stored in standard date format (seconds since January, 1st 1970) within an array that is returned.

The most comfortable functions are `authenticate_user($dn, $pwd)`, which tries to bind with the specified distinguished name and password and returns, whether that could be done successfully, and `authorize_user_for_module($mid)`, which must be called after a successful authentication. It compares the current user with the authenticated user and returns the result. Finally, there is one last function that does all the above in one function in order to make it as simple as possible for developers that only need to get the starting and ending time of the current slot under the condition that the specified user is, in fact, the current user and the provided password matches. This function is called `get_slot_if_current_user($dn, $pwd, $mid)`.

5 Configuration and Usage

5.1 LDAP Server Configuration

As described in Section 2.1, schema files are an important part of the LDAP server configuration. The other major part is the configuration file `slapd.conf` for the stand-alone LDAP daemon (slapd). These files are explained in detail in the following Sections.

5.1.1 VITELS Schema File

Within the VITELS directory subtree, module and timeslot information needs to be stored. This information does not use a structure already used by other common directory trees. This means that there are no appropriate object classes and attribute types for the VITELS specific entries. Thus, a new schema file had to be created.

The following file first defines all necessary attribute types and specifies later in the context of which object classes they can or must be used.

```
file: vitels.schema

# VITELS schema file
#
# Depends on:
#   core.schema
#   cosine.schema
#
# author:    Thomas Jampen <jampen@iam.unibe.ch>
# created:   20011119

attributetype ( 1.3.6.1.4.1.9999.2.1.1 NAME 'mid'
    DESC 'A string uniquely defining a module'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{10}
    SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.9999.2.1.2 NAME 'slot'
    DESC 'A string of the form YYYYMMDD HHMM defining the beginning of a time slot'
    EQUALITY caseIgnoreMatch
    SUBSTR caseIgnoreSubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{13}
    SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.9999.2.1.3 NAME 'numslots'
    DESC 'An integer defining the number of timeslots/day'
    EQUALITY numericStringMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27{2}
```

```

SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.9999.2.1.4 NAME 'firstslot'
  DESC 'A string defining the starting time of the first slot'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{4}
  SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.9999.2.1.5 NAME 'slotlen'
  DESC 'A string defining the length of the slots in hours'
  EQUALITY numericStringMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27{2}
  SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.9999.2.1.6 NAME 'expires'
  DESC 'A string of the form YYYYMMDD defining when a slot expires'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{8}
  SINGLE-VALUE )

objectclass ( 1.3.6.1.4.1.9999.2.2.1 NAME 'module' SUP top STRUCTURAL
  DESC 'VITELS Module Object'
  MUST ( mid )
  MAY ( objectClass $ description $ numslots $ firstslot $ slotlen ) )

objectclass ( 1.3.6.1.4.1.9999.2.2.2 NAME 'timetable' SUP top STRUCTURAL
  DESC 'VITELS Timetable Object'
  MUST ( slot )
  MAY ( objectClass $ expires $ description ) )

```

The first attribute is a unique module identification `mid`. Currently, this is only a number but it is defined to hold any - at most ten character - string identifying a single module. One module cannot have more than one module identification. The attribute `slot` is a string consisting of exactly thirteen digits (the format is `YYYYMMDD HHMM`). To be more clear, this attribute specifies the beginning of a timeslot. It consists of a date and - after a space character - the starting time in hours and minutes. The next four attribute types specify attributes defining the duration a single slot, the number of slots per day, the starting time of a day's first slot and the slot expiry date.

Below these specifications, two object classes define within which context the above attributes can or must be used. The first object class has the name `module`. It defines the behavior of the VITELS modules. Each module needs to have a unique module id (`mid`) and is allowed to have additional information like a description, the length of the timeslots for this module, a number of slots per day and the time of a day's first slot. A module entry is also allowed to contain additional object classes in order to provide extensibility. The second object class specifies a VITELS timetable object which is actually a timeslot. A timetable entry must provide a value for the `slot` attribute and is allowed to give additional information for the attributes `description` and `expires`. The attribute `expires` can be used in order to prevent the directory from growing constantly as expired slots are removed automatically from the directory by a cron job. A timetable object is also allowed to specify additional object classes like for example the object class `alias` with the mandatory attribute `aliasedobjectname`. This allows to specify an alias (pointer) to the student that has reserved the timeslot.

5.1.2 Slapd Configuration File

The configuration file for the stand-alone LDAP daemon (slapd) is not presented as a whole because a part of the configuration file is system specific and has no influence on the behavior for the VITELS project (e.g. location of the database). Nevertheless, there are two important parts within the configuration file `slapd.conf` that are mentioned here: schema files and access control list (ACL).

Schema Files

It is possible to include standard, as well as user-defined, schema files. As the VITELS directory structure is well-defined, it makes sense to turn on schema checking. This means that changes to the LDAP directory tree are only accepted if they conform to the rules specified by the included schema files. Schema checking might be turned off when using the server for a project where users can add entries and attributes not yet defined when starting the LDAP daemon, but this is not the case for VITELS project. Schema files are included in `slapd.conf` as listed below.

```
_____ file: slapd.conf - Schema Files _____  
  
# Schema and objectClass definitions  
include      /etc/ldap/schema/core.schema  
include      /etc/ldap/schema/cosine.schema  
include      /etc/ldap/schema/inetorgperson.schema  
include      /etc/ldap/schema/vitels.schema  
  
# Schema checking  
schemacheck  on
```

Access Control List

The `slapd.conf` file allows you to specify an access control list (ACL), where access to certain parts of the directory tree can be limited or granted.

```
_____ file: slapd.conf - ACL: Students/Staff _____  
  
# The userPassword by default can be changed  
# by the entry owning it if they are authenticated.  
# Others should not be able to see it, except the  
# admin entry below  
access to attribute=userPassword  
    by dn="uid=admin,ou=Staff,o=VITELS,c=CH" write  
    by anonymous auth  
    by self write  
    by * none  
  
access to dn="uid=.*,ou=Staff,o=VITELS,c=CH"  
    by dn="uid=admin,ou=Staff,o=VITELS,c=CH" write  
    by self write  
    by * read  
  
access to dn="uid=.*,ou=users,o=Universitaet Bern,c=CH"  
    by dn="uid=admin,ou=Staff,o=VITELS,c=CH" write  
    by self write  
    by * read
```

The first entry specifies that the attribute `userPassword` can be changed by the administrator listed and by the owner of the entry (`self`). It can be used to authenticate a user, but all users - except the admin and the owner - have got no access and, thus, cannot read the password. The next two entries define the access restrictions to the staff member and the student entries. Again, the administrator and the owner have got the right to write changes, all other users are allowed to read the entry. It is essential that the above access restriction to the attribute `userPassword` is mentioned before this one because otherwise, the password is readable by everyone.

file: **slapd.conf** - ACL: Administrators

```
# Restrict module write access to the global administrator
# and the module administrator
access to dn="mid=6,ou=Modules,o=VITELS,c=CH"
    by dn="uid=admin,ou=Staff,o=VITELS,c=CH" write
    by dn="uid=jampen,ou=Staff,o=VITELS,c=CH" write
    by * read

access to dn="mid=6,ou=Timetable,o=VITELS,c=CH"
    by dn="uid=admin,ou=Staff,o=VITELS,c=CH" write
    by dn="uid=jampen,ou=Staff,o=VITELS,c=CH" write
    by * read

access to dn="*,mid=6,ou=Timetable,o=VITELS,c=CH"
    by dn="uid=admin,ou=Staff,o=VITELS,c=CH" write
    by dn="uid=jampen,ou=Staff,o=VITELS,c=CH" write
    by * read
```

These three rules define access to the module entries in the `Modules` and in the `Timetable` directory subtree. Such rules need to exist for each module (the example shows the rules for module 6). The first rule describes access to one specific module within the `Modules` subtree, where important module information is stored. Write access is granted to the global administrator and to a specified module administrator. Each involved educational institute develops and maintains at least one module for the VITELS project. For each module, the responsible person should be listed here as a module administrator. Module administrators must have a VITELS staff member entry in the `Staff` directory subtree because they are recognized by the scheduling script as administrators only if they authenticate to the `Staff` subtree. As a consequence, a module administrator listed in the student directory subtree will not be shown the administration menu below the online timetable.

The second entry specifies access to one module's entry in the `Timetable` subtree. The third entry defines the access rules for the subtree or, to be more precise, the slot entries of the specified module. Again, write access is granted to the global administrator and to the module administrator. Everyone else is just allowed to read the entries.

In the example above, only one module administrator is registered. Of course, there could be more than one for each module. It is also possible to split the module administration task into two realms. The professor, for example, could intend to define the module information (timeslot properties), but let an assistant do the administration tasks for slot reservations. So, two different administrators could have been listed above, one for the professor in the first entry and another one for the assistant in the following two entries.

5.2 Web-based User Interface

This Section shows the web-based user interfaces of the implementations described in detail in Chapter 4 and explains how they are supposed to be used.

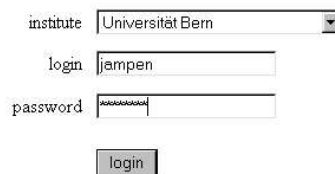
5.2.1 Online Timetable and Administration Interface

The online timetable and the administration menu are a user interface to the scheduling script. Due to the fact that students are visiting an online course and intend to access the timetable through the Internet, the user interface is programmed as a set of websites that can be viewed in every browser. The websites do not contain special code that requires certain plug-ins, they are just shown to the user as normal HTML pages containing basic JavaScript code.

Login

At the beginning, the user (e.g. student or administrator) is prompted to enter his username and password. He also has to choose from a pull-down list to which educational institute he belongs (see Figure 5.1). Module administrators choose “VITELS Staff” in order to get the appropriate privileges.

VITELS Timetable Login



The screenshot shows a web form titled "VITELS Timetable Login". It contains three input fields and a button. The first field is a dropdown menu labeled "institute" with "Universität Bern" selected. The second field is a text input labeled "login" with the text "jampen" entered. The third field is a password input labeled "password" with masked characters. Below the fields is a button labeled "login".

Figure 5.1: Login Website

If the login is successful, the user is directly brought to the online timetable. If not, the same login site is displayed again.

Timeslot Reservation

After a successful login, the current week’s timetable of one module is displayed (see Figure 5.2). There are three symbols used in order to visualize the state of each timeslot within the timetable. A red cross indicates that the slot has already been reserved by another student. A green circle shows that this slot is still available and can be reserved simply by clicking on the circle. The blue check mark is used to mark slots that have been reserved by the user himself. These slots can be freed by clicking on the blue symbol.

Timetable



Figure 5.2: Timetable

Above the timetable the displayed week is indicated. On the left and on the right side links have been placed in order to change to the timetable of the previous or the next week, respectively. On top, there is a list of all available modules. The module that is currently selected is marked with a bigger font face. If a user clicks on a module id, the timetable of that module is displayed. The time period stays the same as before changing the module. In the case that the directory does not contain entries for the currently selected week and module, the message “no timeslots available” is displayed instead of the timeslot information symbols.

LDAP Directory Administration

If a VITELS administrator is logged in, an administration menu is displayed below the timetable (see Figure 5.3).



Figure 5.3: Administration Menu

An administrator is allowed to see who has reserved which slot. This allows him to track students that reserve lots of slots and, thus, block other students. A module administrator is even

allowed to free slots, but only for modules he administrates. In order to free a slot he has to click on *view names* first and then, select the slot he likes to free. When the administrator moves the mouse over a username the user's real name appears within a little window. The administrator can only free slots, when clicking on a username whose slot he intends to free. It is not possible to free slots without seeing the corresponding username (e.g. when seeing only the symbols). This is done not only for error prevention but also because a module administrator should be allowed to reserve slots as well and, thus, being able to solve the module or parts of it himself in order to make sure everything works as planned.

When an administrator clicks on *add/remove slots*, he is presented a page where he can add and remove timeslots (see Figure 5.4).

Add/Remove LDAP Entries (Slots)

module	<input type="text" value="6"/>	(module id)
starting date	<input type="text" value="20020506"/>	(YYYYMMDD)
ending date	<input type="text" value="20020512"/>	(blank = same as starting date)
expiry date	<input type="text" value="20020513"/>	(blank = slot never expires)
<input type="button" value="add"/> <input type="button" value="remove"/>		

Figure 5.4: Adding and Removing Timeslots

The first field has to be filled with the module identification. Unfortunately, this cannot be filled automatically because the access restrictions for administrators are not saved within the LDAP directory tree but in the LDAP server's configuration file. Thus, it is not possible for the scheduling script to determine whether an administrator is allowed to do changes to just a single module, to several or to none of them. This is not a security problem because if an administrator tries to add or remove slots to or from a module he does not administrate, the LDAP server daemon prohibits the changes. There is a status report field in the administration menu that reports whether and how many entries have been added or removed. The second and third field must be filled with the date for the first and last slot to be added or removed, respectively (the format for the date is YYYYMMDD). The last field allows the administrator to specify an expiration date for the slots he intends to add. This field is ignored when removing slots. A daily cron job (see Section 4.3.2) checks the LDAP directory tree for expired slots and removes them automatically. Slots without expiration date will have to be removed manually by the module administrator. Before any changes to the LDAP directory tree are performed, the administrator is displayed a pop-up window, explaining the steps that will be executed. Now, he has the possibility to commit the changes or to abort without changing anything.

By clicking on *change module settings* the administrator is shown a page where the slot duration, the number of slots per day and the starting time of the first slot can be changed (see Figure 5.5). **Caution:** This should be done only at the beginning of a semester because all information stored within the module's timetable will be deleted when changing these values.

Change Module Settings

module	<input type="text" value="6"/>	(mid)
first slot	<input type="text" value="0000"/>	(HHMM)
slot length	<input type="text" value="3"/>	(duration of a slot in hours)
number of slots	<input type="text" value="8"/>	(number of slots per day)
<div><input type="button" value="load"/> <input type="button" value="set"/></div>		

Figure 5.5: Change Module Settings

The first field takes the identity of the module to change. Administrators are strongly recommended to press the *load* button after entering the module identity in order to check the current values before performing unnecessary changes. The second field takes a four digit number specifying the starting time of a day's first slot (the format is HHMM). Starting with this time, all the other slot times will be calculated without any break in between. The duration of a single slot can be entered in the third field. Module administrators should pay attention to specify a number representing the duration in hours. The last field allows to specify how many slots you would like to make available per day. It is important to enter reasonable values, otherwise the execution will fail or produce unwanted results. Before any changes to the LDAP directory tree are applied, the administrator has to validate the changes and press the OK button in a pop-up window.

5.2.2 VITELS Staff Website

The staff member list on the VITELS website [6] described in Section 4.4 displays the contact information of the staff members. The first page lists all VITELS staff members (see Figure 5.6) and the educational institute they are belonging to.

After clicking on a single entry, detailed information for this person is displayed (see Figure 5.7). The details contain links to the homepage of the staff member, email address, phone and fax numbers and the corresponding educational institute. The button *change entry* placed below the table can be used in order to change the own personal details, while the button *change password* can be used to change the own password.

After clicking on *change entry*, the personal details are presented in the same table as described before, but the values are written in text field which allows staff members to make the appropriate changes (see Figure 5.8). Each staff member can only change its own entry because a password has to be specified. The button *save changes* can be used to write the changes to the LDAP directory.

When a staff member intends to change its password, the old password has to be supplied together with the new password. This new password must be entered two times in order to prevent mistyping (see Figure 5.9).

Staff

Name	Organisation
Bergia Amine	University of Geneva
Braun Torsten	University of Berne
Casties Robert	University of Berne
Di-Vitantonio Giuliano	HP

Figure 5.6: VITELS Staff List

Jampen Thomas	
Organisation	University of Bern
Telephone Number	+41 31 631 33 12
Fax Number	+41 31 631 32 62
Mail	jampen@iam.unibe.ch
Homepage	http://www.iam.unibe.ch/~jampen/

change entry

change password

Figure 5.7: VITELS Staff Details

Jampen Thomas	
Organisation	<input type="text" value="University of Bern"/>
Telephone Number	<input type="text" value="+41 31 631 33 12"/>
Fax Number	<input type="text" value="+41 31 631 32 62"/>
Mail	<input type="text" value="jampen@iam.unibe.ch"/>
Homepage	<input type="text" value="http://www.iam.unibe.ch/~"/>
password	<input type="password" value="*****"/>

save changes

Figure 5.8: Change VITELS Staff Details

Jampen Thomas	
Old Password	<input type="password"/>
New Password	<input type="password"/>
Retype New Password	<input type="password"/>

Figure 5.9: Change VITELS Staff Password

5.2.3 VITELS Staff Address Book

The above described staff member list can also be accessed using Netscape's address book. Figure 5.10 shows the configuration window for accessing a directory. `Description` is the name of the address book, `LDAP Server` takes the IP address or the domain name of the directory server and the base dn can be specified in the `Search Root` field. Verify the port and check `Login with name and password` if anonymous binding is not allowed.

When selecting the newly created VITELS staff directory in the address book list, a pop-up window prompting for a username and a password appears (see Figure 5.11). The username that must be entered here is the email address stored in the personal details in the directory tree. After entering the password and clicking the OK button, the address book can be searched. If all addresses are to be displayed, the value for the name to search is `*` (see Figure 5.12).

Directory Server Property

General | Offline Settings

Description:

LDAP Server:

Search Root:

Port Number:

Don't show more than results

☐ Secure

☒ Login with name and password

☐ Save Password

Figure 5.10: Netscape - VITELS Staff Configuration



Figure 5.11: Netscape - VITELS Staff Password

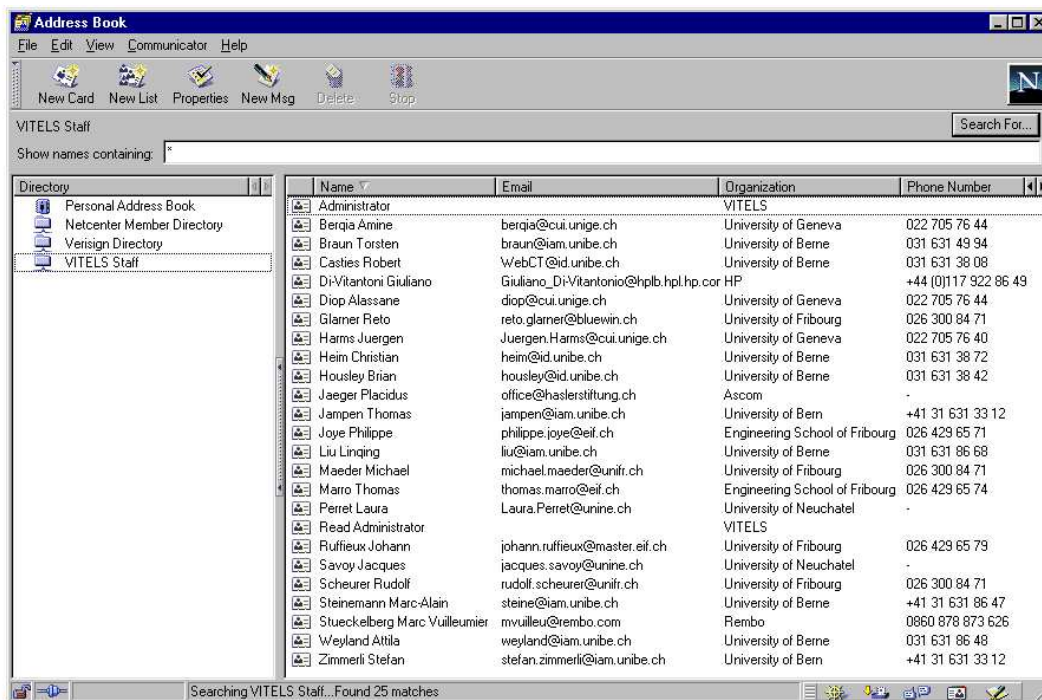


Figure 5.12: Netscape - VITELS Staff Address Book

6 Integration of Security Architecture and Laboratory Hardware

The integration of the security architecture and the laboratory hardware will be shown based on Stefan Zimmerli's diploma thesis *Gateway for a Remote Laboratory* [4]. This diploma thesis is still in process (May 2002) but it is partially described in the paper *Architectural Issues of a Remote Network Laboratory* [1]. Chapter six of this paper describes the VITELS IP Security module as follows: *"The isolated laboratory network was connected to a portal server and additional connections were made with the routers over serial links. The portal server of this prototype implementation manages the routers and the hosts. [...] The IP Security module consists mainly of two parts. In the first part students learn to configure two routers in order to set up a secured VPN tunnel between the routers. In the second part, students have to perform measuring tasks like traffic analysis on the VPN link [...] with and without encrypted traffic and analyzing traffic [...] to verify whether the exchanged traffic is really encrypted."*

Students that intend to access the laboratory hardware, connect to the Portal Server using the Java SSH Applet. From there, they are automatically redirected to the host - using the SSH protocol again - or to the routers - using a serial line connection. These Java Applets are launched from a website that should be accessible by the current user only. This website can be protected using two different means. One possibility is to integrate an Apache module that authenticates users based on the VITELS LDAP directory, the other one is to use the PHP-based API for accessing a module's current user. These possible solutions are described in the following Sections.

6.1 Apache Authentication Based on LDAP

The Apache webserver [9] offers the possibility to restrict access to certain webpages with `.htaccess` files. These files can be placed within the directories to be protected from unauthorized access.

Usually, these files refer to a password file which looks similar to a Unix standard `passwd` or `shadow` file. The password file contains usernames and the corresponding encrypted password. Every time someone opens a browser in order to access webpages that are located within a protected directory he must enter his username and password in order to authenticate himself to the webserver. Once authenticated, the user can access all restricted directories and subdirectories without being prompted for the password again. The session holds as long as the browser is open.

If an LDAP server is running, the Apache webserver can use an LDAP module in order to access an LDAP directory. The user authentication on websites can be done with the help of the LDAP directory instead of the password file and, thus, it is possible to reduce the administrative overhead of maintaining an additional password file.

The proposed LDAP authentication module for Apache is called `auth_ldap` and was written by Dave Carrigan [18]. `auth_ldap` is designed to have excellent performance and to support Apache version 1.3.x on both Unix and Windows. The use of this module is recommended as its configuration is easy and plain, although there are several other modules available on the web [10]. Furthermore, the new version of Apache - Apache version 2.0.x - comes with integrated LDAP authentication support. The directives used there are exactly the same as the ones described below. Thus, the other modules are not recommended as they use another syntax.

6.1.1 Installation

If the `auth_ldap` module is intended to be installed on a Debian/GNU Linux system using a pre-compiled Apache webserver, the appropriate package `libapache-auth-ldap` can be downloaded from the Debian homepage [19]. The following command automatically installs the package:

```
debian:~# dpkg -i libapache-auth-ldap_version_system.deb
```

It is also possible to use `dselect` or `apt-get`. Dependency problems with other packages will be displayed automatically.

If the operating system is not Debian/GNU Linux, there is a `tar.gz` archive on the module's website [18] that can be downloaded and extracted. Further instructions for compilation can be found on this webpage as well.

6.1.2 Apache Configuration

In order to load the module, the appropriate `LoadModule` directive has to be added to the `httpd.conf` file:

```
LoadModule auth_ldap_module /usr/lib/apache/auth_ldap.so
```

Finally, the webserver has to be restarted using the command:

```
debian:~# /etc/init.d/apache restart
```

It has to be considered that the `.htaccess` files can only be used if the directives specified within the Apache configuration file are allowed to be overridden by the `.htaccess` files. This can be achieved by changing the parameter `AllowOverride` in the `httpd.conf` file from `None` to `AuthConfig`.

6.1.3 .htaccess Files

An `.htaccess` file is a special file that can be placed within any directory below the directory specified by `DocumentRoot`. This file prevents the directory and all its subdirectories from being accessed without successful authentication. A simple `.htaccess` file looks as follows:

```
file: .htaccess

AuthType Basic
AuthName "Restricted Area"
AuthLDAPURL ldap://webct.unibe.ch:389/ou=users,o=Universitaet Bern,c=CH?uid
require valid-user
```

Most often `Basic` can be used for the `AuthType` and any quoted string for `AuthName`. This string is displayed when the user is prompted for username and password. These two directives are mandatory. The next line contains the LDAP server and the port followed by the base dn from where the search will be started. After an interrogation mark, the attribute to be used for the authentication is specified. Finally, a `require` directive is needed in order to indicate which users should be accepted. In this example `valid-user` is used, which means that every user can access the page if he provides a valid username and password.

6.1.4 LDAP Directives

The most important directives are listed and described here. For further details refer to the `auth_ldap` homepage [18].

AuthLDAPBindDN and **AuthLDAPBindPassword** can be specified if the module has to authenticate itself before performing the search.

AuthLDAPDereferenceAliases allows to specify how aliases should be dereferenced during the LDAP operations. `never`, `always`, `searching` and `finding` are allowed. Default is `always`. For further information see Section 2.1.3.

AuthLDAPEnabled allows to completely disable LDAP authentication when set to `off`. This can be useful if `auth_ldap` is used but needed to be disabled within certain subdirectories.

AuthLDAPGroupAttribute specifies the attribute used by the LDAP server to store members of a group. This directive can be specified multiple times. Default values are `member` and `uniquemember`.

AuthLDAPUrl specifies the LDAP search parameters to be used. The format is the following: `ldap://server:port/basedn?attribute?scope?filter`. The values `server` and `port` specify the LDAP server to use for the search. Default is `localhost` and `389`, respectively. In order to list multiple servers they can be separated by spaces. `basedn` specifies the base dn for all LDAP searches. `attribute` determines the attribute that should be compared with the supplied username. The default is `uid`. `scope`

specifies the scope to be used for LDAP searches. Possible values are `one` or `sub` (see Section 2.1.3). Default is `sub`. `filter` allows to specify a valid LDAP filter. Default is `(objectClass=*)`.

Below the authentication directives, the following authorization directives can be specified:

require valid-user allows any user that is successfully authenticated.

require user specifies a list of allowed user. A user is defined by the attribute used in the `AuthLDAPUrl`.

require group allows access to any successfully authenticated user of the given group. The group members must be listed in the attribute given by `AuthLDAPGroupAttribute`.

Note: The group dn must be a full dn.

require dn allows to specify distinguished names of the allowed users.

All the above described directives can be entered in a `.htaccess` file or within a `<Location /some/directory>` directive of the `httpd.conf` file.

6.1.5 Usage of `auth_ldap` for the VITELS Project

The `auth_ldap` module allows to dereference aliases and, thus, can be configured to check the `ou=Modules, o=VITELS, c=CH` directory subtree. This allows the use of Apache authentication in order to restrict access to certain webpages to the currently registered student. A sample `.htaccess` file looks as follows:

```
file: .htaccess

AuthType Basic
AuthName "Restricted Area"
AuthLDAPDereferenceAliases finding
AuthLDAPEnabled on
AuthLDAPURL ldap://webct.unibe.ch:389/mid=6,ou=Modules,o=VITELS,c=CH?uid
require valid-user
```

Alias dereferencing is set to `finding` in order to dereference already the base object because the aliased entry is directly specified in the base dn `mid=6, ou=Modules, o=VITELS, c=CH`. Then, authentication is enabled. After that, the LDAP server and the port are specified, followed by the base dn - pointing to the module to be accessed - and the attribute to search for (`uid`).

This `.htaccess` file can be placed in the directory containing the webpages that load the Java SSH Applets. This prevents unregistered users from accessing the course and the Applet.

6.2 Using the API for Accessing a Module's Current User

In order to send cookies with slot information to the browser when students visit a module's website, the module's current user and the corresponding slot settings have to be accessible. The file `vitelsldap.inc.php` offers an easy to use interface in order to check whether a user is the current user of a module, whether the supplied password is correct and what the starting time and the ending time of the current slot is.

The needed functionality is provided by the class `VitelsLDAP` which has already been described in detail in Section 4.5. There are several functions that can be called in this API, but the most convenient way to verify a current user and to get the slot boundaries is to use the function `get_slot_if_current_user($dn, $pwd, $mid)`. This function returns an array containing the starting and ending time of the slot only if the user specified by `$dn` is the current user of the module `$mid` and if his password stored in `$pwd` is correct. Otherwise, `false` is returned.

file: **APIexample1.php**

```
<?php

require('vitelsldap.inc.php');

// when composing the dn make sure you know from which
// institute the user comes!
// choose one of:
// $unibe_base, $unifr_base, $unige_base, $unine_base, $ingfr_base
$dn = "uid=username,$unibe_base";

$pwd = "some-password";

// for example module 6
$mid = "6"

// access to the LDAP directory
$ldap = new VitelsLDAP();
$result = $ldap->get_slot_if_current_user($dn, $pwd, $mid);

if ($result) {
    // the date is specified as seconds since 1.1.1970
    $starting_time = $result["starttime"];
    $ending_time   = $result["endtime"];

    // get printable date, e.g. 20020426 1830
    $start = date("Ymd Hi", $starting_time);
    $end   = date("Ymd Hi", $ending_time);

    // ...
}
else {
    print "wrong password or not current user!";
}

// ...

?>
```

The above code shows a little example on how to use the class `VitelsLDAP`. The distinguished names for the different educational institutes are saved in the variables `$unibe_base`, `$unifr_base`, `$unige_base`, `$unine_base` and `$ingfr_base`. In order to create a valid dn, the string `uid=username` has to be prepended to one of the base dns. After that, an new instance of the class `VitelsLDAP` has to be created in order to access the above described function. The starting and ending times returned within the array are represented as the seconds past since January, 1st 1970. They can be easily converted to the preferred format using the PHP `date($format, $time)` function. The `$format` variable contains a string specifying the desired time format. In the above example `Ymd Hi` is used in order to get a date formatted as follows: `20020426 1830`. Other possible formats are described on the PHP homepage [35].

A useful function is `authenticate_user($dn, $pwd)` which only returns `true` if the user specified by `$dn` is registered and if the password given by `$pwd` is valid. In order to authenticate a user, the connection to the LDAP server has to be established already. This can be done using the function `connect()` before authentication. After successful authentication, the user has to be authorized for accessing the desired module. In order to do that, the function `authorize_user_for_module($mid)` can be called. This function returns `true` if the authenticated user is, in fact, the current user of the module specified by `$mid`. After the authorization, the LDAP connection can be closed using the function `cleanup()`. The code below shows a little example.

file: **APIexample2.php**

```
<?php

require('vitelsldap.inc.php');

$dn = "uid=username,$unibe_base";
$pwd = "some-password";
$mid = "6"

$ldap = new VitelsLDAP();
$ldap->connect();

if ($ldap->authenticate_user($dn, $pwd)) {
    if ($ldap->authorize_user_for_module($mid)) {
        // authorization successful
        // ...
    }
    else {
        print "not current user!";
    }
}
else {
    print "wrong password!";
}

$ldap->cleanup();

?>
```

The second file provided with this little API is `ldapererror.php`. This file contains the HTML code for an error message to display if the LDAP server is not reachable at the moment.

6.3 LDAP-based Shadow-File Modification

The laboratory hardware of module 6, *IP Security*, is placed in a local network and, thus, is not accessible from the Internet directly. The connections have to be established through the Portal Server. On the Portal Server a shell account for each laboratory resource has been created (e.g. user *cisco2600* for the cisco 2600 router or user *host1* for the host with the name *host1*, etc.). A login to the Portal Server with one of these login names is redirected to the corresponding laboratory resource. This can be easily done by setting the default shell of these users to a program used to communicate with the routers or hosts (e.g. SSH for accessing the hosts or Minicom for connecting to the routers).

The problem is, that always the same username has to be used for connecting to a certain router or host, because the Portal Server determines where to redirect the request by the username used for the login. As a consequence, students are not able to log in with their own username and password. In order to grant access only to the user that has currently reserved the timeslot, the passwords of these special users *cisco2600*, *host1*, etc. have to be changed at the end of a slot or when a student frees the slot before the slot is over. The simplest solution is to change the passwords for the redirection users always to the password - stored in the LDAP directory - of the current user. To be more precise, the user that has reserved the current slot of the specified module can login to the routers and hosts by specifying the corresponding username and his own password. His password is only valid during his timeslot and will be replaced by the password of the succeeding slot's user.

In Unix environments passwords are stored in the *shadow* file. This file has to be changed by a script in order to update the password of the special users. This script has been implemented in Perl [36] and is called *update_shadow.pl*. It retrieves the current user of the specified module and writes his password into the shadow file. Additionally, the script removes temporary files and closes open connections to the laboratory resources if the current user has changed.

This script has to be installed as a cron job in order to make sure that changes are recognized immediately. Such a cron job needs to be run every minute and not only, when the slot ends, because imagine a student that finishes his work before the slot is over. If he is fair enough to free the slot in order to allow other students to reserve this very slot, such a change would not be recognized. Thus, the appropriate *crontab* file looks as follows:

```
----- file: crontab -----
SHELL=/bin/sh
# crontab file for the VITELS project
#
# check for current user every minute and update shadow if necessary
# m h dom mon dow command
* * * * * /usr/local/bin/update_shadow.pl
```


7 Related Work

7.1 Shibboleth

Shibboleth is a project of Internet2/MACE (Middleware Architecture Committee for Education) and IBM. The aim of Shibboleth [37] is to develop an architecture that controls web access based on already defined security standards. Shibboleth intends to provide authentication and authorization for web-based applications. Accounting is not planned to be included.

Shibboleth has the advantage that inter-institutional authentication and authorization has been included from the beginning. A user is able to control which information is transmitted to which resource. Authentication is done by the user's home organization, authorization by the resource provider. Another advantage is that Shibboleth seems to be a scalable architecture that will be released as Open Source.

The disadvantages are that Shibboleth is designed for web-based applications only. It depends on HTTP redirection which means that Shibboleth is limited to browser applications and does not allow to control direct connections to the hardware as needed in the VITELS project. Shibboleth uses hard coded answers, an LDAP back-end is not (yet) supported. Furthermore, Shibboleth is not yet stable, the alpha version has just been released. That makes it impossible to deploy Shibboleth for the VITELS project because the VITELS project needs a stable solution now.

7.2 PAPI

The Point of Access to Providers of Information (PAPI) system [38] provides control mechanisms for accessing restricted information resources in the Internet. It does not specify a single authentication mechanism, each organization is able to use its own scheme. Based on this authentication mechanism, the organization provides the required attributes for access control decisions to the information providers. These access control mechanisms are designed to be transparent to the user and ought to be compatible with the most common web browsers. After a successful authentication the user gets a list containing URLs of available resources and timeslot information for each resource. In the meantime, all necessary points of access are contacted and a webpage informs about the authentication results for the different resources. Now, the user is able to access the resources during the specified timeslot.

Advantages are that there is no need for additional software on the user's side, common authentication schemes can still be used, no adaptations to new schemes are needed and the

software is freely available. Furthermore, PAPI is already operational in Spain in order to control access to libraries.

The disadvantages are the followings: The system is designed to be web-based and does not support access mechanisms to laboratory hardware. It is assumed that the desired resources are stored on web servers. The authentication server of the home organization has to know all the available services for each employee, because a list of these services has to be sent to the user. All available points of access need to be accessed after authentication in order to perform the authorization. This list of resources will presumably be rather large, while a user is - for the moment - only interested in a small subset. This implies that this architecture will not scale very well. Furthermore, the distribution of the list of available services to all points of access might be seen as a privacy problem.

7.3 GASPAR

GASPAR (Guichet d'Accès Sécurisé aux Prestations Administratives et autres Ressources) is a system used to authenticate students and professors at the EPFL (École Polytechnique Fédérale de Lausanne) [39]. Users are identified by GASPAR using a unique ID called SCIPER and a corresponding password or by cookie, if present. If a user intends to access restricted information, the application server redirects him to GASPAR in order to get a valid session identifier stored in a cookie. GASPAR redirects the user back to the application server that verifies the cookie and redirects the user to the desired resources.

GASPAR is a simple but working, home-made solution. But it is limited to a single organization. An extension to a global Authentication and Authorization Infrastructure (AAI) requires a major re-implementation. This makes it unusable for the VITELS project as inter-organizational use is mandatory and a working solution is needed now.

7.4 FEIDHE

FEIDHE is a project of the Finnish higher education [40]. The goal is to implement a PKI based on user identification with smart cards. These smart cards are mainly used for storing certificates.

Of course, the FEIDHE project offers interesting experiences in deploying a PKI using smart cards, because smart cards might be one of the main technologies of the future. But on the other hand, this is a major disadvantage because the whole architecture depends on smart cards. As Swiss educational institutes already use different mechanisms of identifying students, they cannot be forced to change to smart cards immediately and, thus, FEIDHE does not seem to be appropriate for the VITELS project. Furthermore, FEIDHE only specifies authentication - authorization is not a key element of the project. The current status of the project is unclear as no evaluation reports are available.

8 Conclusions and Outlook

8.1 Conclusions

Choosing PKI as a security architecture seemed to be a good choice. All necessary connections could have been secured using already existing, well-known and tested, standard protocols. IPsec tunnels could have been even built between Unix and Linux computers. There is a Howto developed and maintained by the Informatikdienste (ID) of the University of Bern that explains the steps needed to be performed by other educational institutes in order to securely connect their module to the VITELS directory server using IPsec. This Howto can be found on the VITELS homepage [6]. Up to now, the PKI has not been implemented completely, the Certificate Authorities (CAs) have not yet been set up. All required certificates have been created and distributed by an administrator and not by a dedicated CA server.

During the development phase of this project, the LDAP servers and the protocol have been tested a lot. The performance has always been very satisfactory, LDAP really seems to be the optimal student data management system for the VITELS project. Its uses are manifold, very comfortable, for example, is its use together with the Apache webserver in order to authenticate web users. The installation and configuration of the module is easy and straight forward as seen in Section 6.1.

The scheduling mechanism is working reliably and the web-based user interface has been designed to conveniently allow to perform every action needed by the students as well as by the module administrators. The daily cron job that removes no longer required timeslots prevents the directory from growing more and more, and prevents performance losses due to unnecessary large directories.

The little PHP package offering an easy to use API for accessing a module's current user and slot boundaries has been integrated into the work of Stefan Zimmerli (*Gateway for a Remote Laboratory* [4]). This API meets all the requirements for other module developers and can be integrated easily into their work.

Within the test network, the scheduling mechanism has been successfully tested with multiple LDAP servers. These servers have been connected using IPsec tunnels. One server stored the main VITELS directory tree and one university tree. The other server stored additional university trees. These trees have been connected using referrals (explained in Section 2.1.7). One major problem emerged when using aliases and referrals together (as done in the directory subtree `ou=Timetable,o=VITELS,c=CH`, where aliases to students are stored. These aliases can point via a referral to another LDAP server). When searching for a student - that is stored on another server - using `uid=student,ou=users,o=institute,c=CH` as the base dn

(instead of using `ou=users,o=institute,c=CH` as the base dn and specifying the filter `"(uid=student)"`), all users from the specified institute are returned instead of just the entry of the specified student. The base dn is chosen to contain the user id because the dn of the alias entry stored in the timetable looks exactly like that. This problem could have been fixed for the scheduling implementation, but the Apache authentication module fails to authenticate users stored on another directory server if accessed as described in Section 6.1.5. At the moment, it is planned to use a centralized student directory, but the test results with the scheduling system open the possibility of using separate directories later. This shows that the chosen architecture should be smoothly extensible.

After the successful test phase, the whole scheduling system (the scheduling script, the web-based user interface, the LDAP directory tree and the corresponding cron jobs) has been integrated into the productive system of the ID. This process allowed to correct minor bugs that have been detected due to the slightly different setup of the two environments. The productive system is operational at the moment and the whole project is tested within the ordinary network laboratory using module six *IP Security* as an example. In accordance with some feedback received from student that solved this lab module, it can be said that the scheduling system and the authentication mechanisms work reliably. There seem to be minor problems on the side of the Informatikdienste (ID) with rebuilding the entire LDAP directory every night. This has to be done because the student entries are extracted from the Course Server running WebCT [42]. Another problem appears when using some versions of Netscape Communicator. There seem to be a problem when minimizing the Java SSH Applets. But generally, the feedback has been positive.

SWITCH is very interested in the work of the VITELS members and, thus, the collaboration between the University of Bern and SWITCH has been deepened. As a consequence, the relations between the universities and SWITCH are ameliorating. This could result in a standardized, federal authentication and authorization system in the near future.

8.2 Outlook

8.2.1 Reservation Limitations

One could implement a limitation mechanism that prevents students from reserving unreasonable amounts of timeslots. At the moment, this task is performed manually by the module administrator. His job is to verify that every student gets the chance to reserve slots. He has the means to free slots if a student acquires too many of them.

The design of such a mechanism is trickier than it looks. Questions like “What is a reasonable limitation tactics?”, “How many students will access the lab?” and “How long should the limitation periods last (one week, two weeks, etc.)?” have to be answered first. A possibility is to simply allow the reservation of further slots for the same week only after every student has reserved exactly one slot. This approach does not seem to be very comfortable to the students because they depend on each individual. If for example a single student becomes ill, the whole system is blocked.

Another approach could be to divide the slots in two levels: *first level* and *second level* slots. The first level slots are slots that take place during the office hours. Second level slots are evening or nightly slots. Only one first level slot can be reserved per week (or per two weeks), but students have the possibility to reserve additional two or three second level slots per week (or per two weeks). This would allow every student to get one daily slot per week and the right to reserve additional nightly slots without waiting for other students to finish their reservations.

These changes can be easily performed within the file `ldapqueries.inc.php`. There is a function called `reservation_allowed($user_data, $mid, $slot)` that can be extended. At the moment, this function only checks whether the student has provided a valid username and the corresponding password.

8.2.2 Multiple LDAP Servers

This issue has already been solved but tested only in a test network. It has not yet been applied to the productive system. The institute maintaining the VITELS directory subtree needs to create referrals for each university that intends to maintain its own LDAP directory tree. Such a referral entry on the VITELS LDAP server looks as depicted in Figure 8.1.

```
dn: o=University-of-Geneva,c=CH
objectclass: organization
o=University-of-Geneva

dn: ou=users,o=University-of-Geneva,c=CH
objectclass: referral
objectclass: extensibleobject
ou: users
ref: ldap://ldap.unige.ch/ou=etudiants,o=University-of-Geneva,c=CH
```

Figure 8.1: Referral Entry for Multiple LDAP Servers

Important for compatibility reasons is, that the organizational unit `ou` on the VITELS directory server is `users`. This must not be the case on the university side, the above example shows an organizational unit `etudiants`. This is no problem because the subtree is included into the VITELS tree under the name of the specified `dn` (where the organizational unit is `users`).

The only thing a university has to consider is the fact that the student entries must contain at least the object classes `person` and `account`. This should not be a problem because these object classes are really standard.

An open problem is the fact that when chasing referrals with a base `dn` that is more specific than the referral (e.g. the base `dn` `uid=student,ou=users,o=institute,c=CH`) some hierarchy levels (in this example `uid=student`) seem to get lost. As a consequence the entire subtree is returned which is useless and not efficient at all. This problem could have been solved by changing the implementation of the scheduling script but still remains when using Apache LDAP authentication. As LDAP is a still developing standard and referrals are not yet supported directly by the server (the client implementation has to chase them), one can assume that this problem will be fixed in the near future.

8.2.3 Certificate Authorities

Another open issue is that in order to build a complete PKI, each university needs to have its own Certificate Authority (CA) that creates the certificates for the university web server, for the Portal Server and for the IPsec connections. These CAs should have their own certificates signed by another approved CA such as VeriSign [41]. It is planned to implement dedicated CA servers in order to complete the PKI.

8.2.4 VITELS for Students all Over the World

Accounting consists of several parts, of which billing and trend analysis do not have priority at the moment, but could be an issue later on, when access to the lab is no longer restricted to registered students from Swiss educational institutes but open to students from all over the world. One can assume that such a service would raise the needs of a billing system in order to charge foreign students or universities that are accessing the VITELS laboratories.

Appendix

List of Figures

2.1	Sample LDIF Entries	5
2.2	A List of Common Attributes and Abbreviations	6
2.3	Definition of the Attribute <code>name</code>	7
2.4	Definition of the Object Class <code>person</code>	7
2.5	An Alias Entry	9
2.6	Two Branches of a Company with Separate LDAP Servers	10
2.7	A Referral Entry	10
2.8	Kerberos Overview	12
3.1	General Architecture	18
3.2	Security Architecture	21
3.3	Domain-based Structure	22
3.4	Organization-based Structure	22
3.5	Directory Structure for Universities	23
3.6	Directory Structure for VITELS-specific Entries	23
3.7	A Student Entry	23
3.8	A Timetable Entry	24
3.9	A Timeslot Entry	24
3.10	A Module Entry	25
5.1	Login Website	41
5.2	Timetable	42
5.3	Administration Menu	42
5.4	Adding and Removing Timeslots	43
5.5	Change Module Settings	44
5.6	VITELS Staff List	45
5.7	VITELS Staff Details	45
5.8	Change VITELS Staff Details	45
5.9	Change VITELS Staff Password	46
5.10	Netscape - VITELS Staff Configuration	46
5.11	Netscape - VITELS Staff Password	47
5.12	Netscape - VITELS Staff Address Book	47
8.1	Referral Entry for Multiple LDAP Servers	61

Abbreviations

AAA	Authentication, Authorization, Accounting
AAI	Authentication and Authorization Infrastructure
ACL	Access Control List
AH	Authentication Header
API	Application Programming Interface
ASN.1	Abstract Syntax Notation number One
CGI	Common Gateway Interface
DN	Distinguished Name
DSS	Digital Signature Standard
ESP	Encapsulating Security Payload
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Secured Sockets Layer
IKE	Internet Key Exchange
KDC	Key Distribution Center
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
MAC	Message Authentication Code
MD5	Message Digest (one-way hash function)

MIT Massachusetts Institute of Technology

PHP PHP Hypertext Preprocessor

PDF Portable Document Format

PKI Public Key Infrastructure

RFC Request For Comment

RSA Public key crypto system named by its inventors Rivest, Shamir and Adleman.

SHA Secure Hash Algorithm

SSH Secure SHell

SSL Secure Sockets Layer

SVC Swiss Virtual Campus

TCP Transmission Control Protocol

TGS Ticket Granting Service

TGT Ticket Granting Ticket

TLS Transport Layer Security

TTP Trusted Third Party

URL Uniform Resource Locator

VITELS Virtual Internet TElecommunications Laboratory of Switzerland

VPN Virtual Private Network

WWW World Wide Web

XML eXtensible Markup Language

Glossary

Accounting Accounting is the process which measures the resources a user consumes during his session.

Apache Apache is a very common web server running on Unix, Linux and Windows.

Authentication Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be.

Authorization Authorization is the process of giving someone permission to do something. Usually, this is done after successful authentication.

Certificate A certificate consists of the public key and the identity of an entity rendered unforgeable by digitally signing the entire information with the private key of the issuing Certificate Authority (CA).

Certificate Authority A Certificate Authority (CA) is an authority in a network that issues and manages security credentials and public keys for message encryption.

Confidentiality Confidentiality is the non-occurrence of the unauthorized disclosure of information.

Cron Job A cron job specifies a program and a point of time when the given program is automatically run by the cron daemon.

Data Integrity Data integrity ensures that information is not altered in storage or transit by unauthorized persons in a way that is not detectable by authorized users.

Digital Signature A digital signature is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document and, possibly, to ensure that the original content of the message or document is unchanged. Digital signatures are easily transportable and cannot be imitated by someone else. The ability to ensure that the original signed message arrived means that the sender cannot repudiate it later.

Directory Service A directory is similar to a database, but tends to contain more descriptive, attribute-based information. The information in a directory is generally read much more often than it is written. As a consequence, directories don't usually implement the complicated transaction or roll-back schemes that regular databases use. Directories are tuned to give quick responses to high-volume lookup or search operations.

IPsec Internet Protocol Security. IPsec is a developing standard for security at the network or packet layer of network communication. IPsec is especially useful for implementing virtual private networks and for remote user access through dial-up connection to private networks.

Java Applet A Java Applet is a piece of code that cannot be run stand-alone. An Applet needs an existing environment as for example a web browser (e.g. Netscape Communicator or Internet Explorer) that contains the Java Virtual Machine (VM).

Key Pair A key pair consists of a private and a public key.

LDAP Lightweight Directory Access Protocol. Directories containing information such as, for example, names, phone numbers and addresses. LDAP provides a relatively simple protocol for updating and searching such directories. See Directory Service.

Man-in-the-Middle-Attack A man-in-the-middle-attack is an attack where an attacker sniffs packets (reads packets that are intended for someone else) from network, modifies them and inserts them back into the network in a way that communicating parties do not realize. They still think that they are communicating directly with each other.

MAC Message Authentication Code. A MAC is a one-way hash computed from a message and some secret data. Its purpose is to make message alteration detectable.

Non-Repudiation Non-repudiation means that a user cannot deny having digitally signed a document which contains his signature. This is possible with public key cryptography because the user is supposed to be the only one that knows his private key. As a consequence, he is the only one being able to produce the signature.

One-Way Hash Function A one-way hash function is a transformation that converts an arbitrary amount of data into a fixed-length hash. It is called one-way function because it is designed to be computationally hard to reverse the transformation or to find collisions (two different messages with the same hash value). MD5 and SHA are examples of one-way hash functions.

Open Source Open Source refers to the fact that the source code of Free Software is open to and for the world to take, to modify and to reuse.

Passphrase A passphrase is a string of characters longer than the usual password (which is typically from 4 to 16 characters long) that is used for creating a digital signature, for the encryption or for the decryption of a message.

Password-Guessing-Attack A password-guessing-attack can be used in order to attack users that do not seem to use cryptographically secure passwords. This kind of attack is done either by trying names and values related to that person or by using a dictionary (list of often used passwords).

PHP PHP Hypertext Preprocessor (PHP) is a scripting language that is especially suited for web development and can be embedded into HTML.

Private Key A private key is a value - known only to one party - that can be used to decrypt encrypted messages, issue digital signatures and compute the corresponding public key. The private key must be kept private and must not be made publicly available. This term is most often used in the context of public key cryptography and not in the context of traditional (or secret key) cryptography (see also *secret key*).

Public Key A public key is a value that can be used to effectively encrypt messages and verify digital signatures. The public key can be made publicly available, it does not contain secret information.

Public Key Cryptography Public key cryptography is the science of information security that uses *private key* and *public key* pairs for encryption, decryption and signature creation and verification. The problem of the key distribution is solved because the public key can be made publicly available, just the private key is kept as a secret. RSA is an example of a public key crypto system.

Revoked Certificate A revoked certificate is a certificate that contains a special marking indicating that the certified key should no longer be used because the corresponding private key has been compromised.

Secret Key A secret key is a key that is intended for use by a limited number of correspondents for encryption and decryption. This term is most often used in the context of traditional (or secret key) cryptography, not to be confused with *private key*.

Secret Key Cryptography Secret key cryptography is the science of information security that uses the same key (*secret key*) for encryption and decryption. Secret key cryptography often implicates the problem of distributing this secret key among the communicating parties. DES, Triple-DES, RC4 and IDEA are examples of secret key ciphers.

Suspended Certificate A suspended certificate is a certificate that contains a special marking indicating that the certified key is not in use at the moment but can be used again later. It does not indicate that the corresponding private key has been compromised.

Sniffing Network sniffing is the process of gathering information (from a network) that is designated for someone else.

X.500 An electronic directory service (also known as the White Pages). See Directory Service.

Bibliography

- [1] M.-A. Steinemann, S. Zimmerli, T. Jampen, and T. Braun. *Architectural Issues of a Remote Network Laboratory*. 2002.
- [2] M.-A. Steinemann, S. Zimmerli, T. Jampen, and T. Braun. *Didactical Issues of a Remote Network Laboratory*. 2002.
- [3] M.-A. Steinemann, S. Zimmerli, T. Jampen, and T. Braun. *Global Architecture and Partial Prototype Implementation for Enhanced Remote Courses*. 2002.
- [4] S. Zimmerli, M.-A. Steinemann, and T. Braun. *Gateway for a Remote Laboratory*. 2002.
- [5] Swiss Virtual Campus.
<http://www.virtualcampus.ch/>.
- [6] Virtual Internet and Telecommunications Laboratory of Switzerland.
<http://www.vitels.ch/>.
- [7] Swiss Academic and Research Network.
<http://www.switch.ch/>.
- [8] The Open Source Definition.
<http://www.opensource.org/docs/definition.html>.
- [9] Apache HTTP Server Project.
<http://httpd.apache.org/>.
- [10] Apache Module Registry.
<http://modules.apache.org>.
- [11] Lightweight Directory Access Protocol.
<http://www.ietf.org/rfc/rfc1777.txt>.
- [12] The LDAP Data Interchange Format (LDIF) - Technical Specification.
<http://www.ietf.org/rfc/rfc2849.txt>.
- [13] IETF RFC Page.
<http://www.ietf.org/rfc>.

- [14] The COSINE and Internet X.500 Schema.
<http://www.ietf.org/rfc/rfc1274.txt>.
- [15] The String Representation of LDAP Search Filters.
<http://www.ietf.org/rfc/rfc2254.txt>.
- [16] ASN.1 Information Site.
<http://asn1.elibel.tm.fr/>.
- [17] Open Source Implementation of the Lightweight Directory Access Protocol.
<http://www.openldap.org/>.
- [18] An Authentication Module for Apache.
http://www.rudedog.org/auth_ldap/.
- [19] LDAP Authentication Module for Apache.
<http://packages.debian.org/testing/interpreters/libapache-auth-ldap.html>.
- [20] Kerberos: The Network Authentication Protocol.
<http://web.mit.edu/kerberos/www/>.
- [21] Kerberos v4 Web Authentication Plug-in.
<http://www.monkey.org/~dugsong/krb-www/kplugin/>.
- [22] Kerberos v4 Apache Module.
<http://www.monkey.org/~dugsong/krb-www/kapache/>.
- [23] Apache Kerberos Module.
http://stonecold.unity.ncsu.edu/software/mod_auth_kerb/index.html.
- [24] SSL 3.0 Specification.
<http://www.netscape.com/eng/ssl3/>.
- [25] The TLS Protocol.
<http://www.ietf.org/rfc/rfc2246.txt>.
- [26] OpenSSL: The Open Source Toolkit SSL/TLS.
<http://www.openssl.org/>.
- [27] mod_ssl: The Apache Interface to OpenSSL.
<http://www.modssl.org/>.
- [28] OpenSSH.
<http://www.openssh.org/>.

- [29] **MindTerm: SSH Client.**
<http://www.mindbright.se/mindterm/>.
- [30] **Telnet Issues.**
<http://asg.web.cmu.edu/rfc/rfc435.html>.
- [31] **IETF - IP Security Protocol.**
<http://www.ietf.org/html.charters/ipsec-charter.html>.
- [32] **FreeS/WAN Project.**
<http://www.freeswan.org/>.
- [33] **The Source for Java™ Technology.**
<http://java.sun.com/>.
- [34] **PHP Hypertext Preprocessor.**
<http://www.php.net/>.
- [35] **PHP Manual: Date.**
<http://www.php.net/manual/en/function.date.php>.
- [36] **The Source for Perl.**
<http://www.perl.com/>.
- [37] **Shibboleth.**
<http://middleware.internet2.edu/shibboleth/>.
- [38] **PAPI - Point of Access to Providers of Information.**
<http://www.rediris.es/app/papi/dist/PAPI.html>.
- [39] **GASPAR.**
<https://gaspar.epfl.ch/>.
- [40] **FEIDHE.**
<https://hstya.funet.fi/>.
- [41] **VeriSign, The Value of Trust.**
<http://www.verisign.com/>.
- [42] **WebCT.**
<http://www.webct.com/>.