A Framework and Prototyping Environment for a W3 Security Architecture

Gustaf Neumann[†], *Stefan Nusser*[‡]
† *Information Systems and Software Techniques, University of Essen, Germany*
‡ *Management Information Systems,*
Vienna University of Economics and Business Administration, Austria
E-mail: neumann@wi-inf.uni-essen.de, nusser@wu-wien.ac.at

Abstract

[*] The goal of this paper is to identify and describe the services necessary to build a secure corporate intranet and to show how prototypical implementations of these components can be realized to evaluate different approaches and configurations. The paper presents an architectural framework, which identifies the core services necessary for a secure Internet-based communication and information infrastructure. We focus on the authentication service, which is responsible for authenticating users and services. We then show how security infrastructures can be developed and tested using the rapid prototyping environment *Wafe* and the extensible Web browser *Cineast*. We explain how basic operations such as secure transfer and certificate acquisition can be realized and demonstrate the implementation of different certificate validation strategies.

Keywords
Security architecture, authentication, certificate management, WWW-browser

1 INTRODUCTION AND MOTIVATION

With the advent of the large scale deployment of Internet technology as a foundation for electronic commerce, new requirements for document and communication security emerged in interlinked environments. This tendency was reinforced by the use of Internet applications as a low-priced alternative to classical IS development and to commercial groupware products, which lead to the increasing propagation of corporate intranets.

This evolution of the World Wide Web (W3) towards a company's information system and telecommunication infrastructure triggered the development of several enhancements of the original Internet concepts with the goal of providing a secure distributed computing environment. Most of the security proposals are based on asymmetric cryptography and require the establishment of a public key infrastructure (PKI), which refers to the distribution and maintenance of trusted public keys. While the technical concepts are not new, their deployment is difficult for several reasons:

[*]Published in: Proceedings of Communications and Multimedia Security, Athens, September, 1997.

- Deploying Internet security techniques on an enterprise scale is mostly not a technical, but an engineering and organizational problem. Existing security protocols provide only mechanisms not an architecture.
- The use of the same techniques for securing intranet and public Internet communication often leads to a trade-off: Adhering to standards might restrict a company's freedom and flexibility, but is an advantage when internal resources should be made publicly available.
- The purpose of PKIs is changing. While the currently used certificate standards (CCITT 1989) were designed to provide an authentication framework, the motivation behind recent developments in certification techniques is to enhance this mechanism to a means of access control.
- The most important security access points in a W3 security architecture are along the information flows, where the information source is a W3 server, and the primary information sink is a W3 browser. In order to prototype new approaches it is necessary to modify and extend the involved software packages in a substantial way. For popular W3 browsers this is not feasible or might be even impossible (plug-ins are not sufficient). Therefore the development of new security features is practically in the hands of a few companies who can integrate new features in their browsers.

This paper identifies the building blocks of a secure Internet-based communication and information infrastructure. It describes the functionality of these core services and examines different implementation strategies (Section 2). We concentrate in Section 3 on the authentication service used to identify users and services. Section 4 presents our implementation framework for different security schemes. We demonstrate how to extend existing concepts into new directions using an extensible W3 browser. A summary and suggestions for future research conclude this paper.

2 CORE SERVICES

Figure 1 shows our framework of a corporate security architecture. This framework incorporates many results of distributed object security and role based access control. Especially we were inspirated by the work of Yialelis, Lupu and Sloman (Yialelis *et al*. 1996) and Sandhu (Sandhu *et al*. 1996).

The overall goal of our security architecture is to manage access control in a large intranet. This is realized by the *authorization service* shown at the top of Figure 1. The purpose of this service is to determine whether a user has the right to access a specified resource. This decision is usually based on some sort of access control list, which can be used to determine the privileges of a given user with regard to the resource accessed. The authorization service depends on the following supporting components of our framework.

The *authentication service* is mainly concerned with the reliability and validity of public keys. It authenticates users and services in accordance with a well defined policy, a process which is often referred to as "*Management of Trust*". We will take a closer look at the different models and approaches in the next section.

The authentication service uses the *directory service* in order to make information about users and services publicly available. The information propagated by means of the directory service should be sufficient to identify users or services by their commonly known names. In

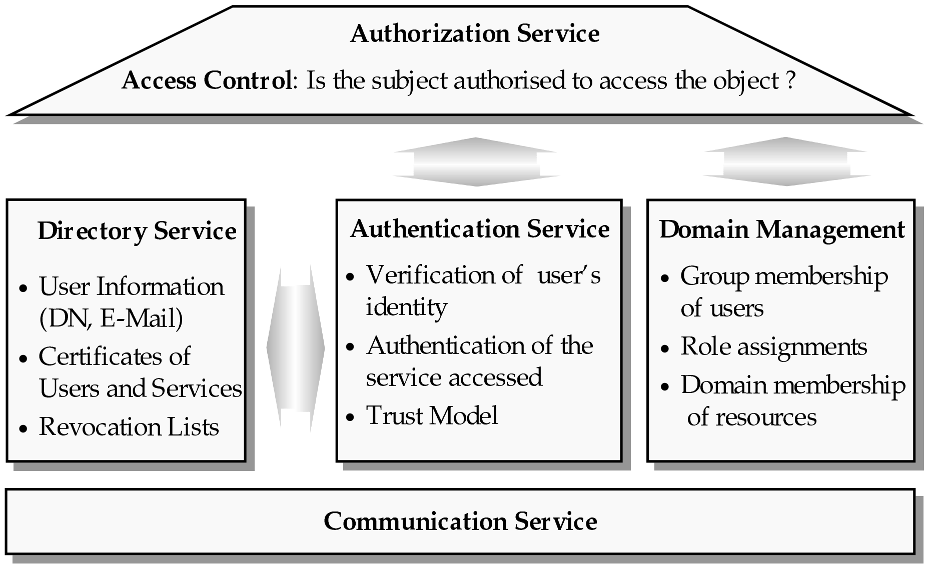

Authorization Service
Access Control: Is the subject authorised to access the object ?
Directory Service
• User Information (DN, E-Mail)
• Certificates of Users and Services
• Revocation Lists
Authentication Service
• Verification of user's identity
• Authentication of the service accessed
• Trust Model
Domain Management
• Group membership of users
• Role assignments
• Domain membership of resources
Communication Service

Figure 1 Security Architecture.

our context it is important that the directory service is the primary device to make public keys available for the public. An emerging standard for accessing directory services is the LDAP protocol (Yeong *et al*. 1995), which is based on the X.500 directory access protocol (DAP). In recent product announcements, the LDAP protocol is also used for retrieving access control lists from a central repository (Xcert 1997).

Since every real world authorization service has to deal with a large number of resources (objects) and users (subjects) it has to use some grouping concept to keep access control manageable. For this reason, our framework includes the *domain management*, which provides information on membership relationships for both users and objects. This service can also be used to maintain role assignments and is therefore a good foundation for the implementation of role based access control (RBAC) in an intranet context.

All services described so far are based on the Internet protocols as a communication infrastructure. See Oppliger (Oppliger 1997) or Lipp and Hassler (Lipp *et al*. 1996) for an overview of different approaches to transmitting encrypted data or distributing public keys over the Internet. The currently most widespread mechanism is Netscape's secure socket layer (SSL) which implements encryption and certificate based authentication at the transport layer (Freier *et al*. 1996). An IETF working group was chartered in order to develop a standardised transport layer security (TLSP) protocol, which will in most aspects be based on SSL. We summarize these mechanisms in Figure 1 as *communication service*.

3 AUTHENTICATION SERVICE

In this section, we further examine the authentication service. We first briefly describe the characteristics of the underlying public key infrastructure and then we focus on the use of certificates as a means of binding a person's identity to a public key.

3.1 The Public Key Infrastructure

Public key cryptography basically facilitates two different kinds of operations:

Encrypt/decrypt: Data encrypted with the private key of the sender (or the public key of the recipient) is decrypted by the recipient with the corresponding key half.

Sign/verify: A hash code computed from the data is encrypted by the sender with his private key and decrypted (and thereby verified) by the recipient with the corresponding public key.

Encryption provides message confidentiality when the data is encrypted with the public key of the recipient and authentication when the data is encrypted with the private key of the sender. Signing provides authentication of the signer and message integrity. These operations are not mutually exclusive. (Stallings 1995) offers a good introduction to these topics.

The collection of algorithms deployed in connection with the public key operations described above is called a public key cryptosystem. Examples for public key algorithms are *Diffie-Hellman* or *RSA*, common one-way hash functions are *MD5* or *SHA-1* and an example for a symmetrical cipher suitable for the encryption of large amounts of data is *DES*. For an excellent description of these algorithms see (Schneier 1996).

All public key operations require the existence of trusted public keys. A public key infrastructure provides mechanisms for generating, verifying, distributing and revoking public keys in a global environment.

3.2 Certificates and Management of Trust

A *certificate*, as defined in the ITU-T recommendation X.509 (CCITT 1989), binds an entity's distinguished name and some optional attributes to a public key with a digital signature. The certificate also contains the distinguished name and the signature (with an signature identifier) of the certificate issuer, an issuer-specific serial number and a validity period. A *Certification Authority (CA)* is trusted to verify a user's identity and issue certificates in accordance with its published and well-known policy. The most common kind of certificates are *identity certificates*.

In addition, a CA can also certify other CAs, transferring by this means trust to the other authority's signature. Consequently, certificates issued by the trusted CA will be accepted by any user accepting certificates from the signing CA. This delegation of trust is referred to as creating a *chain of trust*, the overall process of setting up and maintaining these chains of trust is called *trust management*.

In virtually all existing implementations, the certification mechanism is based on the ITU-T X.509 standards (CCITT 1989), which were developed as part of the X.500 framework. It is important to note that – while many aspects of the X.500 standards were never realized – the X.509 certificate format is now in widespread use for authenticating persons and services on the Internet.

By creating chains of trust, different trust models can be realized (see Figure 2). The trust models discussed in this paper have some underlying basic definitions in common. See (Maley 1996) for a detailed discussion of the distributed and centralized trust model.

- A trust model defines the chain of trust between trust points, which are representing either persons (users) or CAs. A trust point representing a person has its identity certified by at least one CA.

- A set of CAs and the persons certified by these CAs are referred to as a *domain*. Trust relationships of different quality are established between the CAs by direct certification or by chains of trust.
- A CA is trusted to certify persons identities or other CAs, which can be in a hierarchy with the certifying CA (superior or subordinate) or in a different hierarchy (cross certification).
- Every trust point has a most trusted CA, whose public key was typically obtained by out-of-band mechanisms. Frequently the most trusted CA issues the personal certificates. In Figure 2 the most trusted CA is shown with a shaded background.
- The user's CA determines the circumstances under which certificates issued by another CA may be trusted.
- Once a trust relationship with a foreign CA is established, certificates issued by the foreign CA can be accepted without re-establishing the relationship every time, provided that all of the certificates included in the chain are still valid.

It is important to note that the relationships described in the following trust models are concerned exclusively with the *identity* of a person. In developing the X.509 standard (CCITT 1989), the intention was to provide a global authentication framework — to bind a person's public key to his name.

We will focus here first on the simplest model (a flat model with a single CA) and on a highly distributed "web of trust" which is proposed by PGP. Then we will present the distributed and the centralized trust model, which are both based on the ANSI X9.75 (Ansi 1995) document.

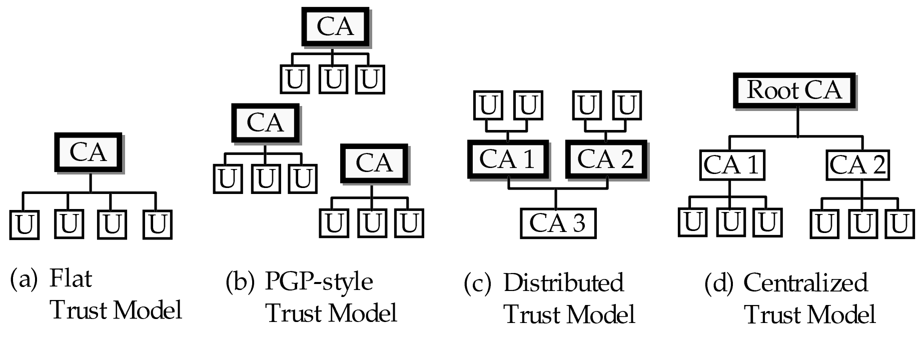

CA
U
U
U
U
(a) Flat Trust Model
CA
U
U
U
CA
U
U
U
CA
U
U
U
(b) PGP-style Trust Model
U
U
U
U
CA 1
CA 2
CA 3
(c) Distributed Trust Model
Root CA
CA 1
CA 2
U
U
U
U
U
U
(d) Centralized Trust Model

Figure 2 Different Trust Models.

Flat Trust Model The flat trust model (Figure 2 a) represents the trivial case of a single CA which certifies all users and services. While this might be sufficient in a completely secluded environment, this structure has to evolve towards one of the other models as soon as communication with externally certified parties is desired or internal services are to be made publicly accessible.

PGP-Style Trust Model The PGP-style trust model (Figure 2, b) is based on the assumption that every user is his own certification authority. Accordingly, it is up to the user to decide who is trusted and to define his own chain of trust by accepting other people as CA. The resulting structure is often referred to as a "web of trust" and is typical for PGP's decentralized trust management. PGP signatures and encryption methods (Zimmermann 1995) are widely used for electronic mail, but have currently no importance for other Internet services.

Distributed Trust Model The distributed trust model (Figure 2, c) is based on decentralized CAs where the CA issuing a personal certificate is the most trusted CA. These personal CAs operate in an independent manner, each one having its own requirements for establishing identities and distributing a trusted copy of the CA's public key. In addition to the personal CAs there are as well higher level CAs (such as CA3 in Figure 2, c) purely for key management and cross certification purposes. If a public key of a personal CA becomes compromised, the damage is restricted, only the personal certificates of the compromised CA have to be re-issued. A distributed trust model might be chosen for example by two relatively autonomous units of an organization in order to make certificates mutually acceptable.

Centralized Trust Model In contrary to the distributed trust model, the centralized trust model uses one central CA (the root CA) as the most trusted authority of the whole domain. The public key of the root CA is not signed by another CA and is the most trusted public key of the domain. Consequently, every user must be in possession of a trusted copy of the certificate of the root CA. A valid certification path can only be set up, if a trust relationship exists between the root CA and all of the parties (CAs and users). If the certificate of the root CA is compromised, all the users are affected in the recovery process. The centralized trust model implies the existence of a very powerful root CA which makes this model suited for closed hierarchically structured organizations.

3.3 Recent Developments in Certification Technologies

X.509 is a rather old specification which leaves much room for the interpretation of its fields. Unfortunately, due to this freedom, different X.509 implementations do not always interoperate. Currently there are two different efforts extending the functionality of certificates from a means of authentification towards a more general instrument:

1. *X.509 extension fields:* The latest version of the X.509 standard (*X.509v3*) allows the inclusion of user defined data, so called *extensions* in certificates. This extends the original idea of certifying a user's identity towards a means to bind arbitrary attributes to a public key. The *PKIX* working group was chartered by the IETF in order to standardize the use of extensions and thereby ensure the interoperability of different implementations (Housley *et al*. 1996).

   An important application of the *X.509v3* extension fields can be found in the standards for credit card based electronic transactions *SET* (Mastercard 1996), where data (such as a merchant identifier) is associated with a public key. The purpose of the X.509 certificate in *SET* is not only the identification of users, but also an authorization to perform certain transactions.

2. *Alternative certificate formats:* Currently at least two different concepts are under development to define a new PKI from scratch – *SPKI* (Ellison *et al*. 1996) and *SDSI* (Rivest *et al*. 1996). Both proposals regard the domain of access control as a central issue and reject further use of the X.509 standards, suggesting new encoding schemes and new key acquisition mechanisms.

4 PROTOTYPING A SECURITY INFRASTRUCTURE

In this section, we first give a brief overview on the functionality of the development environment Wafe and on the extensible Web browser Cineast which is implemented using Wafe. After this introduction we demonstrate how different implementations of a security architecture can be tested, evaluated, and extended in this environment.

4.1 Basic Components for Security Enhanced Web Clients

Wafe (Neumann *et al*. 1993) is a public available prototyping environment built upon the Tcl language (Ousterhout 1990) and was originally created for GUI-based applications in the X Window environment. Wafe is in some respects similar to the Tcl/Tk package (Ousterhout 1991) but, instead of providing its own look and feel, integrates a variety of different GUI components primarily based on MIT's generic X Toolkit. Several components are completely unrelated with GUI. Wafe provides a C code generator to ease the integration of new components.

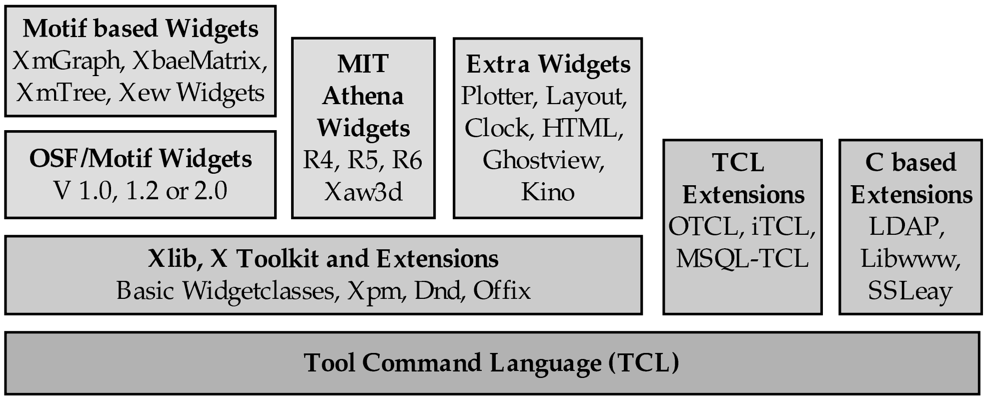

Motif based Widgets
XmGraph, XbaeMatrix, XmTree, Xew Widgets
MIT Athena Widgets
R4, R5, R6 Xaw3d
Extra Widgets
Plotter, Layout, Clock, HTML, Ghostview, Kino
OSF/Motif Widgets
V 1.0, 1.2 or 2.0
TCL Extensions
OTCL, iTCL, MSQL-TCL
C based Extensions
LDAP, Libwww, SSLeay
Xlib, X Toolkit and Extensions
Basic Widgetclasses, Xpm, Dnd, Offix
Tool Command Language (TCL)

Figure 3 Components of Wafe (version 1.0.17).

The Wafe system follows the principle that all basic functionality (Figure 3) is implemented completely in C, whereas Tcl merely functions as a “glue language” (similar to a 4th generation language) for controlling and configuring these components. The aspect of a glue language was emphasized by Ousterhout especially in the early Tcl papers (Ousterhout 1990).

The Wafe package includes several applications, most important for this paper is the W3 browser *Cineast* which is a freely available, extensible web client intending to provide an environment for prototyping new client side Internet technologies. The complete infrastructure for the browser is available in Wafe. The user interface of Cineast uses OSF/Motif and a self developed widget for HTML parsing and rendering. For general networking functionality the Cineast browser uses *libwww*, the reference library of the W3 Consortium (Frystyk-Nielsen 1996). Libwww provides a high level event-based interface to asynchronous network programming. It supports several protocols (such as HTTP/1.1 and FTP), MIME-parsing and incorporates many basic Web programming mechanisms such as Proxy and Cache handling, format management etc. Cineast obtains its cryptographic support from SSLeay (Young *et al*. 1997) and access to directory services from LDAP (Yeong *et al*. 1995). These components are finally mixed together using the object oriented Tcl variant OTcl (Wetherall *et al*. 1995). It should be noted that the source code size of the browser (about 2400 lines of code in OTcl) is about one hundredth of the involved code size counting only the core components. This indicates that Wafe and Cineast provide a powerful prototyping environment for Web based applications. The Cineast browser was presented first at the WWW6 conference in a poster session (Köppen *et al*. 1997).

4.2 Security Support

In the Cineast browser we use SSLeay (Young *et al*. 1997) for cryptographic support. SSLeay is a free implementation of SSL (currently SSL version 2) supporting *DES*, *RSA*, *RC4* and *IDEA* as encryption algorithms and *MD2*, *MD5*, *SHA-1* for message hashing. SSLeay supports *X.509v3* certificates and *PKCS#10* certificate requests. SSLeay's functionality is accessible from Wafe by two means:

- Via a *programmatic interface*: Commands to create and manipulate cryptographic keys, certificate requests and certificates are available at the TCL-level. These commands are shown in Table 1. The *create* operations create objects of the respective column, the *input* functions read files in the PEM format and decrypt it if necessary (using a provided key). The *output* functions transform the internal objects either into a PEM encoded representation (optionally encrypted) or they produce a human readable text. Furthermore there are operations to extract certain attributes from an object and to perform certificate management.

	RSA	*X509 Request*	*X509 Certificate*
Create:	w3RSA_new	w3X509_REQ_new	
Input:	w3RSA_decode	w3X509_REQ_decode	w3X509_decode
Output:	w3RSA_pem	w3X509_REQ_pem	w3X509_pem
	w3RSA_text	w3X509_REQ_text	w3X509_text
Sign:		w3X509_REQ_sign	
Verify:			w3X509_verify
GetAttr:		w3X509_REQ_getPubkey	w3X509_get
Special:	w3RSA_spkac		w3X509_loadVerifyLocations
			w3X509_addCertFile

Others: w3Text_sign, w3Text_verify, w3CertCtx_new, w3CertCtx_free

Table 1 SSLeay-Functions currently supported in the Wafe environment

- Through an *additional protocol for libwww*: *HTTPS* (HTTP over SSL) been added as an additional protocol to libwww. Secure requests for documents can be configured and started exactly like HTTP/1.1 requests. Each individual request can be configured from the Tcl level with different security relevant information such as a client certificate or a certificate context for validation. This facilitates the deployment of multiple *certificate validation policies*, which can have different requirements in terms of accepted certificate authorities, cross certification depth or other parameters of a specific trust model. In principle, this certificate validation logic can be stored in a centralized repository and retrieved by client applications using the LDAP protocol.

The following sections discuss certificate acquisition, certificate verification and certificate management in the W3 context.

(a) Certificate Acquisition

In order to obtain a certificate Cineast follows the Public Key Cryptography Standard number 10 (*PKCS#10*) as defined by RSA Laboratories (RSA 1993). In this standard a certificate request contains the distinguished name (DN), a public key and optionally a set of attributes provided by the requester. The requester signs the request with his private key. In the W3 context the primary means to prompt a user for data are HTML forms. Therefore we use a form to collect the organizational and personal data as well as the key size from the user.

When the *PKCS#10* request is generated a *RSA* key pair has to be generated – preferably on the client side. The Cineast browser uses the event type `onSubmit` to call the method `PKCS#10` to post-process the input fields and to transform it into a *PKCS#10* certificate request. This request is submitted to a CGI script on the server side, where it is placed into a request repository of the CA (see Figure 4).

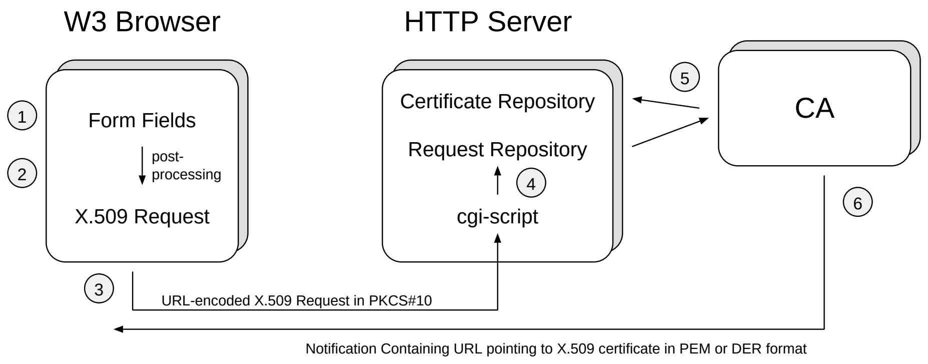


Figure 4 Process of Certificate Acquisition.

When the CA issues the certificate it retrieves the request from the repository, processes it and places the generated *X.509v3* certificate into the certificate repository. Then it either generates a URL pointing to the certificate or, alternatively, sends the certificate to the user via email. When the certificate is transmitted the MIME content type is set to `application/x-x509-user-cert`. There is no encryption necessary since the certificates contain only public data. The format manager of libwww allows to register a special handler for this content type, which will save the obtained certificate into the user's personal certificate repository. When a W3 server requires client authentication in the SSL handshake the browser can present a list of personal certificates to the user who can choose the appropriate one.

Our approach for generating keys differs from Netscape's Navigator and Communicator (Netscape 1996) where a special, non-standard HTML form input field named `KEYGEN` is introduced. This tag is "magically" displayed in the form as an option menu for choosing the key size. When the form is submitted this tag causes the generation of the key pair and the so-called SPKAC (signed public key and challenge, DER and base64 encoded) field is transmitted together with the DN information to the server. The SPKAC format was introduced by Netscape. Compared with the approach used in Navigator our approach has the following advantages:

1. Cineast does not introduce any new formats (such as SPKAC).
2. Cineast does do not introduce any "magical" HTML form input fields.
3. The implementation of the Navigator defines the entries of the option menu. It is not clear how these automatically generated entries can be provided in different languages.
4. Cineast does not require any input method (such as an option menu or a text field), the order of the entries is arbitrary.

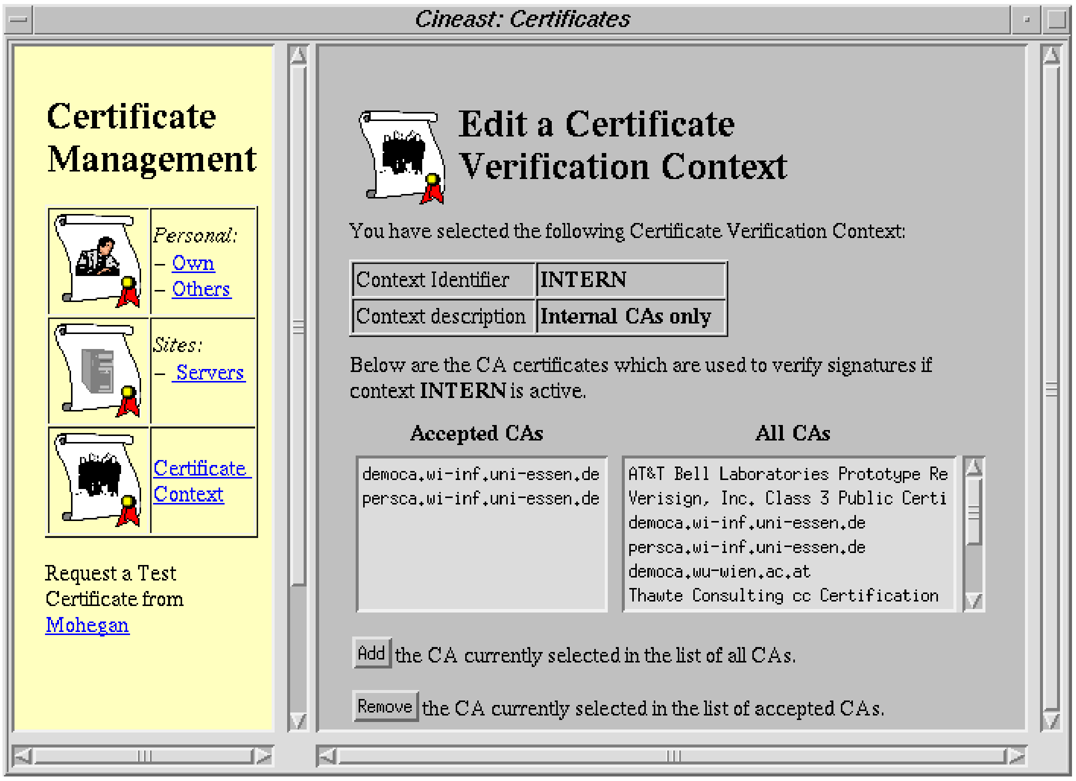
5. Cineast keeps the PEM-encoded certificates in a readable directory, the certificates can be used for other applications as well. The Navigator uses non-documented format.

However, Cineast requires (like Netscape Navigator) certain names for the input fields and it makes use of a more general method which is not restricted to a single application.

Microsoft's Internet Explorer (IE4) seems to offer a functionality similar to Netscape Navigator by means of a "magical" hidden form input field; currently, Microsoft does not publish documentation about this.

(b) Certificate Verification

In addition to the personal certificates the browser maintains a repository of CA certificates. CA certificates can be added similar to user certificates by using the content-type `application/x-x509-ca-cert`. These CA certificates are used to verify the certificates of other parties. To introduce a higher degree of flexibility, SSLeay allows to create multiple certificate contexts, with possibly non-disjoint sets of trusted CA certificates. A certificate context specifies which CAs are trusted in a particular situation. Examples for such context specific trust management might be in interactions with a bank, a cooperation project with other companies, some confidential research projects or private communications. Figure 5 displays



You have selected the following Certificate Verification Context:

Context Identifier	INTERN
Context description	Internal CAs only

Below are the CA certificates which are used to verify signatures if context INTERN is active.

Accepted CAs:
```
democa.wi-inf.uni-essen.de
persca.wi-inf.uni-essen.de
```

All CAs:
```
AT&T Bell Laboratories Prototype Re
Verisign, Inc. Class 3 Public Certi
democa.wi-inf.uni-essen.de
persca.wi-inf.uni-essen.de
democa.wu-wien.ac.at
Thawte Consulting cc Certification
```

Add the CA currently selected in the list of all CAs.

Remove the CA currently selected in the list of accepted CAs.

Figure 5 Certificate Management in the Cineast Browser.

the certificate management window of the Cineast browser. The right part of the window allows to edit the certification context named "*INTERN*" where CAs can be added or removed from the list of trusted CAs.

The Wafe example code below shows the technical realization of the basic tasks in the Wafe environment. A new certificate context is created, a CA is added into this context and a test certificate is verified in this context.

```
set cert [loadFile ~/wafe/src/cineast/test.cert]
set ctx  [w3CertCtx_new]
w3X509_addCertFile ~/.cineast/ssl/certs/vsign1.pem pem $ctx
if [string match 1 [w3X509_verify [w3X509_decode $cert] ""] $ctx] {
  puts "verification of certificate in context $ctx FAILED"
}
```

At the time of this writing all the basic mechanisms to support the certificate contexts are implemented in the Wafe environment. For example, every new `https` request can be configured with a certificate context through a request attribute. What's currently missing from the graphical user interface is some easy and intuitive way to specify the context to be used. We are planning to allow the user to set the certificate context per (named) browser instance.

5 CONCLUSION AND SUMMARY

This paper defines a framework containing the core services needed for the development of a scalable intranet architecture. We concentrated from the architectural point of view on the authentication service that plays the central role in this framework. The technologies for domain management, communication and directory services are today available and well understood, an authorization service is in our application context still in its early days. We tried to emphasize that a prototyping environment is needed and presented our approach using Wafe where only very little code was needed to implement the Cineast browser. We demonstrated that in some respects our browser compares favorably with Netscape's Navigator and Communicator. We believe that our environment is a productive platform for extending security management, to experiment or to develop new security components (e.g. with regard to access control or policy management) and to build novel Web applications.

REFERENCES

American National Standards Institute (1995) Accredited Standards Committee X9 Working Draft: American National Standard X9.57 *Certificate Management*, American Bankers Association.

Benloh, J., Lampson, B., Simon, D., Spies, T. and Yee, B. (1995) *The Private Communication Technology Protocol*, Internet Draft (Work in progress).

CCITT (1989) Recommendation X.509, *The Directory–Authentication Framework*. Blue Book – Melbourne 1988, Fascicle VIII.8: Data Communication networks: Directory, International Telecommunications Union, Geneva, Switzerland.

Ellison, C., Frantz, B. and Thomas, B. (1996) *Simple Public Key Certificate*, Internet Draft (Work in progress).

Freier, A., Karlton, P. and Kocher, P. (1996) *The SSL Protocol Version 3.0*, Internet Draft (Work in progress).

Frystyk-Nielsen, H. (1996) *Libwww - the W3C Reference Library*, `http://www.w3.org/pub/WWW/Library`, W3 Consortium.

Frystyk-Nielsen, H. (1996) *W3C Reference Library Position Statement*, `http://www.w3.org/pub/WWW/Library/Activity.html`, W3 Consortium.

Housley, R., Ford, W., Polk, W. and Solo, D. (1996) *Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile*, Internet Draft (Work in progress).

E. Köppen, G. Neumann, S. Nusser (1997) *Cineast – An extensible Web Browser*, Poster presentation at the sixth international world wide web conference, Santa Clara, CA, USA. `http://www6conf.slac.stanford.edu/`.

Lipp, P. and Hassler, V. (1996) *Security Concepts for the WWW*, in: P. Horster (ed), *Communications and Multimedia Security II*, Chapmann and Hall, London.

Maley J. (1996) *Enterprise Security Infrastructure*, in: Proceedings of the *Fifth Workshops on Enabling Technologies: Infrastructure for Collaborating Enterprises*, Stanford, CA.

Mastercard (1996) *SET: Secure Electronic Transactions*, `http://www.mastercard.com/set/`, Draft 8/7/96.

Netscape Inc. (1996) *Netscape Certificate Specifications*, `http://home.netscape.com/eng/security/certs.html`, Draft by Jeff Weinstein.

Neumann, G. and Nusser, S. (1993) *Wafe – An X Toolkit Based Frontend for Application Programs in Various Programming Languages*, USENIX Winter 1993 Technical Conference, San Diego, CA.

Nye, A. and O'Reilly, T. (1990) *X Toolkit Intrinsics Programming Manual*, O'Reilly and Associates Inc., USA.

Oppliger, R. (1997) *Internet Security: Firewalls and Beyond*, to be published in CACM.

Ousterhout, J.K. (1990) *Tcl: An embeddable Command Language*, Proceedings of the 1990 Winter USENIX Conference.

Ousterhout, J.K. (1991) *An X11 Toolkit Based on the Tcl Language*, Proceedings of the 1991 Winter USENIX Conference.

Rivest R. and Lampson, B. (1996) *SDSI — A Simple Distributed Security Infrastructure*, in Proceedings of *DIMACS Workshop on Trust Management in Networks*, South Plainfield, NJ, USA.

RSA Laboratories (1993) *PKCS#10: Certification Request Syntax Standard*, Version 1.0.

Sandhu, R., Coyne, E., Feinstein, H. and Youman, C. (1996) *Role based access control models*, in: *IEEE Computer*.

Schneier, B. (1996) *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, New York.

Stallings, W. (1995) *Network and Internetwork Security*, Prentice Hall, Englewood Cliffs.

Wetherall, D. and Lindblad, C.J. (1995) *Extending Tcl for Dynamic Object-Oriented Programming*, Proceedings of the Tcl/Tk Workshop '95, Toronto.

Xcert Software Inc. (1997) *The Xcert Software Sentry CA*, product description `http://www.xcert.com/software/sentry/ca/`.

Yeong, W., Howes, T. and Kille, S. (1995) *Lightweight Directory Access Protocol*, RFC 1777.

Yialelis, N., Lupu, E. and Sloman, M. (1996) *Role Based Security for Distributed Object Systems*, in: Proceedings of the *Fifth Workshops on Enabling Technologies: Infrastructure for Collaborating Enterprises*, Stanford, CA.

Young, E. and Hudson, T. (1997) *SSLeay and SSLapps FAQ*, `http://www.psy.oz.au/~ftp/Crypto/`.

Zimmermann, P. (1995) *The Official PGP User's Guide*, MIT Press, Cambridge, Massachusetts.