Open Software Foundation
Request For Comments: 68.4
Draft 0.4
April 1998

F. Siebenlist (DASCOM)
D. Hemsath (IBM)

DCE v.r.m PUBLIC KEY CERTIFICATE LOGIN — FUNCTIONAL SPECIFICATION

Synopsis

This RFC 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 public key certificate 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.

The functionality defined in this RFC 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 credentials out of the DCE Registry and into an LDAP directory, therefore consolidating the (logical) storage and access of PKI and DCE information.

Notes to RFC reviewers:

1. This is a preliminary draft. There are multiple details yet to be worked out. However, the authors believed it was vital to provide a "snapshot" of the current specification for review at the San Diego meeting of The Open Group. Comments are welcome and actively sought.

2. This draft RFC is a result of the work begun in September 1997 between The Open Group (TOG) and the Securities Industry Middleware Council (SIMC). The goal of this work is to accelerate the design and implementation of DCE function that will enable the use of standard X.509v3 digital certificates, digital signatures and an underlying Public Key Infrastructure for initial authentication to DCE.

3. The standards-based approach described in this RFC has the approval of TOG and SIMC. It also has the approval of the following DCE vendors: DASCOM, Digital, Gradient, HP and IBM. Entrust also agrees with the approach described in this RFC.

4. The authors wish to acknowledge the contributions of Entrust's Tim Moses, Chase's Gerry Gebel in his role of SIMC committee chair, Gradient's Brian Breton and Kevin Farrington, and Intellisoft's Jonathan Chinitz in getting the specification to this point. The authors also wish to express their appreciation to The Open Group staff and members for their continued backing of this effort. Last, but not least, our thanks to DASCOM's Greg Clark for getting the CMS work introduced and accepted into the IETF's Kerberos Public Key Initialization standard at the eleventh hour.

5. The intent of the interested parties is to submit this draft RFC to The Open Group's "fast track" process and deliver a reference implementation later this year.

Table of Contents

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 RFC. *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 RFC, 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.

The authentication information and protocol are based on the PK-INIT Kerberos protocol [DRAFT-PKINIT]. The reference implementation of this RFC requires that the authenticating user's or programmatic entity's signature and encryption certificates and corresponding private keys[1] be 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 RFC 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 (CMS—see [DRAFT-CMS]). CMS is an open standard derived from PKCS Number 7 Version 1.5. 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

[1] 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 RFC specifies that the encryption certificate be duplicated from the signature certificate.

multiple vendors[2]. This RFC defines a `pkinit_cms_*` abstraction layer that handles all required CMS functions. The reference implementation of this RFC provides a `pkinit_cms_*` based on the S/MIME Freeware Library (see [DRAFT-SFL]) and CDSA.

1.1. Changes Since Last Publication

Changes since [RFC 68.3]:

(a) 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 the proposal and is incorporating it into [DRAFT-PKINIT].

(b) 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.

(c) Users' public keys are no longer stored in the DCE Registry. They are obtained from users' X.509v3 public key certificates.

(d) A secure credential acquisition service is introduced to enable flexible mapping of users' Distinguished Names (DNs) [IETF 1779] embodied in their certificates to DCE Extended Privilege Attribute Certificates (EPACs).

(e) 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`.

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:

(a) 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.

(b) 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.

3. Goals and Non-Goals

3.1. Goals

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

(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.

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].

4. Terminology

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

(a) **CMS** — Cryptographic Message Syntax. See [DRAFT-CMS].

(b) **ERA** — OSF DCE 1.1 Extended Registry Attribute. See [RFC 6.0].

(c) **ASN.1** — Abstract Syntax Notation 1. A notation defined in [ITU X.208] for describing abstract types and values.

(d) **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.

(e) **DER** — Distinguished Encoding Rules. A restricted form of BER defined in [ITU X.509] to eliminate most of the ambiguities in BER.

(f) **Smart Card** — A multi-purpose, tamper-resistant, portable personal security device, utilizing VLSI chip technology for information storage and processing.

(g) **User** — The human user (and any associated private key storage).

(h) **Client** — An application running on the user's workstation. The login process is an example of a client.

(i) **KDC** — The Kerberos Key Distribution Center.[3]

(j) **TGT** — A Kerberos Ticket Granting Ticket.

(k) $K\{M\}$ — Message *M* encrypted with symmetric (a.k.a. secret) key *K*.

(l) $\{M\}_X$ — Message *M* encrypted with *X*'s public key.

(m) $[M]_X$ — Message *M* signed with *X*'s private key.

5. 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.

5.1. Business Requirements

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

5.2. Technical Requirements

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

(b) The function must be predicated, where appropriate, on other open standards from ITEF (*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 DN embodied in the client's signature certificate to a DCE EPAC. 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 DNs to a common EPAC.

(d) "DCE-less" clients, e.g., secure web browsers with client certificates, should be able to acquire DCE credentials. Note that this type of proxied login has been implemented in several forms by multiple vendors. A

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

requirement exists for a standard " Credential Acquisition" service for proxies or other middle-tier servers. A good example scenario was generated by the SIMC members and is shown in "Figure 1: SIMC Example" 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-v.r.m 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.

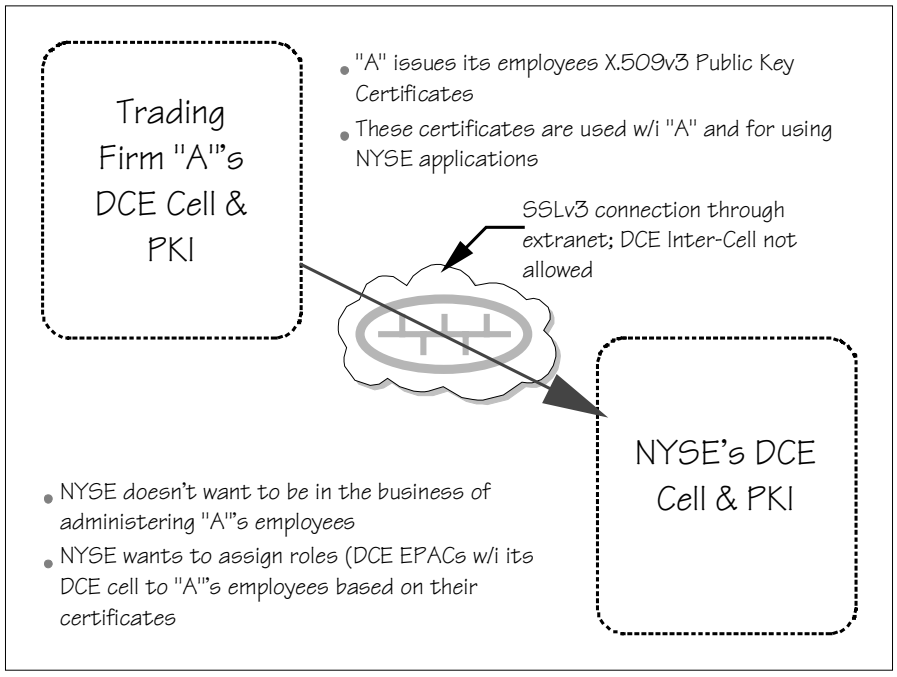

Figure 1: SIMC Example

6. Functional Definition

6.1. 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 key being stored locally on a CDSA-accessed smart card.

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, unless the KDC error information indicates that the principal is required to use public key pre-authentication. Sites that do not wish to allow any fall-back must not have a principal defined in the DCE Registry (Rgy). The value of such a principal name would be equal to the Common Name (CN) portion of the client's DN. Therefore, it can be seen that a potential for "name space collisions" is possible. However, this is a transient problem. As an installation moves towards certificates for authentication and LDAP for holding credentials, old-style principals will be eliminated from Rgy.

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-v.r.m 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-v.r.m servers examine and verify the first PADATA, and ignore any remaining PADATA. DCE v.r.m 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 diagrammed as follows.

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+
v.r.m	v.r.m	Yes	Yes	Yes	TP, PK	PK
v.r.m	v.r.m	Yes	Yes	No	TP, PK	PK
v.r.m	v.r.m	Yes	No	Yes	TP	TP*
v.r.m	v.r.m	Yes	No	No	TP	None
v.r.m	v.r.m	No	Yes	Yes	TS, PK	PK
v.r.m	v.r.m	No	Yes	No	TS, PK	PK
v.r.m	v.r.m	No	No	Yes	TS	TS*
v.r.m	v.r.m	No	No	No	TS	None
v.r.m	<v.r.m	Yes	Yes	Yes	TP, PK	TP

v.r.m	<v.r.m	Yes	Yes	No	TP, PK	None
v.r.m	<v.r.m	Yes	No	Yes	TP	TP
v.r.m	<v.r.m	Yes	No	No	TP	None
v.r.m	<v.r.m	No	Yes	Yes	TS, PK	TS*
v.r.m	<v.r.m	No	Yes	No	TS, PK	None
v.r.m	<v.r.m	No	No	Yes	TS	TS*
v.r.m	<v.r.m	No	No	No	TS	None
<v.r.m	v.r.m	Yes	N/A	Yes	TP	TP*
<v.r.m	v.r.m	Yes	N/A	No	TP	None
<v.r.m	v.r.m	No	N/A	Yes	TS	TS*
<v.r.m	v.r.m	No	N/A	No	TS	None

Notes:

** PADATA passes verification only if the client's effective* `pre_auth_req` *value allows the client to use this PADATA type*

+ TS: Timestamps PADATA (`KRB5_PADATA_ENC_UNIX_TIME` *from pre-v.r.m clients,* `KRB5_PADATA_ENC_UNIX_TIME` *followed by* `KRB5_PADATA_ENC_TIMESTAMP` *from v.r.m clients)*

+ TP: Third-Party PADATA (`KRB5_PADATA_ENC_OSF_DCE`*)*

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

Table 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].

6.1.1. Client-to-KDC Message

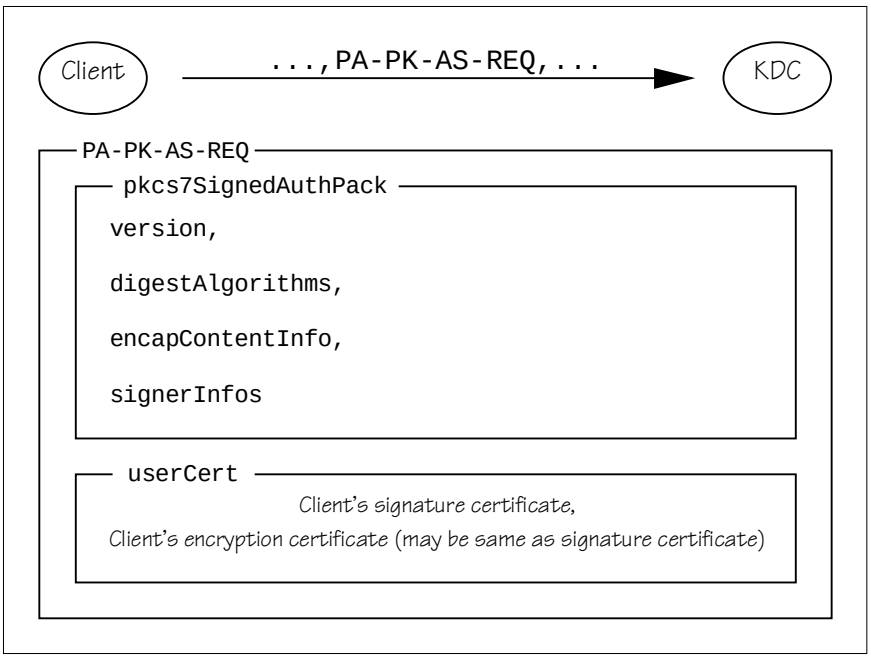

Figure 2: Client-to-KDC Message Overview

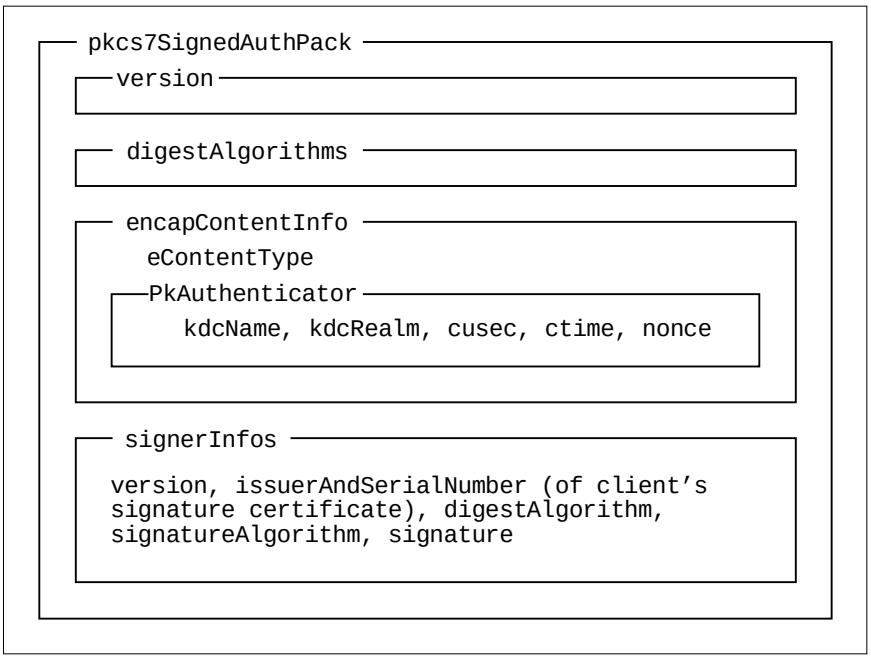

Figure 3: Detail of pkcs7SignedAuthPack (a CMS SignedData object)

The client process creates a CMS SignedData object, using the `pkinit_cms_sign_as_req` function. The encapsulated signed message (PkAuthenticator) includes the identity of the KDC, a time stamp and a nonce. The signature is done with the client's private digital signature key. This SignedData 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 an [IETF 1779] Distinguished Name (DN).

6.1.2. KDC-to-Client Message

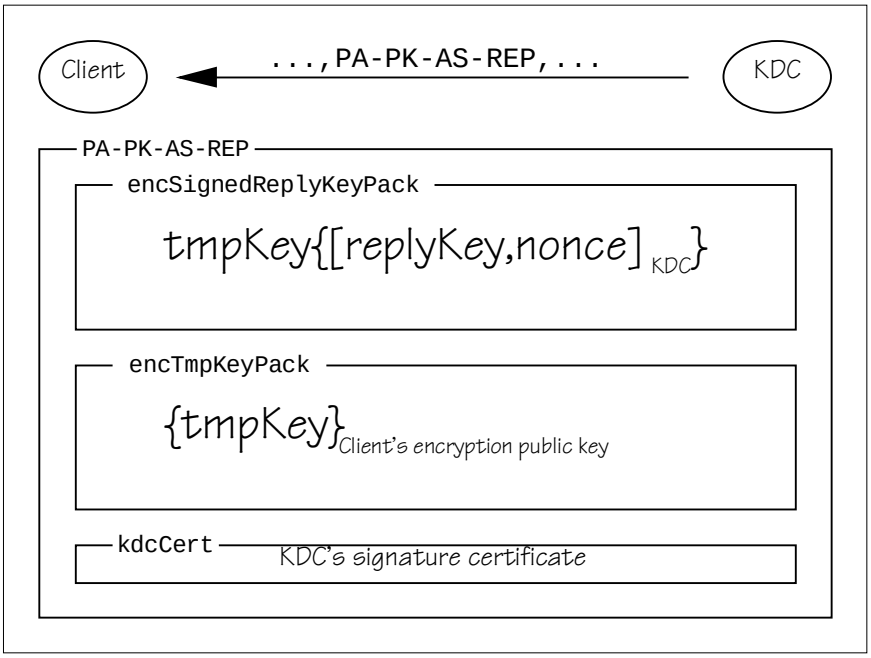

Figure 4: KDC-to-client response overview

The KDC uses `pkinit_cms_*` functions to:

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

The KDC verifies that the client's signature certificate matches the signerInfos.issuerAndSerialNumber. The KDC then checks the time stamp. If the time stamp is sufficiently current, the KDC builds the Kerberos `KRB_AS_REP` message in which the PA-PK-AS-REP (Type 15) PADATA field 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. The client's verified DN is returned in the `cname` field of the ticket.

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-CRED, that can be used in conjunction with tickets for future use. It is also the intent to ensure that DCE properly handles 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 }
```

DCE should ensure that it processes those `ad-type`s 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 used in association with the TGT is returned in the standard EncKDCRepPart field of the KRB_AS_REP message. This EncKDCRepPart field is encrypted using the reply key (replyKey) returned in the signed and encrypted authentication data from the KDC.

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 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.

6.1.3. Changes to Existing TGT Acquisition Protocols

<tbd>

6.2. 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 values may or may not match the DCE shared-secret password value.

6.3. 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 5: pkinit_cms_* Overview" 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. Note that the DLL packaging question depends on a satisfactory resolution of theTCB and exportability issues described in "16. O" below.

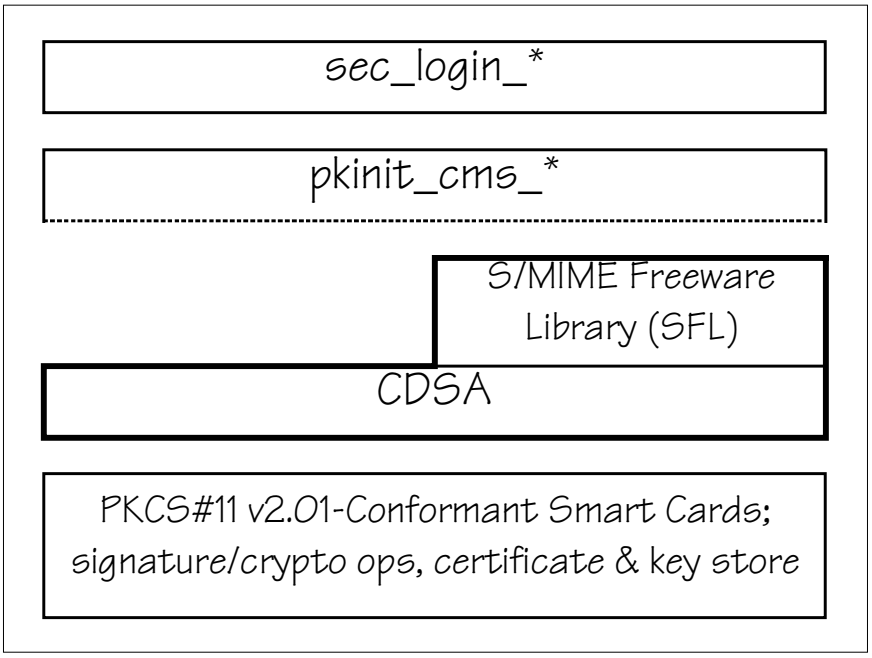

Figure 5: pkinit_cms_* Overview

6.3.1. 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.

6.3.2. pkinit_cms_close()

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

6.3.3. pkinit_cms_sign_as_req()

This API is called by the client's `krb5_pkinit_sign_as_req()` to generate the CMS SignedData object as shown in "Figure 3: Detail of pkcs7SignedAuthPack (a CMS SignedData object)" above.

6.3.4. 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.

6.3.5. pkinit_cms_sign_enc_as_rep()

This API is called by the KDC's `krb5_pkinit_sign_as_rep()` to produce the CMS-formatted contents of the `PA-PK-AS-REP` portion of the Kerberos `KRB_AS_REP` message, as shown in "Figure 4: KDC-to-client response overview" above.

6.3.6. 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()`.

6.4. Privilege Service

In "Figure 6: DN-Based Credential Acquisition" below, DCE credential information such as principal UUID, group UUIDs, etc. resides in an LDAP-accessible directory and are "keyed" by the client's verified DN. The DCE Privilege Service (PS) calls out to a new secure credential mapping/acquisition service to obtain the credential it needs to build an EPAC for the client based on the client's DN. LDAP is only an example of credential storage, albeit the preferred one as PKIs mature and LDAP becomes "the" directory service. For reasons of clarity, existing PS-Rgy functions to build an EPAC for a conventional SSK-authenticated principal are note shown.

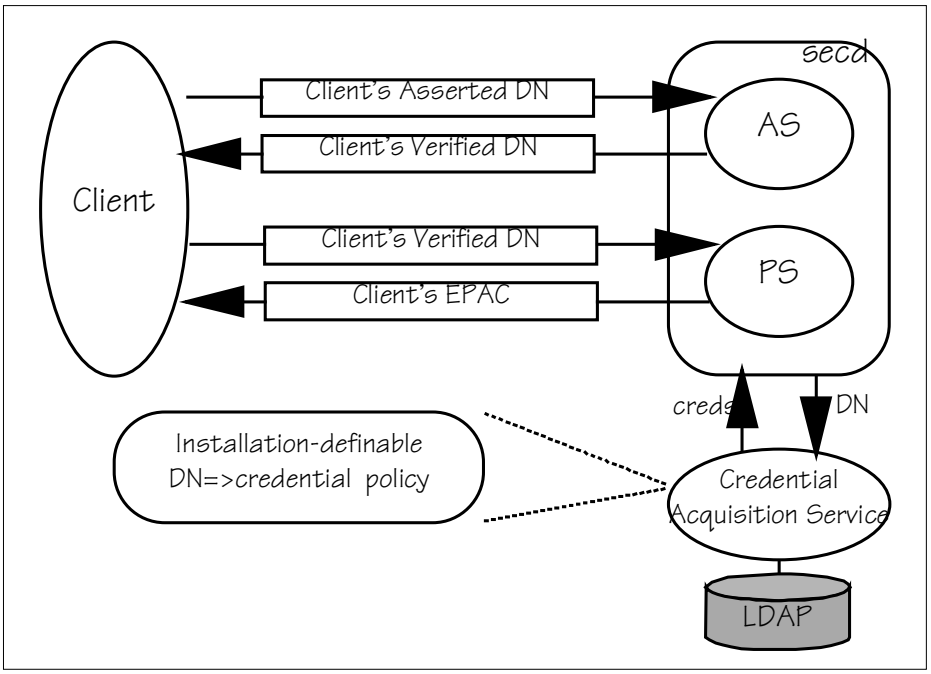

Figure 6: DN-Based Credential Acquisition

7. Data Structures

<tbd>

8. User Interfaces

<tbd>

9. APIs and Interfaces

<tbd>

10. Remote Interfaces

<tbd>

11. Management Interfaces

Minimal management interfaces are provided. CDSA framework management will be provided by the particular CDSA implementation used (e.g., IBM's KeyWorks). PKI management will be handled by an installation's particular PKI (e.g., Entrust).

11.1. Installation

Installing the new public key functionality requires only stopping DCE, installing the software upgrades (client, security server), and restarting DCE.

11.2. DCE Security Service Configuration

11.3. Enabling OSF DCE v.r.m Features

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

```
dcecp> registry modify -version secd.dce.v.r.m
```

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

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

11.4. Enabling Public Key Login

11.5. Configuring Public Key Login Users

12. Restrictions and Limitations

12.1. Exportability

12.1.1. Export of Binary (Executable) Code

<TBD>

12.1.2. Export of Source Code

<TBD>

12.2. Size

<TBD>

12.3. Performance

Actual performance targets are to-be-set. A preliminary examination of the current DCE Security Server (**secd**) code indicates that it great care will have to be taken in moving some operations out of **secd**'s single address space to the PKI and the Identity Mapping service. 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.

13. Other Component Dependencies

13.1. CDSA

It's the authors' intent to use the IBM KeyWorks (*a.k.a.* SCCS Toolkit) SDK to provide CDSA for the reference implementation.

13.2. 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.

14. Compatibility

14.1. Interoperability

<TBD>

14.2. Migration

<TBD>

The reference implementation will provide a utility to migrate relevant portions of the Rgy to an LDAP-accessed directory.

15. Standards

[ITU X.208], [ITU X.209], [IETF 1510], IETF[1779].

16. Open Issues

(a) Work is needed to ascertain feasibility of current DCE servers using shared secret key DCE Keytab files being able to use public key certificate-based login in addition to the current support.

(b) Are current smart cards capable of holding at least two (potentially large) public key certificates and their corresponding private keys?

(c) Work required to ensure integrity of Credential Acquisiion service as part of the Trusted Computing Base (TCB).

(d) 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 is not a problem unique to `pkinit_cms_*`.

(e) 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.

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

(g) Need to investigate using the [XSSO-PAM] APIs for any identity mapping and credential acquisition functions.

(h) Might want to consider building a "centralized smart card server" for the software smart card.

References

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

[DRAFT-CMS] IETF, "draft-ietf-smime-cms-04.txt", by R. Housley, March 1998.

[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, March 1998.

[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, “RFC 1510: The Kerberos Network Authentication Service (V5)", by J. Kohl and C. Neuman, September 1993.

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

[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, “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, “Recommendation X.208: Specification of abstract syntax notation one (ASN.1)”, ISO/IEC 8824, 1987.

[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.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, “ISO/IEC 9594-3/ITU-T Rec X.511, Information Technology — Open Systems Interconnection — The Directory: Abstract Service Definition”, 1993 (E).

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

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

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

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

[RFC 68.3] A. Anderson, S. Cuti, “DCE 1.2.2 Public Key Login—Functional Specification”, January 1997.

[RFC 80.0] J. Wray, “DCE Certification API — Functional Specification”, January 1995.

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

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

[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

Authors’ Addresses

Frank Siebenlist
DASCOM, Inc.

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

3004 Mission Street
Santa Cruz, CA 95060
USA

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

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