Draft Technical Standard

DCE 1.2.3 Public Key Certificate Login (Draft 0.8 for Company Review)

The Open Group

Draft Technical Standard

DCE 1.2.3 Public Key Certificate Login (Draft 0.8 for Company Review)

ISBN: 1-85912-216-7
Document Number: P801

Published in the U.K. by The Open Group, August 1998.

Any comments relating to the material contained in this document may be submitted to:

The Open Group
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

OGSpecs@opengroup.org

Contents

List of Figures

List of Tables

Preface

The Open Group

The Open Group is the leading vendor-neutral, international consortium for buyers and suppliers of technology. Its mission is to cause the development of a viable global information infrastructure that is ubiquitous, trusted, reliable, and as easy-to-use as the telephone. The essential functionality embedded in this infrastructure is what we term the *IT DialTone*. The Open Group creates an environment where all elements involved in technology development can cooperate to deliver less costly and more flexible IT solutions.

Formed in 1996 by the merger of the X/Open Company Ltd. (founded in 1984) and the Open Software Foundation (founded in 1988), The Open Group is supported by most of the world's largest user organizations, information systems vendors, and software suppliers. By combining the strengths of open systems specifications and a proven branding scheme with collaborative technology development and advanced research, The Open Group is well positioned to meet its new mission, as well as to assist user organizations, vendors, and suppliers in the development and implementation of products supporting the adoption and proliferation of systems which conform to standard specifications.

With more than 200 member companies, The Open Group helps the IT industry to advance technologically while managing the change caused by innovation. It does this by:

- Consolidating, prioritizing, and communicating customer requirements to vendors
- Conducting research and development with industry, academia, and government agencies to deliver innovation and economy through projects associated with its Research Institute
- Managing cost-effective development efforts that accelerate consistent multi-vendor deployment of technology in response to customer requirements
- Adopting, integrating, and publishing industry standard specifications that provide an essential set of blueprints for building open information systems and integrating new technology as it becomes available
- Licensing and promoting the Open Brand, represented by the ‘‘X’’ Device, that designates vendor products which conform to Open Group Product Standards
- Promoting the benefits of the IT DialTone to customers, vendors, and the public

The Open Group operates in all phases of the open systems technology lifecycle including innovation, market adoption, product development, and proliferation. Presently, it focuses on seven strategic areas: open systems application platform development, architecture, distributed systems management, interoperability, distributed computing environment, security, and the information superhighway. The Open Group is also responsible for the management of the UNIX trademark on behalf of the industry.

Development of Product Standards

This process includes the identification of requirements for open systems and, now, the IT DialTone, development of Technical Standards (formerly CAE and Preliminary Specifications) through an industry consensus review and adoption procedure (in parallel with formal standards work), and the development of tests and conformance criteria.

This leads to the preparation of a Product Standard which is the name used for the documentation that records the conformance requirements (and other information) to which a vendor may register a product.

The ''X'' Device is used by vendors to demonstrate that their products conform to the relevant Product Standard. By use of the Open Brand they guarantee, through the Open Brand Trade Mark License Agreement (TMLA), to maintain their products in conformance with the Product Standard so that the product works, will continue to work, and that any problems will be fixed by the vendor.

Open Group Publications

The Open Group publishes a wide range of technical documentation, the main part of which is focused on development of Technical Standards and product documentation, but which also includes Guides, Snapshots, Technical Studies, Branding and Testing documentation, industry surveys, and business titles.

There are several types of specification:

- *Technical Standards* (formerly *CAE Specifications*)

  The Open Group Technical Standards form the basis for our Product Standards. These Standards are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement a Technical Standard can enjoy the benefits of a single, widely supported industry standard. Where appropriate, they can demonstrate product compliance through the Open Brand. Technical Standards are published as soon as they are developed, so enabling vendors to proceed with development of conformant products without delay.

- *CAE Specifications*

  CAE Specifications and Developers' Specifications published prior to January 1998 have the same status as Technical Standards (see above).

- *Preliminary Specifications*

  Preliminary Specifications have usually addressed an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations. They are published for the purpose of validation through implementation of products. A Preliminary Specification is as stable as can be achieved, through applying The Open Group's rigorous development and review procedures.

  Preliminary Specifications are analogous to the *trial-use* standards issued by formal standards organizations, and developers are encouraged to develop products on the basis of them. However, experience through implementation work may result in significant (possibly upwardly incompatible) changes before its progression to becoming a Technical Standard. While the intent is to progress Preliminary Specifications to corresponding Technical Standards, the ability to do so depends on consensus among Open Group members.

- *Consortium and Technology Specifications*

  The Open Group publishes specifications on behalf of industry consortia. For example, it publishes the NMF SPIRIT procurement specifications on behalf of the Network Management Forum. It also publishes Technology Specifications relating to OSF/1, DCE, OSF/Motif, and CDE.

  Technology Specifications (formerly AES Specifications) are often candidates for consensus review, and may be adopted as Technical Standards, in which case the relevant Technology Specification is superseded by a Technical Standard.

In addition, The Open Group publishes:

- *Product Documentation*

  This includes product documentation—programmer's guides, user manuals, and so on—relating to the Pre-structured Technology Projects (PSTs), such as DCE and CDE. It also includes the Single UNIX Documentation, designed for use as common product documentation for the whole industry.

- *Guides*

  These provide information that is useful in the evaluation, procurement, development, or management of open systems, particularly those that relate to the Technical Standards or Preliminary Specifications. The Open Group Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming conformance to a Product Standard.

- *Technical Studies*

  Technical Studies present results of analyses performed on subjects of interest in areas relevant to The Open Group's Technical Program. They are intended to communicate the findings to the outside world so as to stimulate discussion and activity in other bodies and the industry in general.

Versions and Issues of Specifications

As with all *live* documents, Technical Standards and Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards compatible and those which are not:

- A new *Version* indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it *replaces* the previous publication.
- A new *Issue* indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

Corrigenda

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published on the World-Wide Web at **http://www.opengroup.org/corrigenda**.

Ordering Information

Full catalogue and ordering information on all Open Group publications is available on the World-Wide Web at **http://www.opengroup.org/pubs**.

This Document

The *DCE 1.2.3 Public Key Certificate Login (Functional Specification)* is a Draft Technical Standard from The Open Group, August 1998.

Synopsis

DCE RFC 68.4 (on which this specification is wholly based) is a follow-on replacement to DCE RFC 68.3. It provides the following functional enhancements.

- Use of X.509v3 Public Key Certificates for DCE client authentication to the KDC.
- Use of Cryptographic Message Standard (CMS) for digitally signing and enveloping parts of Kerberos authentication flows.
- Isolation of the details of the Kerberos Public Key Initial Authentication ASN.1 structures, public key infrastructures, and CMS functionality under a pluggable (DLL) component, the *pkinit_cms_** functions.
- Support for smart cards and delivery of a software smart card in the reference implementation of *pkinit_cms_**.
- ‘‘PKI-neutral’’ implementation that supports multiple PKIs.
- An Identity Mapping Service (IDMS).
- A new registered Kerberos Authorization Data type for sealing a client’s original certificate based identity information in the DCE TGT and subsequent tickets.
- An enhanced Audit Service that transparently extracts the client’s certificate-based identity, if present, from an RPC binding handle and places it in the audit log records.
- A new API, *sec_id_get_certid*( ), that an application can use to extract the certificate-based identity information, if present, from an RPC binding handle.

The functionality defined in this Specification supports a security model that moves towards the use of PKI (i.e., X.509v3 public key certificates) for authentication, and the use of DCE for authorization. This model strongly suggests the desirability of moving long-term user information out of the DCE Registry and into an LDAP directory, therefore consolidating the (logical) storage and access of PKI and DCE information.

Summary of Changes

Draft 0.4

Presented 30Apr1998 at The Open Group’s Members Meeting in San Diego, California.

Draft 0.5

1. Minor spelling and grammar updates throughout the document.
2. Added Identity Mapping Service (IDMS) and Credential Acquisition Service (CAS) to Synopsis.
3. Added Rgy-to-LDAP Utility placeholder.
4. Removed ‘‘Notes to RFC reviewers’’ on Page 2 of Draft 0.4.
5. Added ‘‘Acknowledgements’’ section toward end of document.

6. Removed ‘‘fall-back to shared-secret key login’’ support; added special X509USER DCE principal and DCEX509 environment variable.

Draft 0.6

1. Removed Credential Acquisition Service (CAS); the existing DCE Privilege Service (PS) is unchanged.
2. Removed ‘‘Rgy-to-LDAP Utility.’’
3. To maintain accountability, added use of OSF-DCE-PKI-CERTID Authorization Data within the TGT to carry a client’s original certificate-based identity, along with the IDMS-provided mapped DCE principal name. The DCE Audit Service is enhanced to store OSF-DCE-PKI- CERTID information in audit records. A new API, *sec_id_get_certid*( ) is defined to enable applications to access the OSF-DCE-PKI-CERTID information.
4. Reinstated ‘‘fall-back to shared-secret key login’’ support; eliminated special X509USER DCE principal. The DCEX509 environment variable has been renamed to DCE_PKI_INI.

Draft 0.7

1. Updated references to IETF RFC 1779 with RFC 2253, per [DRAFT-PKINIT].
2. Added information regarding new Kerberos error types introduced by [DRAFT-PKINIT].
3. Changed definition of OSF-DCE-PKI-CERTID to use the [DRAFT-CMS] CertUid ASN.1 definition.
4. Placed IDMS IDL in section 6.4.1.
5. Added detail for the sec_id_get_certid() API. Added gssdce_extract_certid_from_cred() for the GSSAPI equivalent.
6. Added detail regarding the use of the DCE_PKI_INI environment variable with the dce_login command and sec_login_* APIs.
7. Removed sections 9 and 10; carry-overs from [RFC 68.3].
8. Removed sections 9.4 and 9.5 (old 11.4 and 11.5); PKI administration is now handled by the PKI.
9. Updated sections 10.1.2 (old: 12.1.2) and removed section 10.2 (old: 12.2).
10. Removed section 12 (old:14); information provided elsewhere in the RFC.

Draft 0.8

Draft 0.8 has minor changes over draft 0.7. For example, some ’Notes’ have been changed into footnotes. Draft 0.8 is this draft, and is presented for Company Review, August 1998.

Trademarks

Motif®, OSF/1®, UNIX®, and the ‘‘X Device’’® are registered trademarks and IT DialTone™ and The Open Group™ are trademarks of The Open Group in the U.S. and other countries.

Acknowledgements

This standardization effort is supported and funded by DASCOM, Gradient, Hewlett-Packard, and IBM.

The authors wish to express their appreciation and thanks to the following individuals for their contributions to this specification: Tim Moses of Entrust, Gerry Gebel of Chase Manhattan Bank, Brian Breton and Kevin Farrington, of Gradient and Jonathan Chinitz of Intellisoft. The authors also wish to express their appreciation to The Open Group, SIMC, and their respective companies for their continued backing of this effort.

Authors' Addresses:

Frank Siebenlist
DASCOM, Inc.
3004 Mission Street
Santa Cruz, CA 95060
USA

Internet e-mail: frank@dascom.com
Telephone: +1-408-460-3600

Dave Hemsath
IBM Corporation
11400 Burnet Road
Austin, TX 78758
USA

Internet e-mail: hemsath@us.ibm.com
Telephone: +1-512-838-3618

Referenced Documents

The following documents are referenced in this specification:

CDSA

[CDSA] The Open Group, CAE Specification, ‘‘Common Security: CDSA and CSSM’’, December 1997, ISBN 1- 85912-194-2, Document Number C707.

DRAFT-CMS

[DRAFT-CMS] IETF, ‘‘draft-ietf-smime-cms-06.txt’’, by R. Housley, June 1998.

DRAFT-PKINIT

[DRAFT-PKINIT] IETF, ‘‘draft-ietf-cat-kerberos-pk-init- 07.txt’’, by B. Tung, C. Neuman, J. Wray, A. Medvinsky, M. Hur and J, Trostle, <Month> 1998.

DRAFT-SFL

[DRAFT-SFL] ‘‘Draft S/MIME Freeware Library-Application Programming Interface Version 0.3’’, 27 March 1998, J.G. Van Dyke & Associates, Inc.

IETF 1510

[IETF 1510] IETF, ‘‘RFC 1510: The Kerberos Network Authentication Service (V5)’’, by J. Kohl and C. Neuman, September 1993.

IETF 1779

[IETF 1779] IETF, ‘‘RFC 1779: A String Representation of Distinguished Names’’, by S. Kille, March 1995.

IETF 2315

[IETF 2315] IETF, ‘‘RFC 2315: PKCS#7: Cryptographic Message Syntax Version 1.5,’’ by B. Kaliski, March 1998.

ITU AM1

[ITU AM1] ITU, ‘‘PDAM 1 to ITU-T X.509 (1993) | ISO/IEC 9594-8:1995, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework’’, ISO/IEC JTC 1/SC 21 N 9214, December 1994.

ITU AM4

[ITU AM4] ITU, ‘‘PDAM 4 to ITU-T X.511 (1993) | ISO/IEC 9594-3:1995, Information Technology - Open Systems Interconnections - The Directory: Abstract Service Definition’’, ISO/IEC JTC 1/SC 21 N, 24 November 1995.

ITU X.208

[ITU X.208] ITU, ‘‘Recommendation X.208: Specification of abstract syntax notation one (ASN.1)’’, ISO/IEC 8824, 1987.

ITU X.209

[ITU X.209] ITU, ‘‘Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).’’, ISO/IEC 8825, 1987.

ITU X.5091

[ITU X.509] ITU, ‘‘Final Text of the 1993 Edition of ISO/IEC 9594-8/ITU-T Rec X.509, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework’’, ISO/IEC JTC 1/SC 31 N 8696, 28 June 1994.

ITU X.511

[ITU X.511] ITU, ‘‘ISO/IEC 9594-3/ITU-T Rec X.511, Information Technology - Open Systems Interconnection - The Directory: Abstract Service Definition’’, 1993 (E).

PKCS 8

[PKCS 8] RSA Laboratories, ‘‘PKCS #8: Private-Key Information Syntax Standard’’, Version 1.2, November 1, 1993.

PKCS 11

[PKCS 11] RSA Laboratories, ‘‘PKCS #11: Cryptographic Token Interface Standard’’, Version 2.01, December 22, 1997

RFC 6.0

[RFC 6.0] J. Pato, ‘‘A Generic Interface for Extended Registry Attributes’’, June 1992.

RFC 68.1

[RFC 68.1] A. Anderson, J. Wray, ‘‘DCE 1.2 Public-Key Login - Functional Specification’’, February 1995.

RFC 68.3

[RFC 68.3] A. Anderson, S. Cuti, ‘‘DCE 1.2.2 Public Key Login-Functional Specification’’, January 1997.

RFC 80.0

[RFC 80.0] J. Wray, ‘‘DCE Certification API - Functional Specification’’, January 1995.

RFC 85.0

[RFC 85.0] M. Warner, ‘‘Improved Public Key Login Protocols for DCE’’, October 1995.

RFC 86.0

[RFC 86.0] V. Samar, R. Schemers, ‘‘Unified Login with Pluggable Authentication Modules (PAM)’’, October 1995.

XSSO-PAM

[XSSO-PAM] The Open Group, Preliminary Specification, ‘‘X/Open Single Sign-on Service (XSSO) - Pluggable Authentication Modules,’’ X/Open Document Number: P702, ISBN: 1-85912-144-6

See Also

A number of publications relevant to DCE are available from the publications department at The Open Group.

DCE 1.1: Authentication and Security Services C311
DCE 1.1: Directory Services C705
DCE 1.1: Distributed File Service Specification P409
DCE 1.1: Remote Procedure Call C706
DCE 1.1: Time Services Specification C310
DCE 1.2.2 Administration Guide - Core Components F208
DCE 1.2.2 Administration Guide - Introduction F207
DCE 1.2.2 Application Development - Introduction and Style Guide F202
DCE 1.2.2 Application Development Guide - Core Components Vol 1 F203A
DCE 1.2.2 Application Development Guide - Core Components Vol 2 F203B
DCE 1.2.2 Application Development Guide - Directory Services F204
DCE 1.2.2 Application Development Reference Volume 1 F205A
DCE 1.2.2 Application Development Reference Volume 2 F205B
DCE 1.2.2 Application Development Reference Volume 3 F205C
DCE 1.2.2 DCE Command Reference F212
DCE 1.2.2 DFS Administration Guide and Reference - Volume 1 F209A
DCE 1.2.2 DFS Administration Guide and Reference - Volume 2 F209B
DCE 1.2.2 File-Access Administration Guide & Reference F216

DCE 1.2.2 File-Access FVT User’s Guide F210
DCE 1.2.2 File-Access User’s Guide F217
DCE 1.2.2 GDS Administration Guide and Reference F211
DCE 1.2.2 Introduction to OSF DCE F201
DCE 1.2.2 Problem Determination Guide - Volume 1 F213A
DCE 1.2.2 Problem Determination Guide - Volume 2 F213B
DCE 1.2.2 Release Notes F218
DCE 1.2.2 Testing Guide F215
DCE Version 1.0.3 Fourteen Volume Set T010
DCE Version 1.1 Thirteen Volume Set T110
DCE: Authentication and Security Services P315
DCE: Directory Services C312
DCE: Directory Services P314
DCE: Remote Procedure Call C309
DCE: Remote Procedure Call P312
DCE: Time Service P313
Distributed Software Administration - DCE Interoperability C430
Introduction to OSF DCE 1.1 F101P
Introduction to OSF DCE Rev 1 F001P
OSF DCE 1.0.3 Administration Guide - Extended Services F029P
OSF DCE 1.0.3 Administration Guide - Introduction F006P
OSF DCE 1.0.3 Administration Reference F030P
OSF DCE 1.0.3 AE Specification - Remote Procedure Call F012P
OSF DCE 1.0.3 Application Development Guide Volume 1 F023P
OSF DCE 1.1 - Administration Guide: Volume 1: Introduction F107P
OSF DCE 1.1 - Administration Guide: Volume 2 - Core Components F108P
OSF DCE 1.1 - Application Development Guide: Core Components F103P
OSF DCE 1.1 - Application Development Guide: Directory Services F104P
OSF DCE 1.1 - Application Development Reference: Volume 1 F105P
OSF DCE 1.1 - GDS Administration Guide: Extended Services F110P
OSF DCE 1.1 Application Development Guide - Intro & Style Guide F102P
OSF DCE 1.1 Application Development Reference Volume 2 F106P
OSF DCE 1.1 DCE Command Reference F111P
OSF DCE 1.1 DFS Administration Guide and Reference F109P
OSF DCE Administration Guide - Core 1.0.3 F027P
OSF DCE Administration Guide - Core Components F008P
OSF DCE Administration Guide - Extended Services F009P
OSF DCE Administration Reference F010P
OSF DCE Administration Reference Rel 1.0.2 F011P
OSF DCE Application Development Guide F003P
OSF DCE Application Development Reference - 1.0.3 F025P
OSF DCE Application Development Reference - Volume 1 Release 1.0 F004P
OSF DCE Users Guide and Reference F002P

Chapter 1

Introduction

This document specifies the functionality required to integrate public key mechanisms into DCE login, that is, into the initial DCE Kerberos Ticket-Granting Ticket protocol. This specification obsoletes [RFC 68.3]. Note that there has been such a high volume of change activity in the IETF relative to Public Key Infrastructure (PKI) and Kerberos that the [RFC 68.3] functionality will not be forward compatible with this Specification.[1] *Therefore, current users of DCE 1.2.2-based products with [RFC 68.3] functionality should refrain from deploying the public key-based login support.*

The goal of this effort is to allow DCE users to use an X.509v3 digital signature certificate and its associated private key rather than a shared-secret password to prove their identity to the Authentication Service (AS) of the DCE Key Distribution Center (KDC) (*a.k.a.* Key Distribution Server, KDS).

An immediate benefit is that, in the event of a compromise of the KDC, public key users do not have any identifying information exposed to the intruder. If the KDC is compromised, all user secret keys will be revealed to the intruder. This means they become worthless as a proof of identity, and therefore the cell administrator must re- issue passwords to all such users before they can be allowed to log-in to the cell. Under the design described in this Specification, public key users prove their identity by knowledge of a private key that is never known to the KDC, and therefore a compromise of the KDC cannot reveal these keys.

Another benefit is that the basic authentication flows are made more secure by virtue of public key cryptographic methods, coupled with large signature and encryption asymmetric key-pairs.

A third benefit is using DCE to improved scalability over ‘‘pure PKI’’ deployments. Consider an environment with C clients and S servers. During the course of an operational shift, each client has to connect to each server. In a pure PKI environment, assume each client connects to each server using Secure Sockets Layer Version 3 (SSLv3) with client-side certificates part of the authentication and session establishment exchange. In this scenario, there are at least $C \times S$ computationally expensive public key cryptographic operations. Now consider the same scenario with clients and servers using the PKI to authenticate to the DCE Authentication Service (AS), but then obtaining computationally efficient normal shared secret key (SSK) DCE service tickets for client-server mutual authentication and session establishment. Then there are only $C + S$ public key cryptographic operations required.

A fourth benefit is reduced DCE administration via the ability to map multiple certificate-based identities to a relatively smaller set of DCE principals. Admittedly, this is a small sleight-of-hand, with user administration shifted to the PKI. However, with the generally-accepted view of moving towards PKIs for authentication, overall user administration (e.g., enrollment) is reduced.

The authentication information and protocol are based on the PK-INIT Kerberos protocol [DRAFT-PKINIT]. The reference implementation of this Specification requires that the authenticating user's signature and encryption certificates and corresponding private keys[2] be

---

1. For example, the order of the data fields in the *pkAuthenticator* structure has changed, and the pre- authentication (PA) type values have changed for the authentication request and reply.

2. Some PKIs such as Entrust assign a pair of certificates to each user; one for signature operations and one for encryption operations. Hence, there is a private key for each certificate. Other PKIs collapse the signature and cryptographic operations into one user certificate. In the case of dual-use certificates, this Specification specifies that the encryption certificate be duplicated from the signature certificate.

stored in a smart card. This provides a standard place to look for the certificates and keys, thus avoiding several problems associated with proprietary ‘‘key ring’’ implementations. In addition to acting as a secure store for the certificates and keys, the smart card is used to perform the cryptographic operations required for certificate-based login. That is, signature generation and verification operations, and public key ‘‘wrapping’’ of symmetric cryptographic keys. The reference implementation of this Specification will provide a software implementation of a smart card that is accessed through the Common Data Security Architecture [CDSA] framework. CDSA supports smart cards that support the Public-Key Cryptographic Standard (PKCS) Number 11 [PKCS 11]. Note that the smart card support is embedded in the reference implementation’s *pkinit_cms_** DLL.

Public key certificate-based signed and encrypted (*a.k.a.* enveloped) messages that are transported in the [DRAFT- PKINIT] protocol are formatted using the Cryptographic Message Syntax[3] (CMS-see [DRAFT-CMS]). CMS is an open standard derived from PKCS Number 7 Version 1.5 (see [IETF 2315]). CMS standardization is under the charter of the IETF Secure / MIME Working Group. CMS software development kits (SDKs) are available in the public domain and multiple vendors.[4] This Specification defines a *pkinit_cms_** abstraction layer that handles all required CMS functions. The reference implementation of this Specification provides a *pkinit_cms_** based on the S / MIME Freeware Library (see [DRAFT-SFL]) and CDSA. However, implementers of this Specification may choose to offer additional or alternate implementations of *pkinit_cms_** using other CMS and cryptographic SDKs.

1.1 Changes Since Last Publication

1.1.1 Changes since [RFC 68.3]

1. The public key login protocol is one of the protocols specified in [DRAFT-PKINIT], extended with support for Cryptographic Message Syntax (CMS) formatted messages. This enhancement to [DRAFT-PKINIT] was submitted to the IETF Common Authentication Technology (CAT) Working Group at the IETF’s meeting in March 1998. The CAT WG accepted most of the proposal and is incorporating it into [DRAFT-PKINIT].

2. CMS functions are provided by the new *pkinit_cms_** function. The reference implementation of *pkinit_cms_** is built using a combination of the S / MIME Freeware Library [DRAFT-SFL] that uses the CDSA Framework for its underlying cryptographic, certificate and data services, including smart card-based services.

3. Users’ public keys are no longer stored in the DCE Registry. They are obtained from users’ X.509v3 public key certificates.

4. A secure Identity Mapping Service (IDMS) is introduced to enable flexible mapping of users’ certificate-based identities to a DCE principal. Installations have widely varied security policies regarding granting of access rights to users based on their X.509v3 public key certificates. ‘‘Identity’’ is likely to be more than just the [IETF 2253] Distinguished

---

3. As of 15 July 1998, the number 7 version of [DRAFT- PKINIT] has yet to be published by the IETF. According to the version received by the authors on 16 June 1998, the authentication request and reply messages are ‘‘CMS-like,’’ but not identical to the [DRAFT-CMS] formats. The authors have verified that it’s still possible to use CMS SDKs to create the [DRAFT-PKINIT] ASN.1 constructs. The *pkinit_cms_** DLL will have to perform more operations such as OID mapping, and structure dissassembly / reassembly to be in conformance with [DRAFT- PKINIT]. An alternative considered, but rejected for the time being would be to use private (DCE-proprietary) PA types that used ‘‘pure CMS’’ formats.

4. Any CMS SDK used to implement the *pkinit_cms_** functions should be thread-safe, and export ANSI-C bindings.

Name (DN) bound in the certificate. A DN may not be unique (although ‘‘reputable’’ Certification and Registration Authorities will seek to minimize DN collisions). The IDMS can use other factors such as the identity of the Certification Authority (CA) that issued the certificate, the certificate's serial number, certificate extensions, etc. to decide which DCE principal to assign to users. Since installations need to be able to define and implement their own mapping policies, the IDMS is provided in source form. Installations can modify the IDMS to implement their particular mapping policies. Great care must be excercised when modifying the IDMS functions since it's part of the DCE Trusted Computing Base (TCB). For example, one wouldn't want the ‘‘default’’ DCE principal to be *cell_admin*! Note that the IDMS has been reintroduced after determining that the identity mapping step is required ‘‘earlier’’ in the KRB_AS_REQ / REP Kerberos flows. This is because DCE's design assumes, and makes heavy use of DCE principal names and UUIDs. This includes DCE's Audit services which require principal UUIDs (not DNs).

5. Asymmetric key-pair generation, certificate creation, revocation, etc. are to be handled by an installation's PKI. DCE is ‘‘PKI-neutral’’ though its use of the *pkinit_cms_** function.

6. A new registered Kerberos Authorization Data type, tentatively named OSF-DCE-PKI-CERTID, is defined and registered with the owners of the Kerberos standards. This new type is used for sealing a client's original certificate based identity information in the DCE TGT and subsequent tickets.

7. The DCE Audit Service is enhanced to transparently extract a client's certificate-based identity, if present, from an RPC binding handle and place it in the audit log records. This preserves accountability back to the original user.

8. A new API, *sec_id_get_certid*( ), is defined that an application can use to extract the certificate-based identity information, if present, from an RPC binding handle. This is important for many potential applications, that require entity-based access checks. For example, an investment firm may issue X.509v3 digital certificates to its clients, but map them (via its version of the IDMS) to a relatively small number of DCE principals to access ‘‘back-end’’ services. At the same time, certain transactions such as portfolio inquires, changes to investment allocations, etc. will need to know the certificate-based identity of the client requesting the operations.

1.2 Target

This technology is provided for customers who require that their PKI-of-choice be their primary authentication technology. It also provides a higher level of security for:

(1) Initial authentication to DCE using large asymmetric key-pairs for digital signatures and encryption of session keys. This is demonstrably stronger than 56-bit DES shared secret key technology.

(2) Removal of long-term keys from the DCE Registry.

Note that the use of public key technology is only for the purpose of initial authentication to DCE. Service tickets to RPC servers, etc. continue to be obtained in the normal manner after the initial Ticket-Granting Ticket (TGT) is obtained. The support of additional cryptographic mechanisms for system and user data integrity/confidentiality will be addressed in a separate RFC. It is expected that such ‘‘pluggable crypto’’ support will be based on the CDSA Framework and may have to address Key Recovery for exportability.

1.3 Goals and Non-Goals

1.3.1 Goals

(a) Allow users to use an X.509v3 signature certificate and its associated private key rather than a shared secret to prove their identity to the DCE Key Distribution Center.

(b) Provide a standards-based mutual authentication protocol between the user and the DCE Key Distribution Center.

(c) The protocol must not require private keys to be stored in the DCE Registry or to be transmitted across the wire protected by a password-derived key.

(d) Ease recovery from a compromise of the DCE Key Distribution Center.

(e) Allow for use of public key algorithms that need not be RSA through the use of the *pkinit_cms_** component.

(f) Allow for integration with multiple PKIs by isolating PKI-specifics underneath the *pkinit_cms_**.

(g) Implement the certificate-based DCE Login in such a manner as to be fully exportable without requiring a separate export version of keys and/or cryptographic mechanisms.

(h) Improve the scalability of public key certificate- based authentication systems.

(i) Implement the new function without changes to the *dce_login* command syntax.

(j) Implement in a way that supports controlled deployment of the new functions.

1.3.2 Non-Goals

(a) An integrated login between the PKIs and DCE is not specified. At some point, implementing vendors may choose to provide PKI+DCE[+OS]-specific integrated logins or other Single Sign-On (SSO) solutions.

(b) Integrated administration between the PKIs and DCE is not specified. Further investigation is required on how to provide ‘‘administrative end points’’ for popular and prevalent management suites for providing a common and consistent management of DCE and PKIs.

(c) This function is not forward-compatible from DCE 1.2.2. This is due to the significant changes in [DRAFT- PKINIT] since the publication and implementation of [RFC 68.3].

(d) The new certificate-based login function is for users only, i.e., this support is not extended to programmatic entities using Keytab files.

Chapter 2

Requirements

The technology must support an increase to the overall security of a DCE cell. It must also represent a genuine integration of public key technology with the DCE login process. Specific business and technical requirements are listed below.

2.1 Business Requirements

(a) The new function must be available from multiple vendors and be fully interoperable in a multi-vendor DCE 1.2.3 deployment.

(b) Entrust's PKI must be supported, but, ideally, the new function should be ''PKI-neutral.''

(c) A reference implementation is required in 1998. Interoperable, multi-vendor, multi-platform products are needed no later than 1H1999. Note that the SIMC members' designated key platforms are Windows NT and Unix (AIX, HP-UX and Solaris).

(d) Full accountability must be maintained. That is, even in the likely event that many certificate holders are mapped to a common DCE principal, there must be a way for the DCE Audit Service and secure resource managers to correctly identify the original (certificate- based) identity of the user.

2.2 Technical Requirements

(a) Public key certificates and public key infrastructures are the primary method of authentication.

(b) The function must be predicated, where appropriate, on other open standards from IETF (e.g., Kerberos, PKIX and S/MIME), TOG (e.g., CDSA), IMC, W3C, etc.

(c) An installation must be able to define its own policy for mapping the identity embodied in the client's signature certificate to a DCE principal. Some installations have expressed a requirement to perform a one-to-one mapping. Others have stated a need to perform more sophisticated mappings, e.g., mapping multiple certificate- based identities to a common DCE principal.

(d) ''DCE-less'' clients, e.g., secure web browsers with client certificates, should be able to securely use DCE- based resource managers (e.g., DFS), subject to installation policy. Note that this type of proxied login has been implemented in several forms by multiple vendors. A requirement exists for a standard ''Identity Mapping Service'' for the DCE Authentication Service and proxies. A good example scenario was generated by the SIMC members and is shown in Figure 1 below. (e) Administration should be integrated and consistent between DCE and the PKI(s).

(f) The new function should be forward-compatible from previous-versions of DCE. It should support pre-1.2.3 level DCE clients (not server replicas). Note the DCE 1.2.2 exception in ''3.2. Non-Goals'' above.

(g) Smart cards should be supported for holding private signature and encryption keys. They should be usable for generating and verifying digital signatures and for digital enveloping operations. Note that there are potential export issues to be addressed.

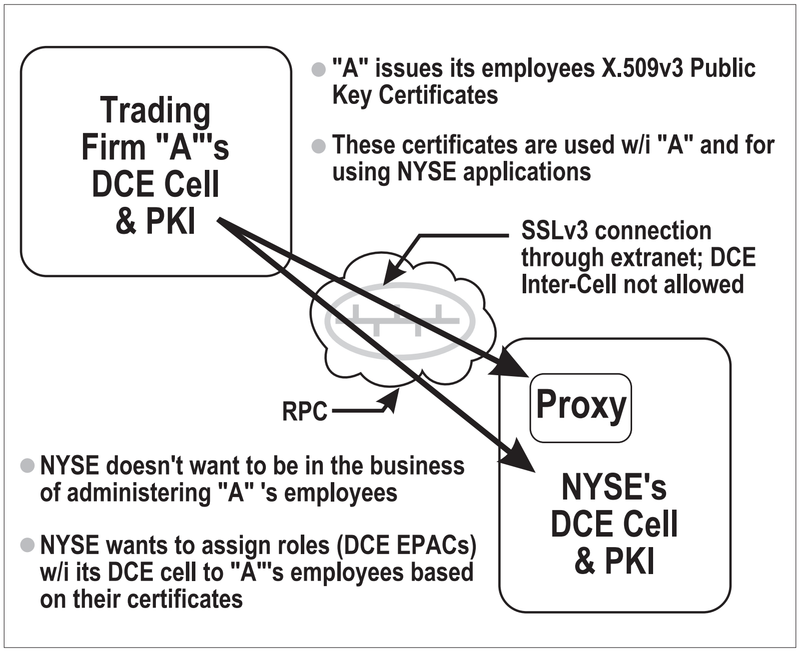

Figure 2-1 SIMC Example

Chapter 3

Functional Definition

An overview of the new functions is shown in Figure 2 below. In step (1), the DCE Login client code sends the Kerberos KRB_AS_REQ message to the DCE Authentication Service (AS), which is part of the DCE security server (secd). The request, enhanced to support the [DRAFT-PKINIT] standard, includes the client's certificates and a digitally signed authenticator. In step (2), the AS makes a Secure RPC call to the IDMS, sending it the client's already-verified signature certificate. The IDMS ''crunches'' the certificate and in step (3) sends back a mapped userid to the AS. The AS then uses this mapped userid to do its ''business-as-usual'' construction of the Ticket Granting Ticket (TGT). In constructing the TGT, the AS now also creates the OSF-DCE-PKI-CERTID Authorization Data based on the [DRAFT-CMS] Certificate-Unique-Indentifier (CertUid) construct. This new Authorization Data structure is incorporated into the TGT which is returned to the client in step (4) via the [DRAFT-PKINIT]-enhanced KRB_AS_REP message. At this point, (step (5) and beyond), the trip to the DCE Privilege Service (PS) to obtain the PTGT, etc. is the same as pre- DCE 1.2.3 implementations.

For non-DCE clients, such as secure web browsers, the IDMS can be called from a trusted proxy, that can also obtain a mapped userid from an already-verified client certificate. This is a generalization, and provision as a system service, of what has been provided for some time now by various proxy services such as ''Web-to-DFS'' access solutions.

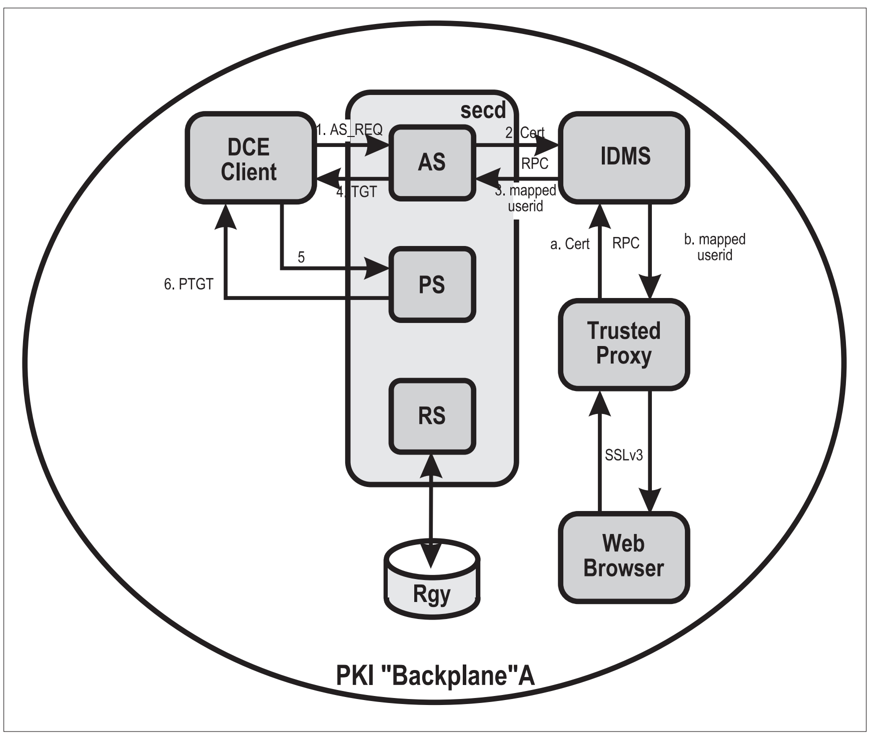

Figure 3-1 Overview of Certificate-Based Login

3.1 TGT Acquisition Protocol

The DCE Public Key TGT acquisition protocol is a subset of the protocol described in [DRAFT-PKINIT], using the option for user's private keys being stored locally on a CDSA-accessed smart card in the reference implementation. Note that other implementations of the *pkinit_cms_** function may store the user's private keys in another manner.

The DCE login APIs (*sec_login_validate_identity*( ), *sec_login_valid_and_cert_ident*( ), and *sec_login_validate_first*( )) attempt to use this protocol initially by default as long as Public Key authentication information can be constructed. If Public Key authentication information can not be constructed, then the default for the initial attempt is the OSF DCE Third Party protocol. If OSF DCE Third Party authentication information can not be constructed, then the default for the initial attempt is the Timestamps protocol (for which information can always be constructed).

If the KDC is unable to authenticate the user with the supplied public key pre-authentication data, the KDC returns error information.

If the initial public key login attempt fails, then the *sec_login* code falls back to the existing symmetric key password-based authentication.

A two-message protocol is used to acquire a TGT. This protocol relies, in part, on time stamps to guarantee the freshness of messages. There is no reason to adopt a challenge-response mechanism since the subsequent Kerberos protocols rely on time stamps. Since the TGT session key is encrypted with a random key that is encrypted with the public key of the client, successful use of the TGT implies the ability to decrypt this session key, and therefore possession of the user's private key.

The authentication information is transmitted in the pre- authentication data fields of the standard Kerberos V5 KRB_AS_REQ and KRB_AS_REP messages [IETF 1510] as new PA-PK-AS-REQ (Type 14) and PA-PK-AS-REP (Type 15) pre-authentication data types.

Note: As an implementation optimization and for backwards compatibility with pre-1.2.3 servers, the client sends both Third-Party (PADATA-ENC-OSF-DCE) and Public Key (PA- PK-AS-REQ) PADATA in the initial TGT request. The Third-Party PADATA is the first PADATA stored in the request. Pre-1.2.3 servers examine and verify the first PADATA, and ignore any remaining PADATA. DCE 1.2.3 servers examine and verify each PADATA type. If the Third-Party PADATA can not be verified, but the Public Key PADATA can, then the KDC returns a TGT to the client using the Public Key reply protocol.

The protocol usage criteria can be shown as follows in Table 1.

The ''TP can be built'' column indicates whether a Third- Party PADATA structure can be built by the *sec_login* client code.

The ''PK can be built'' column indicates whether Public Key Protocol information can be built by the *sec_login* client code. This can be built only if the client has a smart card and if the supplied passphrase is valid for gaining access to that smart card.

The ''PADATA sent'' column indicates which PADATA types are sent in the KRB_AS_REQ, and in what order.

The ''PADATA verified'' column indicates which PADATA type must pass verification in order for a TGT to be returned and which protocol will be used for the PADATA in the KRB_AS_REP. If there is no possibility of a TGT to be returned, the column indicates ''None''.

VERSIONS		CASES			PROTOCOLS USED	
Client version	Server version	TP can be built	PK can be built	Password valid	PADATA sent +	PADATA verified +
1.2.3	1.2.3	Yes	Yes	Yes	TP, PK	PK
1.2.3	1.2.3	Yes	Yes	No	TP, PK	PK
1.2.3	1.2.3	Yes	No	Yes	TP	TP
1.2.3	1.2.3	Yes	No	No	TP	None
1.2.3	1.2.3	No	Yes	Yes	TS, PK	PK
1.2.3	1.2.3	No	Yes	No	TS, PK	PK
1.2.3	1.2.3	No	No	Yes	TS	TS
1.2.3	1.2.3	No	No	No	TS	None
1.2.3	<1.2.3	Yes	Yes	Yes	TP, PK	TP
1.2.3	<1.2.3	Yes	Yes	No	TP, PK	None
1.2.3	<1.2.3	Yes	No	Yes	TP	TP
1.2.3	<1.2.3	Yes	No	No	TP	None
1.2.3	<1.2.3	No	Yes	Yes	TS, PK	TS
1.2.3	<1.2.3	No	Yes	No	TS, PK	None
1.2.3	<1.2.3	No	No	Yes	TS	TS
1.2.3	<1.2.3	No	No	No	TS	None
<1.2.3	1.2.3	Yes	N/A	Yes	TP	TP
<1.2.3	1.2.3	Yes	N/A	No	TP	None
<1.2.3	1.2.3	No	N/A	Yes	TS	TS
<1.2.3	1.2.3	No	N/A	No	TS	None

Note:

+ TS: Timestamps PADATA (KRB5_PADATA_ENC_UNIX_TIME from pre-1.2.3 clients, KRB5_PADATA_ENC_UNIX_TIME followed by KRB5_PADATA_ENC_TIMESTAMP from 1.2.3 clients)

+ TP: Third-Party PADATA (KRB5_PADATA_ENC_OSF_DCE)

+ PK: Public Key PADATA (PA-PK-AS-REQ, PA-PK-AS-REP)

Table 3-1 Protocol Usage Criteria

Note: The following protocol descriptions are necessarily a high-level simplification of the actual protocols used. For full details, see [IETF 1510], [DRAFT-PKINIT] and [DRAFT-CMS].

3.1.1 Client-to-KDC Message

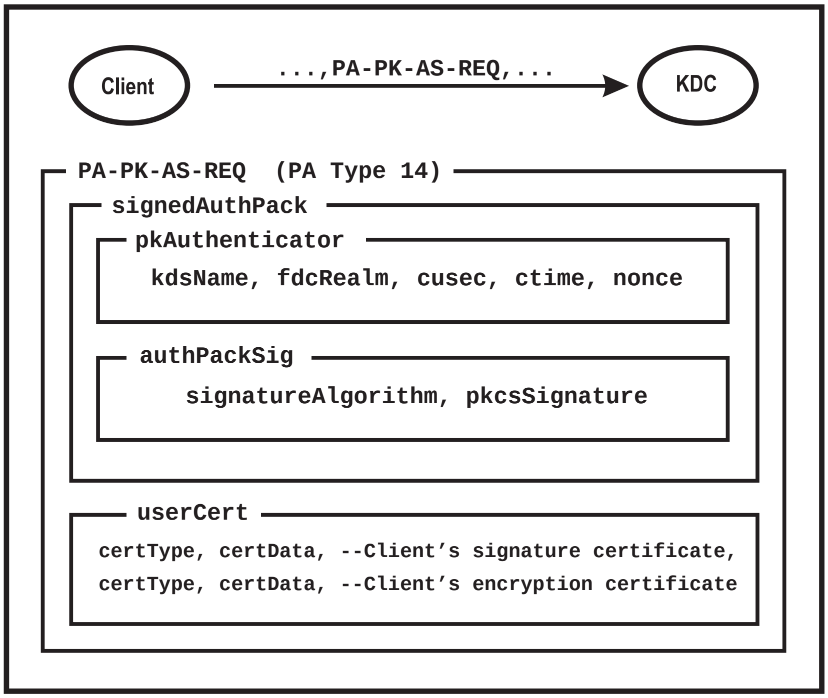

Figure 3-2 Client-to-KDC Request Overview

As shown in Figure 3 above, the client process creates a CMS ‘‘external signature’’ object, using the *pkinit_cms_sign_as_req* function, to create pkcsSignature. The pkAuthenticator includes the identity of the KDC, a time stamp and a nonce. The signature is created with the client’s private digital signature key. The signedAuthPack object is sent to the KDC along with the client’s signature and encryption certificates as the contents of the PADATA (Type 14) field of a standard KRB_AS_REQ message. The client’s identity is part of the existing KRB_AS_REQ message. It is initially set to the value provided to the *dce_login* command and/or the *sec_login_** APIs. The KDC’s Authentication Service (AS) will call the secure Identity Mapping Service (IDMS) to map the client’s ‘‘true identity,’’ as embodied in its signature certificate, to a userid value that the AS will use to construct a TGT that will be returned to the client as part of the KRB_AS_REP message.

3.1.2 KDC-to-Client Message

Figure 4 below shows a simplified overview of the reponse generated by the KDC and returned to the client.

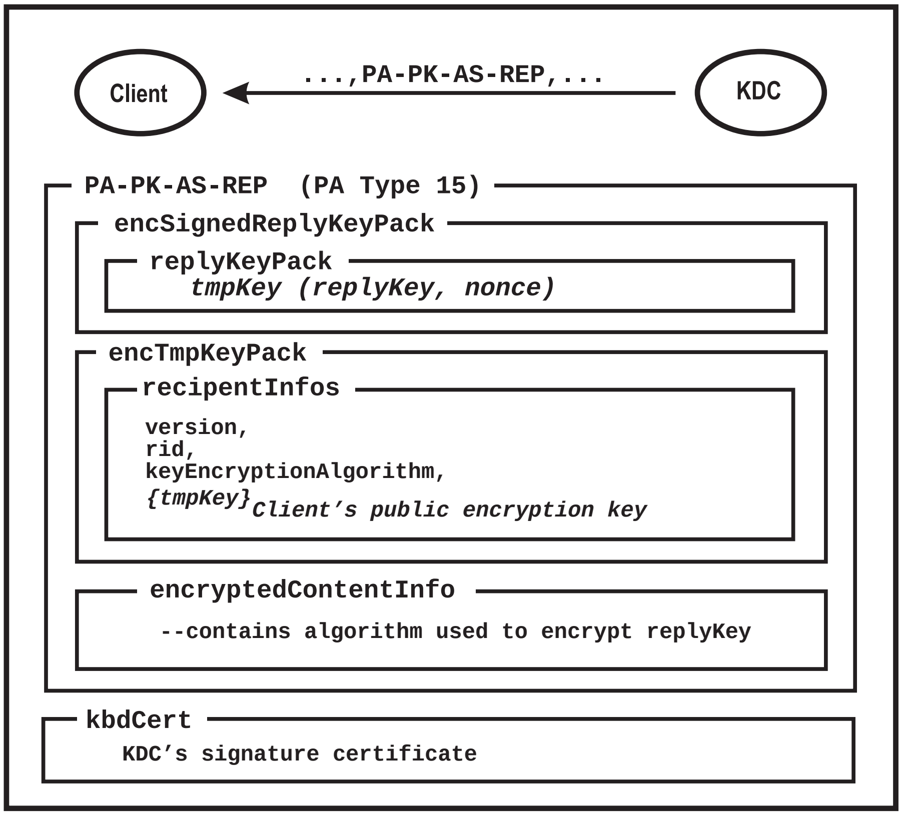

Figure 3-3 KDC-to-Client Response Overview

The KDC uses *pkinit_cms_** functions to:

- Validate the client's signature and encryption certificates.
- Validate the client's signature and extract the pkAuthenticator.

The KDC checks that the time stamp is sufficiently current. The KDC then calls the secure Identity Mapping Service (IDMS) to obtain the DCE userid to be assigned to the client based on its certificate-based identity. This userid is treated as the principal name. The KDC verifies the existence of this name in the Rgy, and places it in the *cname* field of the KRB_AS_REP message that the KDC then builds. This message contains the PA-PK-AS-REP (Type 15) PADATA field that contains a random symmetric reply key (*replyKey*) and the client's nonce. The reply key and client nonce are first signed using the KDC's private digital signature key, then encrypted using a temporary random symmetric key (*tmpKey*). This temporary random symmetric key is encrypted with the client's public key-encipherment key. The combination of symmetrically encrypted signed data and asymmetrically encrypted key is called digital enveloping. The reply key is used to encrypt the encrypted portion of the standard KRB_AS_REP, which includes the symmetric session key associated with the TGT. The KDC includes its signature certificate in the PADATA field of the response.

Note that it is the intent of the authors to register a new authorization data type (ad-type) with the IETF CAT WG, tentatively named OSF-DCE-PKI-CERTID, that can be used to return ‘‘original identity’’ information in the TGT. For performance and networking reasons it's undesirable to place the client's entire certificate in this structure. The authors propose that the information be based on the [DRAFT-CMS] CertUid definition (in ASN.1):

```
OSF-DCE-PKI-CERTID ::= CertUid
CertUid ::= SEQUENCE {
  issuerAndSerialNumber   IssuerAndSerialNumber,
  hashIssuerPublicKey     HashIssuerPublicKey OPTIONAL}

HashIssuerPublicKey ::= SEQUENCE {
  hashOid                 ObjectIdentifier,
  hashedIssuerPublicKey   OCTET STRING}
```

from the client's signature certificate. This is sufficient to guarantee the ''reasonable uniqueness'' of the original identity information.

Note: *An alternative under consideration to the ''hard coding'' of the contents of OSF-DCE-PKI-CERTID is to make its content an output from the IDMS. This might also facilitate principal-to-principal principal mapping as discussed later in the section on IDMS IDL.*

Note that it is also the intent to ensure that DCE properly handles multiple instances of the optional authorization-data field of Kerberos tickets. The ASN.1 definition is

```
AuthorizationData ::= SEQUENCE OF SEQUENCE {
                      ad-type[0]   INTEGER,
                      ad-data[1]   OCTET STRING }
```

OSF-DCE-PKI-CERTID is mapped into the OCTET STRING of *ad-data*. The *ad-type* value for OSF-DCE-PKI-CERTID will be assigned by the IETF.

DCE should ensure that it processes those ad-types it understands, and passes through those it does not.

*pkinit_cms_** functions will be used to construct both *encSignedReplyKeyPack* and *encTmpKeyPack*.

The TGT is passed in the standard KRB_AS_REP ticket field. The TGT is returned without additional encryption (portions of it were encrypted by the KDC) since it is subsequently used in the clear by the client. The symmetric session key (*replyKey*) used in association with the TGT is returned in *replyKeyPack*.

By verifying the KDC's signing certificate and checking the KDC's signature on this response, the client can be assured that the reply is from the KDC. The nonce is also checked. The session key can only be decrypted by the legitimate client who possesses the private key needed to decrypt the key encryption key. The TGT and associated session key are then used as normal.

New Kerberos Error Types

Per [DRAFT-PKINIT], the following new Kerberos error types are defined.

```
KDC_ERR_CLIENT_NOT_TRUSTED    62
KDC_ERR_KDC_NOT_TRUSTED       63
KDC_ERR_INVALID_SIG           64
KDC_ERR_KEY_TOO_WEAK          65
KDC_ERR_CERTIFICATE_MISMATCH  66
```

Note that these PKI-related errors such as signatures and trust issues are handled below the pkinit_cms_**layer*.

3.1.3 Changes to Existing TGT Acquisition Protocols

The existing protocols, prior to the function introduced by DCE RFC 68.3 in DCE 1.2.2, are unchanged. The DCE RFC 68.3 function is superceded by this document and is no longer supported. For the new certificate-based login support to be used in a DCE cell, the master DCE security server (i.e., the **secd** daemon) and all replicas will have to be at the new level of function defined by this document.

3.2 Passwords

During login operations, including *dce_login* and *dcecp*> login, the string entered as the password value is used first as a passphrase in an attempt to access the *pkinit_cms_** functions. If this failsthen the string is used as a DCE shared-secret password.

Except for login operations, the *dcecp -password* option always refers to a user's DCE shared-secret password.

A user's *pkinit_cms_** passphrase value may or may not match the DCE shared-secret password value.

3.3 pkinit_cms_* Overview and APIs

The *pkinit_cms_** set of APIs provide an abstraction layer for all Cryptographic Message Syntax (CMS) services required to build and consume all CMS-formatted content with the PA-PK-AS-REQ/REP PADATA portions of the Kerberos KRB_AS_REQ/REP messages. It is a design and implementation goal to package the *pkinit_cms_** APIs in a DLL for maximum flexibility. As shown in Figure 6 below, the reference implementation provides a *pkinit_cms_** built using the S/MIME Freeware Library and CDSA. Other *pkinit_cms_** DLLs could be created using other CMS SDKs and used to augment or replace the reference implementation's version.

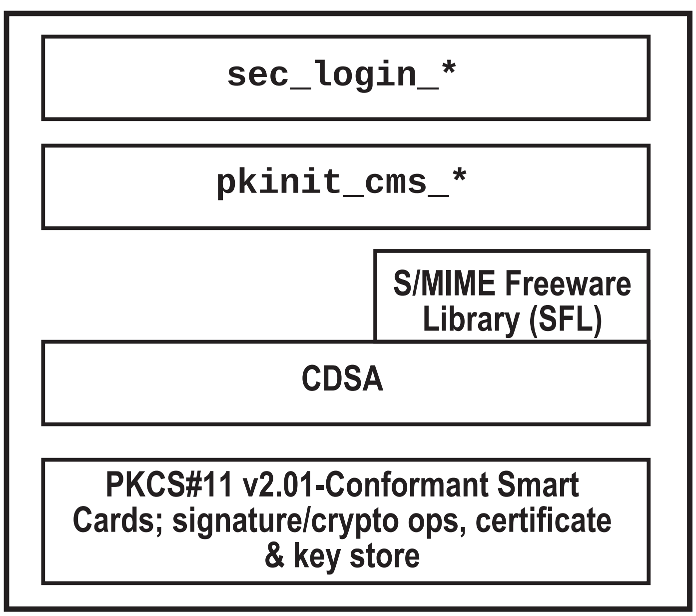

Figure 3-4 pkinit_cms_* Overview

3.3.1 pkinit_cms_open

This API is called from the ''CMS-ized'' *sec_psm_open*( ) to unlock and initialize the underlying CMS functions, including the creation and return of a CMS handle that points to the CMS context.

Syntax

```
unsigned long pkinit_cms_open(
        const char *name,
        const char *passphrase,
        cmsToolkitHandle_t *cms_h )
```

Parameters

name [in] username or path to token
passphrase [in] passphrase
cms_h [out] opaque cms-toolkit context

Return values

Returned 0 for successfully and other for error

When possible, the types will be changed to match DCE's existing types.

3.3.2 pkinit_cms_close()

This API is called from *sec_psm_close*( ) to perform cleanup operations, including deletion of the CMS context.

Syntax

```
unsigned long pkinit_cms_close(cmsToolkitHandle_tcms_h)
```

Parameters

cms_h [in] reference to cms-toolkit context

Return values

Returned 0 for successfully and other for error

3.3.3 pkinit_cms_sign_as_req()

This API is called by the client's *krb5_pkinit_sign_as_req*( ) to generate the CMS ''external signature'' object.

Syntax

```
unsigned long pkinit_cms_sign_as_req(
        cmsToolkitHandle_t csm_h,
        pkinit_cms_data_t *inputData,
        pkinit_cms_data_t *signedCmsOutput)
```

Parameters

cms_h [in] cms-toolkit context
inputData [in] input buffer
signedCmsOutput [out] signed, CMS formatted, DER encoded output

Return values

Returned 0 for successfully and other for error

3.3.4 pkinit_cms_verify_as_req()

This server is called by the KDC's *krb5_pkinit_decode_as_req*( ) to verify and parse the client's CMS SignedData object.

Syntax

```
unsigned long pkinit_cms_verify_as_req(
        cmsToolkitHandle_t cms_h,
        pkinit_cms_data_t *signedCmsInput,
        pkinit_cms_data_t *outputData,
        cmsCertHandle_t *signCertReference,
        cmsCertHandle_t *encrCertReference )
```

Parameters

cms_h [in] cms-toolkit context
signedCmsInput [in] signed, CMS formatted, DER encoded input
outputData [out] verified data
signCertReference [out] signature cerificate info of requester; needed by the AS to pass to IDMS
encrCertReference [out] encryption certificate info of requester

Return values

Returned 0 for successfully and other for error

3.3.5 pkinit_cms_sign_enc_as_rep()

This API is called by the KDC's *krb5_pkinit_sign_as_rep*( ) to produce the CMS-like-formatted contents of the PA-PK-AS-REP portion of the Kerberos KRB_AS_REP message.

Syntax

```
unsigned long pkinit_cms_sign_enc_as_rep(
        cmsToolkitHandle_t cms_h,
        cmsCertHandle_t *encrCertReference,
        pkinit_cms_data_t *input,
        pkinit_cms_data_t *envelopedCmsData )
```

Parameters

cms_h [in] cms-toolkit context
encCertInfo [in] encryption cert info
input [in] input buffer
envelopedCmsData [out] signed, encrypted, CMS formatted, DER encode output

Return values

Returned 0 for successfully and other for error

3.3.6 pkinit_cms_ver_dec_as_rep()

This API is called by the client's *krb5_pkinit_decode_as_rep*( ) to decrypt and verify the output from the KDC's *pkinit_cms_sign_enc_as_rep*( ).

Syntax

```
unsigned long pkinit_cms_ver_dec_as_rep(
        cmsToolkitHandle_t cms_h,
        pkinit_cms_data_t *envelopedCmsData,
        pkinit_cms_data_t *decryptedVerifiedOutput);
```

Parameters

cms_h [in] cms-toolkit context

envelopedCmsDat [in] signed, encrypted, CMS formatted, DER encode data.

decryptedVerifiedOutput [out] decrypted

Return values

Returned 0 for successfully and other for error

3.3.7 Identity Mapping Service

The Identity Mapping Service (IDMS) is a Secure RPC server that takes an already-verified X.509v3 certificate as input and maps it to a DCE principal name that's returned as its primary output. ''Figure 6: Identity Mapping Service'' below illustrates the use of the IDMS by both the Authentication Service (AS) and a trusted proxy such as might be used by a secure web server that's authenticated the client's signature certificate using Secure Sockets Layer (SSL) v3.

A basic IDMS sample source file will included for each installation to customize in order to implement its own mapping policies. Note that since the primary input to IDMS is the client's signature certificate, the IDMS code must be able to parse the certificate for pertinent information such as the client's DN, the name of the certificate issuer, the serial number of the certificate, etc. The subcomponents of the certificate can be used to determine the mapping. For example, the DN could be used as a key to search an LDAP directory with the desired DCE principal name. The reference implementation requires each installation using IDMS to have a software product capable of parsing the certificates.

Note that since the IDMS is part of the TCB, changes to the IDMS source must be carefully designed and reviewed so as not to compromise the integrity of the TCB.[5]

---

5. It has been suggested that it might be desirable in a specific environment for the AS not to call the IDMS. Said environment would be when an installation issues its own certificates and its PKI is configured to trust only its own certificates. In such an environment it might be possible to use the *SubjectAltName* X.509v3 certificate extension to bind a DCE principal to the DN. This would support one-to-one and many-to-one mappings. However, there are potential security exposures with such an approach, and in general, it's not a good idea to place potentially volatile information in the relatively static signature certificate.

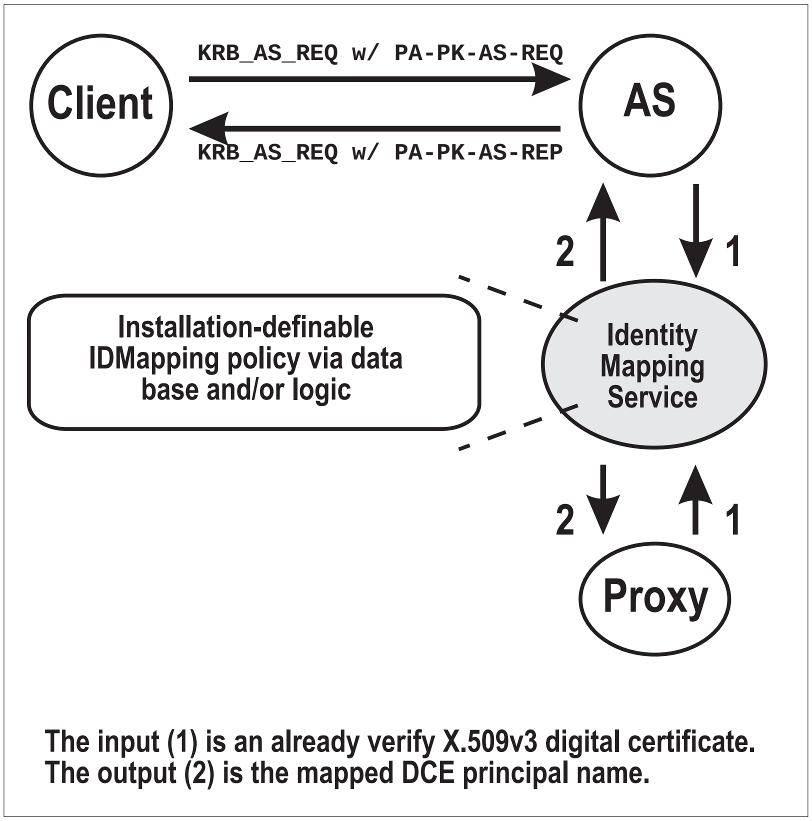

Figure 3-5 Identity Mapping Service.

3.3.8 IDMS IDL

```
/*
 * HISTORY
 * $Log: idms_serv.idl,v $
 * Revision 1.1.1.1  1998/06/11  16:11:47  sae
 *      ID Mapping server
 *
 * $EndLog$
 */
/*
 * FILE NAME:
 *      idms_serv.idl
 *
 * DESCRIPTION
 *      RPC interface exported by all ID Mapping functions.
 */
[
    uuid(272490ac-175f-11d2-9502-0004ac622bd7),
        pointer_default(ptr),
    version(1.0)
]
interface idms
```

```
{
  import "dce/rgynbase.idl";

  /* rsec_pk_idms_x509_to_user
   *
   * maps an (already-verified) asn1_cert into a user principal value.
   *
   * Input:
   *    handle: RPC binding handle. Allows client and sever to choose levels
   *    of encryption and authentication.
   *
   *    asn1_cert:   ASN.1 encoded certificate.
   *
   * Output:
   *    mapped_user: a string representing the principal value
   *    Null means unauthenticated?
   *
   *    stp:  Used for reporting both RPC communication errors and server
   *    errors processing the request.  The following errors may be returned
   *
   *    rsec_pk_idms_not_authorized
   *
   *
   */

  void rsec_pk_idms_x509_to_user(
       [in]       handle_t       handle,
       [in]       byte           *asn1_cert,
       [in]       unsigned long  asn1_cert_len,
       [in, out]  char           *mapped_user,
       [in, out]  error_status_t *stp
    );
```

---

6. An area under investigation is the possibility of supporting two interfaces to the IDMS. The first sends an already-verified (i.e., authenticated) user signature certificate to the IDMS. The IDMS, per installation policy, performs a mapping to a DCE principal name and returns it. The second interface would support principal-to-principal mapping, with the goal to be the acquisition of the same security credentials, regardless of the authentication method used. Special ERAs could be assigned to principals to select the IDMS or the conventional process for setting the cname field of the TGT. This might be of particular use for cases where DCE is acting in its role as a ‘‘vanilla Kerberos’’ server. If this alternative bears out, details will be announced at some future time. Note that adopting this approach would require that the IDMS return the ‘‘original identity’’ (certificate info or principal name) to the AS to be placed in the OSF-DCE-PKI-CERTID structure within the TGT. The name of the field would probably need to be changed to something like OSF DCE-ORIG-IDENT.

3.4 Accountability

Accountability for pre-1.2.3 DCE is unchanged. The changes to accountability are required for the ‘‘many-to- few’’ case. That is, when multiple client certificates are mapped by the IDMS to one DCE principal name. Since the DCE Audit service is based on principals/principal UUIDs, a change is needed so that servers that use the DCE Audit Service will create records containing the client’s original certificate-based identity.

3.4.1 Audit Service Enhancements

Note that since the Audit trail syntax is not part of the DCE AES, the following information should be considered informational.

The DCE Audit Service will be enhanced to extract the OSF-DCE-PKI-CERTID information, if present, from the client’s RPC binding handle and save it in audit records.

3.4.2 sec_id_get_certid()

This new API enables an application to obtain the OSF- DCE-PKI-CERTID information, if present, from an RPC binding handle.

```
error_status_t sec_id_get_certid(
   [in]   rpc_binding_handle_t  *binding_handle,
   [out]  byte                  *certid
);

Return status
   error_status_ok:    Success.
   Other (non-zero):   The ASN.1 CertUid construct is
                       not present in the binding handle.
```

3.4.3 gssdce_extract_certid_from_cred()

Purpose

Extracts a OSF-DCE-PKI-CERTID from a GSSAPI credential.

Format

```
#include <dce/gssapi.h>
OM_uint32 gssdce_extract_certid_from_cred(
OM_uint32 *minor_status,
gss_cred_id_t context_handle,
byte *certid);
```

Parameters

Input

context_handle Specifies the handle of the security context containing the credential.

Output

certid Returns the OSF-DCE-PKI-CERTID.

minor_status Returns a status code from the security mechanism.

Return Codes

This routine returns the following major status codes:

GSS_S_COMPLETE The routine was completed successfully.

GSS_S_FAILURE The routine failed. Check the minor_status parameter for details.

Data Structures

```
typedef struct sec_psm_context {
      void *magic;
      char *name; /* canonical name */
      char *pwd;
      unsigned32 mechanism_index;
      sec_pk_mechanism_handle_t mechanism_handle;
      sec_pk_domain_t domain_id;
} sec_psm_context_t, sec_psm_context_p_t;

typedef struct cmsCert_context {
      char *CertDn;
      char *CertIssuer;
} cmsCert_context_t, cmsCert_context_p_t;

typedef struct{
      unsigned long len;
      unsigned char *data;
} pkinit_cms_data_t;

/* unsigned long is an unsigned long defined in nbase.idl */

typedef void *cmsToolkitHandle_t;

typedef void *cmsCertHandle_t;

/* routine to free the opaque certifcates data through the handle */

void freeCmsCert(cmsCertHandle_t aCmsCert);
```

Chapter 5

Interfaces

5.1 User Interfaces

5.1.1 DCE Login

User interfaces to login utilities have not changed, except that additional new error conditions may be reported.

Login utilities such as *dce_login* invoke the existing *sec_login_** APIs, which changes only by the addition of new error status values that can be returned. Login utilities still need to prompt for a user name and a password.

For certificate-based login, the ‘‘password’’ that the user supplies is first used by *sec_login_** as a passphrase to access the *pkinit_cms_** functions. An environment variable, DCE_PKI_INI, is set at each client needing to use the new certificate-based login function.

This environment variable contains information needed to connect the login process with the underlying PKI via the *pkinit_cms** DLL.[7]

5.2 Management Interfaces

Minimal management interfaces are provided. CDSA framework management will be provided by the particular CDSA implementation used. PKI management will be handled by an installation’s particular PKI.

5.2.1 Installation

Installing the new public key functionality requires:

1. Stopping DCE. Installing the software upgrades (client, security server, IDMS, dced).
2. Adding any additional Secure RPC clients to the IDMS’s Access Control List.
3. Modifying the IDMS source code to reflect the installation’s identity mapping policies; re-building the IDMS executable.
4. Restarting DCE.

---

7. For example, if an EntrustFile toolkit-based *pkinit_cms** DLL is used, the environment variable might contain (1) the location of the Entrust .INI file on the user’s machine, and (2) the name of the Entrust .EPF file to be accessed by the DLL.

5.2.2 DCE Security Service Configuration

Notes to Reviewers

This section with side shading will not appear in the final copy. - Ed.

Contents to be determined. The text for this section to be supplied during the review period.

5.2.3 Enabling OSF DCE 1.2.3 Features

By default, all OSF DCE 1.2.3 features are disabled in a cell originally configured with a release prior to OSF DCE 1.2.3. Once software supporting DCE Public Key Certificate Login has been installed on all DCE Security Server replicas, public key functionality, along with other OSF DCE 1.2.3 functionality, can be enabled using the following *dcecp* command:

```
dcecp> registry modify -version secd.dce.1.2.3
```

When OSF DCE 1.2.3 features are enabled, any DCE Security Server replicas that do not support OSF DCE 1.2.3 features are shut down automatically.

A new cell configured with OSF DCE 1.2.3 release software has OSF DCE 1.2.3 features enabled from the start.

Chapter 6

Restrictions and Limitations

6.1 Exportability

6.1.1 Export of Binary (Executable) Code

Note that the BSAFE code shipped with DCE 1.2.2 implementations is no longer used and can be eliminated from implementations of DCE 1.2.3. All cryptographic operations associated with certificate-based login are encapsulated within the *pkinit_cms_** DLL.

The functionality provided by the binary code for the *pkinit_cms_** functions is not exportable unless its use is confined to the authentication process in such a way that users are unable to use the interfaces to encrypt and decrypt arbitrary data. Implementing vendors have a choice of supporting non-exportable and exportable versions of the DLL (although there may still be ‘‘crypto with a hole issues’’ - an assessment of S/MIME products needs to be made to get some direction on this). Alternatively, they may implement *pkinit_cms_** functions in such a way as to be exportable.[8]

It is the responsibility of each implementing vendor/ISV to determine how to build the *pkinit_cms_** DLL for their platform, and to verify that the resulting product is indeed exportable.

6.1.2 Export of Source Code

Implementations conforming to this specification will utilize underlying encryption and key management services. Source code which contains calls to such encryption routines will be subject to export controls.

6.2 Performance

Specific performance targets will need to be set for implementations of this specification. A preliminary examination of the current DCE Security Server (secd) code indicates that great care will have to be taken in moving some operations out of secd’s single address space to the PKI, the IDMS and the CAS. Reliability, availability and serviceability (RAS) challenges, as well as performance impediments will be introduced by this new function. Latency issues with fetching certficates and CRLs from LDAP directories are handled by the PKI, not DCE. Some tuning of the underlying PKI with respect to DCE may be possible.

---

8. It is the intent that a reference implementation of this specification will enable derived DCE products to be readily approved for export. This will be achieved by implementing PKI facilities specified in this document atop a CDSA-based software smart-card which incorporates appropriate key recovery technology. See also section 6.1 for more about export issues; and section 7.1 for more about the CDSA component dependency.

Chapter 7

Other Component Dependencies

7.1 CDSA

The reference implementation of this specification will provide a software implementation of a smart card that is accessed through the Common Data Security Architecture [CDSA] framework. Vendors implementing the *pkinit_cms_**DLL can do so using CDSA or any CMS and cryptographic SDK. The choice of underlying technology should be transparent to DCE users and programmers.

The contents of the DCE_PKI_INI environment variable may vary depending on an installations underlying PKI and/or the *pkinit_cms_** implementation. Otherwise, the choice of underlying technology should be transparent to DCE users and programmers.

Note: It's the authors' intent to use the IBM KeyWorks (a.k.a. SCCS Toolkit) SDK to provide CDSA for the reference implementation. Other vendors implementing the *pkinit_cms_** DLL can also, if they wish, license/use KeyWorks, for this purpose.

7.2 S/MIME Freeware Library

The S/MIME Freeware Library (SFL) is produced by J.G. Van Dyke & Associates, Inc. (http://www.jgvandyke.com). It's available to organizations without paying any royalties or licensing fees. Note that subsequent to the publication of Draft 0.4 of this document that SFL has been placed under export control by the United States Government.

Chapter 8

Compatibility

8.1 Migration

Notes to Reviewers

This section with side shading will not appear in the final copy. - Ed.

Contents to be determined. The text for this section to be supplied during the review period.

8.2 Standards

[ITU X.208], [ITU X.209], [IETF 1510], IETF[2253].

Appendix A

Implemention Issues

1. Need to assess the impact, if any, of this Specification on the DCE GSS-API support.

2. There are concerns regarding the use of SFL for CMS functions. SFL is only at the ‘‘Beta 0.3’’ level; the source is C++, not ANSI C; and it’s only working on Win32 platforms, *i.e.*, there’s basic cross-platform porting work to be done as well. There’s also a concern with the recent export controls slapped onto SFL.

3. The new pre-authentication fields (types 14 and 15) are potentially large, given that whole X.509v3 certificates are included in the authentication flows. The DCE Security Server (secd) and DCE Login clients use UDP for communications only as a backup when DCE RPCs fail to make the connection. Therefore, the following concerns apply primarily to secd’s role as a vanilla Kerberos KDC. The [DRAFT-PKINIT] recommends the support of TCP in lieu of UDP as the message sizes grow. However, this is not being considered as part of this specification. The IP datagram size needs to be sufficiently large to contain an entire message within a single datagram. This is to prevent a Kerberos `KRB_ERR_FIELD_TOOLONG` error condition. The current DCE design does not handle fragmented datagrams. Unlike some Kerberos implementations, DCE has no capability of retrying the communication using TCP when the message length exceeds the maximum datagram size. Architecturally, datagrams have a maximum length of 65,536 bytes (including the header). However, pragmatically the maximum datagram size is usually less. Some DCE implementations (*e.g.*, IBM’s AIX and OS/390 DCEs) dynamically determine the maximum datagram size based upon an exchange of the maximum sizes between the communicating partners. Host machines in a DCE cell should be configured with datagram sizes of between 8 KB and 16 KB, with the knowledge that this has potential impacts to network and host performance. Underneath the IP datagram size, the physical networks between hosts fragment/reassemble datagrams to fit across their respective MTUs (Maximum Transmission Units. Since DCE currently only supports raw UDP, there are potential reliability and performance problems associated with missing fragments.

4. If the *pkinit_cms_** function is implemented as a dynamic link library (DLL) in order to provide flexibility, there is no standard method across all DCE platforms to provide a ‘‘secure program load’’ facility to ensure the integrity of the *pkinit_cms_** function. This problem is not unique to *pkinit_cms_**.

5. Exportability issues need to be investigated in the light of current S/MIME offerings. If *pkinit_cms_** is packaged as a DLL, applications would have access to its encryption capabilities. Current S/MIME SDKs use 40-bit RC2 and 512-bit RSA keys for their exportable versions. This is insufficient for purposes of DCE authentication. The minimum should probably be set at Triple DES with 2048-bit RSA keys.

   Note that the software smart card, provided as part of the reference implementation, is designed to be exportable by virtue of its use of KeyWorks Key Recovery (KR) functions.

6. Need to verify smart card PKCS#11 (Cryptoki) functions to identify and select certificates and integrate with the *sec_login+pkinit_cms_** process.

   Note: *There appears to be no agreed-to set of schema for this. The reference implementation will work with the CDSA provider to determine what help is possible via the CDSA framework and service provider modules.*

7. Need to get the OSF-DCE-PKI-CERTID Kerberos Authorization Data type registered (and a numeric value assigned to it) with the Kerberos standards owners in the IETF CAT WG.

The goal is to have this work item complete by the 42nd IETF meeting being held August 1998 in Chicago, Illinois USA.

8. The existing Kerberos-based login process assumes / enforces that the username passed via the cname field of the ticket from the client to the KDC remains the same. Since the AS calls the IDMS to map the certificate-based identity to a DCE principal, and places the mapped value into the cname field of the TGT, this causes problems back on the client side. The ''issue'' relative to solving this problem is how to ''do it inexpensively'' vis-'-vis the existing code base.

Terminology

The same terminology and notation used in [RFC 85.0] is carried over here, with a few additions:

- CMS - Cryptographic Message Syntax. See [DRAFT-CMS].
- ERA - OSF DCE 1.1 Extended Registry Attribute. See [RFC 6.0].
- ASN.1 - Abstract Syntax Notation 1. A notation defined in [ITU X.208] for describing abstract types and values.
- BER - Basic Encoding Rules. A set of rules defined in [ITU X.209] and used to encode ASN.1 values as strings of octets. A single value can have multiple valid BER encodings.
- DER - Distinguished Encoding Rules. A restricted form of BER defined in [ITU X.509] to eliminate most of the ambiguities in BER.
- Smart Card - A multi-purpose, tamper-resistant, portable personal security device, utilizing VLSI chip technology for information storage and processing.
- User - The human user (and any associated private key storage).
- Client - An application running on the user's workstation. The login process is an example of a client.
- KDC - The Kerberos Key Distribution Center.[9]
- TGT - A Kerberos Ticket Granting Ticket.
- $K\{M\}$ - Message M encrypted with symmetric (*a.k.a.* secret) key K.
- $\{M\}_x$ - Message M encrypted with X's public key.
- $[M]_x$ - Message M signed with X's private key.

---

9. No distinction is made here between the Authentication Service (AS) and the Ticket Granting Service (TGS) KDC subservices, for reasons of clarity.