NETWORK WORKING GROUP
Internet-Draft
Expires: August 13, 2005

B. Tung
USC Information Sciences Institute
L. Zhu
Microsoft Corporation
February 9, 2005

# Public Key Cryptography for Initial Authentication in Kerberos
draft-ietf-cat-kerberos-pk-init-24

## Status of this Memo



## Copyright Notice



## Abstract

This document describes protocol extensions (hereafter called PKINIT) to the Kerberos protocol specification. These extensions provide a method for integrating public key cryptography into the initial authentication exchange, by using asymmetric-key signature and/or encryption algorithms in pre-authentication data fields.

Table of Contents

## 1. Introduction

A client typically authenticates itself to a service in Kerberos using three distinct though related exchanges. First, the client requests a ticket-granting ticket (TGT) from the Kerberos authentication server (AS). Then, it uses the TGT to request a service ticket from the Kerberos ticket-granting server (TGS). Usually, the AS and TGS are integrated in a single device known as a Kerberos Key Distribution Center, or KDC. (In this document, both the AS and the TGS are referred to as the KDC.) Finally, the client uses the service ticket to authenticate itself to the service.

The advantage afforded by the TGT is that the client exposes his long-term secrets only once. The TGT and its associated session key can then be used for any subsequent service ticket requests. One result of this is that all further authentication is independent of the method by which the initial authentication was performed. Consequently, initial authentication provides a convenient place to integrate public-key cryptography into Kerberos authentication.

As defined in [CLAR], Kerberos authentication exchanges use symmetric-key cryptography, in part for performance. One disadvantage of using symmetric-key cryptography is that the keys must be shared, so that before a client can authenticate itself, he must already be registered with the KDC.

Conversely, public-key cryptography (in conjunction with an established Public Key Infrastructure) permits authentication without prior registration with a KDC. Adding it to Kerberos allows the widespread use of Kerberized applications by clients without requiring them to register first with a KDC--a requirement that has no inherent security benefit.

As noted above, a convenient and efficient place to introduce public-key cryptography into Kerberos is in the initial authentication exchange. This document describes the methods and

data formats for integrating public-key cryptography into Kerberos initial authentication.

## 2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In this document, the encryption key used to encrypt the enc-part field of the KDC-REP in the AS-REP [CLAR] is referred to as the KDC AS reply key.





## 3. Extensions

This section describes extensions to [CLAR] for supporting the use of public-key cryptography in the initial request for a ticket.

Briefly, this document defines the following extensions to [CLAR]:

1. The client indicates the use of public-key authentication by including a special preauthenticator in the initial request. This preauthenticator contains the client's public-key data and a signature.

2. The KDC tests the client's request against its authentication policy and trusted Certification Authorities (CAs).

3. If the request passes the verification tests, the KDC replies as usual, but the reply is encrypted using either:

   a. a key generated through a Diffie-Hellman (DH) key exchange [RFC2631][IEEE1363] with the client, signed using the KDC's signature key; or

   b. a symmetric encryption key, signed using the KDC's signature key and encrypted using the client's public key.

   Any keying material required by the client to obtain the encryption key for decrypting the KDC reply is returned in a pre-authentication field accompanying the usual reply.

4. The client validates the KDC's signature, obtains the encryption key, decrypts the reply, and then proceeds as usual.

Section 3.1 of this document enumerates the required algorithms and necessary extension message types. Section 3.2 describes the extension messages in greater detail.

## 3.1 Definitions, Requirements, and Constants

### 3.1.1 Required Algorithms

All PKINIT implementations MUST support the following algorithms:

- KDC AS reply key enctype: AES256-CTS-HMAC-SHA1-96 etype [KCRYPTO].
- Signature algorithm: sha-1WithRSAEncryption [RFC3279].
- KDC AS reply key delivery method: Diffie-Hellman key exchange [RFC2631].



### 3.1.2 Defined Message and Encryption Types

PKINIT makes use of the following new pre-authentication types:

| | |
|---|---|
| PA_PK_AS_REQ | 16 |
| PA_PK_AS_REP | 17 |

PKINIT also makes use of the following new authorization data type:

| | |
|---|---|
| AD_INITIAL_VERIFIED_CAS | 9 |

PKINIT introduces the following new error codes:

| | |
|---|---|
| KDC_ERR_CLIENT_NOT_TRUSTED | 62 |
| KDC_ERR_INVALID_SIG | 64 |
| KDC_ERR_DH_KEY_PARAMETERS_NOT_ACCEPTED | 65 |
| KDC_ERR_CANT_VERIFY_CERTIFICATE | 70 |
| KDC_ERR_INVALID_CERTIFICATE | 71 |
| KDC_ERR_REVOKED_CERTIFICATE | 72 |
| KDC_ERR_REVOCATION_STATUS_UNKNOWN | 73 |
| KDC_ERR_CLIENT_NAME_MISMATCH | 75 |
| KDC_ERR_INCONSISTENT_KEY_PURPOSE | 76 |

PKINIT uses the following typed data types for errors:

| | |
|---|---|
| TD_TRUSTED_CERTIFIERS | 104 |
| TD_INVLID_CERTIFICATES | 105 |
| TD_DH_PARAMETERS | 109 |

PKINIT defines the following encryption types, for use in the AS-REQ message to indicate acceptance of the corresponding algorithms that can used by Cryptographic Message Syntax (CMS) [RFC3852] messages in the reply:

| | | |
|---|---|---|
| dsaWithSHA1-CmsOID | | 9 |
| md5WithRSAEncryption-CmsOID | | 10 |
| sha1WithRSAEncryption-CmsOID | | 11 |
| rc2CBC-EnvOID | | 12 |
| rsaEncryption-EnvOID | (PKCS1 v1.5) | 13 |
| rsaES-OAEP-EnvOID | (PKCS1 v2.0) | 14 |
| des-ede3-cbc-EnvOID | | 15 |

The ASN.1 module for all structures defined in this document (plus IMPORT statements for all imported structures) is given in Appendix A.

All structures defined in or imported into this document MUST be encoded using Distinguished Encoding Rules (DER) [X690] (unless otherwise noted). All data structures carried in OCTET STRINGs must





be encoded according to the rules specified in corresponding specifications.

Interoperability note: Some implementations may not be able to decode wrapped CMS objects encoded with BER but not DER; specifically, they may not be able to decode infinite length encodings. To maximize interoperability, implementers SHOULD encode CMS objects used in PKINIT with DER.

### 3.1.3 Algorithm Identifiers

PKINIT does not define, but does make use of, the following algorithm identifiers.

PKINIT uses the following algorithm identifiers for Diffie-Hellman key agreement [RFC3279]:

dhpublicnumber (Diffie-Hellman modulo a prime p [RFC2631])
id-ecPublicKey (Elliptic Curve Diffie-Hellman [IEEE1363])

PKINIT uses the following signature algorithm identifiers [RFC3279]:

```
sha-1WithRSAEncryption (RSA with SHA1)
md5WithRSAEncryption   (RSA with MD5)
id-dsa-with-sha1       (DSA with SHA1)
```

PKINIT uses the following encryption algorithm identifiers [RFC3447] for encrypting the temporary key with a public key:

```
rsaEncryption          (PKCS1 v1.5)
id-RSAES-OAEP          (PKCS1 v2.0)
```

PKINIT uses the following algorithm identifiers [RFC3370][RFC3565] for encrypting the KDC AS reply key with the temporary key:

```
des-ede3-cbc           (three-key 3DES, CBC mode)
rc2-cbc                (RC2, CBC mode)
id-aes256-CBC          (AES-256, CBC mode)
```

## 3.2 PKINIT Pre-authentication Syntax and Use

This section defines the syntax and use of the various pre-authentication fields employed by PKINIT.

### 3.2.1 Generation of Client Request

The initial authentication request (AS-REQ) is sent as per [CLAR]; in





addition, a pre-authentication data element, whose padata-type is PA_PK_AS_REQ and whose padata-value contains the DER encoding of the type PA-PK-AS-REQ, is included.

```
PA-PK-AS-REQ ::= SEQUENCE {
   signedAuthPack          [0] IMPLICIT OCTET STRING,
            -- Contains a CMS type ContentInfo encoded
            -- according to [RFC3852].
            -- The contentType field of the type ContentInfo
            -- is id-signedData (1.2.840.113549.1.7.2),
```

```
                -- and the content field is a SignedData.
                -- The eContentType field for the type SignedData is
                -- id-pkauthdata (1.3.6.1.5.2.3.1), and the
                -- eContent field contains the DER encoding of the
                -- type AuthPack.
                -- AuthPack is defined below.
   trustedCertifiers       [1] SEQUENCE OF TrustedCA OPTIONAL,
                -- A list of CAs, trusted by the client, that can
                -- be used as the trust anchor to validate the KDC's
                -- signature.
                -- Each TrustedCA identifies a CA or a CA
                -- certificate (thereby its public key).
   kdcPkId                 [2] IMPLICIT OCTET STRING
                                   OPTIONAL,
                -- Contains a CMS type SignerIdentifier encoded
                -- according to [RFC3852].
                -- Identifies, if present, a particular KDC
                -- public key that the client already has.
   ...
}

DHNonce ::= OCTET STRING

TrustedCA ::= SEQUENCE {
   caName                  [0] IMPLICIT OCTET STRING,
                -- Contains a PKIX type Name encoded according to
                -- [RFC3280].
                -- Specifies the CA distinguished subject name.
   certificateSerialNumber [1] INTEGER OPTIONAL,
                -- Specifies the certificate serial number.
                -- The defintion of the certificate serial number
                -- is taken from X.509 [X.509-97].
   subjectKeyIdentifier    [2] OCTET STRING OPTIONAL,
                -- Identifies the CA's public key by a key
                -- identifier.  When an X.509 certificate is
                -- referenced, this key identifier matches the X.509
                -- subjectKeyIdentifier extension value.  When other
                -- certificate formats are referenced, the documents
```





```
                -- that specify the certificate format and their use
                -- with the CMS must include details on matching the
                -- key identifier to the appropriate certificate
```

```
                -- field.
        ...
    }

    AuthPack ::= SEQUENCE {
        pkAuthenticator         [0] PKAuthenticator,
        clientPublicValue       [1] SubjectPublicKeyInfo OPTIONAL,
                -- Defined in [RFC3280].
                -- The pubic key value (the subjectPublicKey field
                -- of the type SubjectPublicKeyInfo) MUST be encoded
                -- according to [RFC3279].
                -- Present only if the client wishes to use the
                -- Diffie-Hellman key agreement method.
        supportedCMSTypes       [2] SEQUENCE OF AlgorithmIdentifier
                                    OPTIONAL,
                -- List of CMS encryption types supported by
                -- client in order of (decreasing) preference.
        clientDHNonce           [3] DHNonce OPTIONAL,
                -- Present only if the client indicates that it
                -- wishes to reuse DH keys or to allow the KDC to
                -- do so (see Section 3.2.3.1).
        ...
    }

    PKAuthenticator ::= SEQUENCE {
        cusec                   [0] INTEGER (0..999999),
        ctime                   [1] KerberosTime,
                -- cusec and ctime are used as in [CLAR], for replay
                -- prevention.
        nonce                   [2] INTEGER (0..4294967295),
                -- Chosen randomly;  This nonce does not need to
                -- match with the nonce in the KDC-REQ-BODY.
        paChecksum              [3] OCTET STRING,
                -- Contains the SHA1 checksum, performed over
                -- KDC-REQ-BODY.
        ...
    }
```

The ContentInfo [RFC3852] structure for the signedAuthPack field is filled out as follows:

1. The contentType field of the type ContentInfo is id-signedData (as defined in [RFC3852]), and the content field is a SignedData (as defined in [RFC3852]).

2. The eContentType field for the type SignedData is id-pkauthdata: { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) pkauthdata(1) }.

3. The eContent field for the type SignedData contains the DER encoding of the type AuthPack.

4. The signerInfos field of the type SignedData contains a single signerInfo, which contains the signature over the type AuthPack.

5. The certificates field of the type SignedData contains certificates intended to facilitate certification path construction, so that the KDC can verify the signature over the type AuthPack. For path validation, these certificates SHOULD be sufficient to construct at least one certification path from the client certificate to one trust anchor acceptable by the KDC [CAPATH]. The certificates field MUST NOT contain "root" CA certificates.

6. The client's Diffie-Hellman public value (clientPublicValue) is included if and only if the client wishes to use the Diffie-Hellman key agreement method. For the Diffie-Hellman key agreement method, implementations MUST support Oakley 1024-bit MODP well-known group 2 [RFC2412] and SHOULD support Oakley 2048-bit MODP well-known group 14 and Oakley 4096-bit MODP well-known group 16 [RFC3526].

   The Diffie-Hellman field size should be chosen so as to provide sufficient cryptographic security. The following table, based on [LENSTRA], gives approximate comparable key sizes for symmetric- and asymmetric-key cryptosystems based on the best-known algorithms for attacking them.

| Symmetric | ECC | DH/DSA/RSA |
|---|---|---|
| 80 | 163 | 1024 |
| 112 | 233 | 2048 |
| 128 | 283 | 3072 |
| 192 | 409 | 7680 |
| 256 | 571 | 15360 |

Table 1: Comparable key sizes (in bits)

   When Diffie-Hellma modulo a prime p is used, the exponents should have at least twice as many bits as the symmetric keys that will be derived from them [ODL99].

7. The client may wish to reuse DH keys or to allow the KDC to do so (see Section 3.2.3.1). If so, then the client includes the clientDHNonce field. This nonce string needs to be as long as the longest key length of the symmetric key types that the client supports. This nonce MUST be chosen randomly.

### 3.2.2 Receipt of Client Request

Upon receiving the client's request, the KDC validates it. This section describes the steps that the KDC MUST (unless otherwise noted) take in validating the request.

The KDC verifies the client's signature in the signedAuthPack field according to [RFC3852].

If, while validating the client's X.509 certificate [RFC3280], the KDC cannot build a certification path to validate the client's certificate, it sends back a KRB-ERROR [CLAR] message with the code KDC_ERR_CANT_VERIFY_CERTIFICATE. The accompanying e-data for this error message is a TYPED-DATA (as defined in [CLAR]) that contains an element whose data-type is TD_TRUSTED_CERTIFIERS, and whose data-value contains the DER encoding of the type TD-TRUSTED-CERTIFIERS:

```
TD-TRUSTED-CERTIFIERS ::= SEQUENCE OF TrustedCA
            -- Identifies a list of CAs trusted by the KDC.
            -- Each TrustedCA identifies a CA or a CA
            -- certificate (thereby its public key).
```

Upon receiving this error message, the client SHOULD retry only if it has a different set of certificates (from those of the previous requests) that form a certification path (or a partial path) from one of the trust anchors selected by the KDC to its own certificate.

If, while processing the certification path, the KDC determines that the signature on one of the certificates in the signedAuthPack field is invalid, it returns a KRB-ERROR [CLAR] message with the code KDC_ERR_INVALID_CERTIFICATE. The accompanying e-data for this error message is a TYPED-DATA that contains an element whose data-type is

TD_INVALID_CERTIFICATES, and whose data-value contains the DER encoding of the type TD-INVALID-CERTIFICATES:

```
    TD-INVALID-CERTIFICATES ::= SEQUENCE OF OCTET STRING
                -- Each OCTET STRING contains a CMS type
                -- IssuerAndSerialNumber encoded according to
                -- [RFC3852].
                -- Each IssuerAndSerialNumber indentifies a
```





```
                -- certificate (sent by the client) with an invalid
                -- signature.
```

If more than one X.509 certificate signature is invalid, the KDC MAY send one TYPED-DATA element per invalid signature.

Based on local policy, the KDC may also check whether any X.509 certificates in the certification path validating the client's certificate have been revoked. If any of them have been revoked, the KDC MUST return an error message with the code KDC_ERR_REVOKED_CERTIFICATE; if the KDC attempts to determine the revocation status but is unable to do so, it SHOULD return an error message with the code KDC_ERR_REVOCATION_STATUS_UNKNOWN. The certificate or certificates affected are identified exactly as for the error code KDC_ERR_INVALID_CERTIFICATE (see above).

The client's public key is then used to verify the signature. If the signature fails to verify, the KDC MUST return an error message with the code KDC_ERR_INVALID_SIG. There is no accompanying e-data for this error message.

In addition to validating the client's signature, the KDC MUST also check that the client's public key used to verify the client's signature is bound to the client's principal name as specified in the AS-REQ as follows:

1. If the KDC has its own binding between either the client's signature-verification public key or the client's certificate and the client's Kerberos principal name, it uses that binding.

2. Otherwise, if the client's X.509 certificate contains a SubjectAltName extension with a KRB5PrincipalName (defined below) in the otherName field, it binds the client's X.509 certificate to

that name.

The otherName field (of type AnotherName) in the SubjectAltName extension MUST contain KRB5PrincipalName as defined below.

The type-id field of the type AnotherName is id-pksan:

```
id-pksan OBJECT IDENTIFIER ::=
  { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2)
    x509-sanan (2) }
```

The value field of the type AnotherName is the DER encoding of the following ASN.1 type:





```
KRB5PrincipalName ::= SEQUENCE {
    realm                   [0] Realm,
    principalName           [1] PrincipalName
}
```

If the KDC does not have its own binding and there is no KRB5PrincipalName name present in the client's X.509 certificate, and if the Kerberos name in the request does not match the KRB5PrincipalName in the client's X.509 certificate (including the realm name), the KDC MUST return an error message with the code KDC_ERR_CLIENT_NAME_MISMATCH. There is no accompanying e-data for this error message.

The KDC MAY require the presence of an Extended Key Usage (EKU) KeyPurposeId [RFC3280] id-pkekuoid in the extensions field of the client's X.509 certificate:

```
id-pkekuoid OBJECT IDENTIFIER ::=
  { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2)
    pkinit(3) pkekuoid(4) }
       -- PKINIT client authentication.
       -- Key usage bits that MUST be consistent:
       -- digitalSignature;
       -- Key usage bits that MAY be consistent:
       -- nonRepudiation, and (keyEncipherment or keyAgreement).
```

If this EKU is required but is missing, the KDC MUST return an error message of the code KDC_ERR_INCONSISTENT_KEY_PURPOSE. There is no accompanying e-data for this error message. KDCs implementing this requirement SHOULD also accept the EKU KeyPurposeId id-ms-sc-logon (1.3.6.1.4.1.311.20.2.2) as meeting the requirement, as there are a large number of X.509 client certificates deployed for use with PKINIT which have this EKU.

If for any other reasons, the client's public key is not accepted, the KDC MUST return an error message with the code KDC_ERR_CLIENT_NOT_TRUSTED.

The KDC MUST check the timestamp to ensure that the request is not a replay, and that the time skew falls within acceptable limits. The recommendations for clock skew times in [CLAR] apply here. If the check fails, the KDC MUST return error code KRB_AP_ERR_REPEAT or KRB_AP_ERR_SKEW, respectively.

If the clientPublicValue is filled in, indicating that the client wishes to use the Diffie-Hellman key agreement method, the KDC SHOULD check to see if the key parameters satisfy its policy. If they do not, it MUST return an error message with the code





KDC_ERR_DH_KEY_PARAMETERS_NOT_ACCEPTED. The accompanying e-data is a TYPED-DATA that contains an element whose data-type is TD_DH_PARAMETERS, and whose data-value contains the DER encoding of the type TD-DH-PARAMETERS:

```
    TD-DH-PARAMETERS ::= SEQUENCE OF DHDomainParameters
                -- Contains a list of Diffie-Hellman domain
                -- parameters in decreasing preference order.

    DHDomainParameters ::= CHOICE {
        modp                    [0] DomainParameters,
                -- Type DomainParameters is defined in [RFC3279].
        ec                      [1] EcpkParameters,
                -- Type EcpkParameters is defined in [RFC3279].
        ...
    }
```

TD-DH-PARAMETERS contains a list of Diffie-Hellman domain parameters that the KDC supports in decreasing preference order, from which the

client SHOULD pick one to retry the request.

If the client included a kdcPkId field in the PA-PK-AS-REQ and the KDC does not possess the corresponding key, the KDC MUST ignore the kdcPkId field as if the client did not include one.

If the client included a trustedCertifiers field, and the KDC does not possesses the private key for any one of the listed certifiers, the KDC MUST ignore the trustedCertifiers field as if the client did not include any.

If there is a supportedCMSTypes field in the AuthPack, the KDC must check to see if it supports any of the listed types. If it supports more than one of the types, the KDC SHOULD use the one listed first. If it does not support any of them, it MUST return an error message with the code KDC_ERR_ETYPE_NOSUPP [CLAR].

### 3.2.3 Generation of KDC Reply

Assuming that the client's request has been properly validated, the KDC proceeds as per [CLAR], except as follows.

The KDC MUST set the initial flag and include an authorization data element of ad-type [CLAR] AD_INITIAL_VERIFIED_CAS in the issued ticket. The ad-data [CLAR] field contains the DER encoding of the type AD-INITIAL-VERIFIED-CAS:





```
AD-INITIAL-VERIFIED-CAS ::= SEQUENCE OF TrustedCA
            -- Identifies the certification path based on which
            -- the client certificate was validated.
            -- Each TrustedCA identifies a CA or a CA
            -- certificate (thereby its public key).
```

The AS wraps any AD-INITIAL-VERIFIED-CAS data in AD-IF-RELEVANT containers if the list of CAs satisfies the AS' realm's local policy (this corresponds to the TRANSITED-POLICY-CHECKED ticket flag [CLAR]). Furthermore, any TGS MUST copy such authorization data from tickets used in a PA-TGS-REQ [CLAR] of the TGS-REQ to the resulting ticket, and it can wrap or unwrap the data into or out-of the

AD-IF-RELEVANT container, depends on if the list of CAs satisfies the TGS' realm's local policy.

Application servers that understand this authorization data type SHOULD apply local policy to determine whether a given ticket bearing such a type *not* contained within an AD-IF-RELEVANT container is acceptable. (This corresponds to the AP server checking the transited field when the TRANSITED-POLICY-CHECKED flag has not been set [CLAR].) If such a data type is contained within an AD-IF-RELEVANT container, AP servers MAY apply local policy to determine whether the authorization data is acceptable.

The content of the AS-REP is otherwise unchanged from [CLAR]. The KDC encrypts the reply as usual, but not with the client's long-term key. Instead, it encrypts it with either a shared key derived from a Diffie-Hellman exchange, or a generated encryption key. The contents of the PA-PK-AS-REP indicate which key delivery method is used:

```
PA-PK-AS-REP ::= CHOICE {
   dhInfo                  [0] DHRepInfo,
            -- Selected when Diffie-Hellman key exchange is
            -- used.
   encKeyPack              [1] IMPLICIT OCTET STRING,
            -- Selected when public key encryption is used.
            -- Contains a CMS type ContentInfo encoded
            -- according to [RFC3852].
            -- The contentType field of the type ContentInfo is
            -- id-envelopedData (1.2.840.113549.1.7.3).
            -- The content field is an EnvelopedData.
            -- The contentType field for the type EnvelopedData
            -- is id-signedData (1.2.840.113549.1.7.2).
            -- The eContentType field for the inner type
            -- SignedData (when unencrypted) is id-pkrkeydata
            -- (1.2.840.113549.1.7.3) and the eContent field
            -- contains the DER encoding of the type
            -- ReplyKeyPack.
```





```
            -- ReplyKeyPack is defined in Section 3.2.3.2.
   ...
}

DHRepInfo ::= SEQUENCE {
```

```
      dhSignedData            [0] IMPLICIT OCTET STRING,
                -- Contains a CMS type ContentInfo encoded according
                -- to [RFC3852].
                -- The contentType field of the type ContentInfo is
                -- id-signedData (1.2.840.113549.1.7.2), and the
                -- content field is a SignedData.
                -- The eContentType field for the type SignedData is
                -- id-pkdhkeydata (1.3.6.1.5.2.3.2), and the
                -- eContent field contains the DER encoding of the
                -- type KDCDHKeyInfo.
                -- KDCDHKeyInfo is defined below.
      serverDHNonce           [1] DHNonce OPTIONAL
                -- Present if and only if dhKeyExpiration is
                -- present in the KDCDHKeyInfo.
   }

   KDCDHKeyInfo ::= SEQUENCE {
      subjectPublicKey        [0] BIT STRING,
                -- KDC's DH public key.
                -- The DH pubic key value is mapped to a BIT STRING
                -- according to [RFC3279].
      nonce                   [1] INTEGER (0..4294967295),
                -- Contains the nonce in the PKAuthenticator of the
                -- request if DH keys are NOT reused,
                -- 0 otherwise.
      dhKeyExpiration         [2] KerberosTime OPTIONAL,
                -- Expiration time for KDC's key pair,
                -- present if and only if DH keys are reused. If
                -- this field is omitted then the serverDHNonce
                -- field MUST also be omitted. See Section 3.2.3.1.
      ...
   }
```

### 3.2.3.1 Using Diffie-Hellman Key Exchange

In this case, the PA-PK-AS-REP contains a DHRepInfo structure.

The ContentInfo [RFC3852] structure for the dhSignedData field is filled in as follows:

1. The contentType field of the type ContentInfo is id-signedData (as defined in [RFC3852]), and the content field is a SignedData

(as defined in [RFC3852]).

2. The eContentType field for the type SignedData is the OID value for id-pkdhkeydata: { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) pkdhkeydata(2) }.

3. The eContent field for the type SignedData contains the DER encoding of the type KDCDHKeyInfo.

4. The signerInfos field of the type SignedData contains a single signerInfo, which contains the signature over the type KDCDHKeyInfo.

5. The certificates field of the type SignedData contains certificates intended to facilitate certification path construction, so that the client can verify the KDC's signature over the type KDCDHKeyInfo. This field may only be left empty if the KDC public key specified by the kdcPkId field in the PA-PK-AS-REQ was used for signing. Otherwise, for path validation, these certificates SHOULD be sufficient to construct at least one certification path from the KDC certificate to one trust anchor acceptable by the client [CAPATH]. The certificates field MUST NOT contain "root" CA certificates.

6. If the client included the clientDHNonce field, then the KDC may choose to reuse its DH keys (see Section 3.2.3.1). If the server reuses DH keys then it MUST include an expiration time in the dhKeyExperiation field. Past the point of the expiration time, the signature over the type DHRepInfo is considered expired/invalid. When the server reuses DH keys then it MUST include a serverDHNonce at least as long as the length of keys for the symmetric encryption system used to encrypt the AS reply. Note that including the serverDHNonce changes how the client and server calculate the key to use to encrypt the reply; see below for details. The KDC SHOULD NOT reuse DH keys unless the clientDHNonce field is present in the request.

The KDC AS reply key is derived as follows:

1. Both the KDC and the client calculate the shared secret value as follows:

   a) When Diffie-Hellman modulo a prime p ([RFC2631]) is used, let DHSharedSecret be the shared secret value.

b) When Elliptic Curve Diffie-Hellman (ECDH) (with each party contributing one key pair) [IEEE1363] is used, let DHSharedSecret be the x-coordinate of the shared secret value (an elliptic curve point).

DHSharedSecret is first padded with leading zeros such that the size of DHSharedSecret in octets is the same as that of the modulus, then represented as a string of octets in big-endian order.

Implementation note: Both the client and the KDC can cache the triple (ya, yb, DHSharedSecret), where ya is the client's public key and yb is the KDC's public key. If both ya and yb are the same in a later exchange, the cached DHSharedSecret can be used.

2. Let K be the key-generation seed length [KCRYPTO] of the KDC AS reply key whose enctype is selected according to [CLAR].

3. Define the function octetstring2key() as follows:

```
octetstring2key(x) == random-to-key(K-truncate(
                          SHA1(0x00 | x) |
                          SHA1(0x01 | x) |
                          SHA1(0x02 | x) |
                          ...
                          ))
```

where x is an octet string; | is the concatenation operator; 0x00, 0x01, 0x02, etc., are each represented as a single octet; random-to-key() is an operation that generates a protocol key from a bitstring of length K; and K-truncate truncates its input to the first K bits. Both K and random-to-key() are as defined in the kcrypto profile [KCRYPTO] for the enctype of the KDC AS reply key.

4. When DH keys are reused, let n_c be the clientDHNonce, and n_k be the serverDHNonce; otherwise, let both n_c and n_k be empty octet strings.

5. The KDC AS reply key k is:

```
k = octetstring2key(DHSharedSecret | n_c | n_k)
```

### 3.2.3.2 Using Public Key Encryption

In this case, the PA-PK-AS-REP contains a ContentInfo structure wrapped in an OCTET STRING. The KDC AS reply key is encrypted in the encKeyPack field, which contains data of type ReplyKeyPack:





```
ReplyKeyPack ::= SEQUENCE {
   replyKey                [0] EncryptionKey,
            -- Contains the session key used to encrypt the
            -- enc-part field in the AS-REP.
   nonce                   [1] INTEGER (0..4294967295),
            -- Contains the nonce in the PKAuthenticator of the
            -- request.
   ...
}
```

The ContentInfo [RFC3852] structure for the encKeyPack field is filled in as follows:

1. The contentType field of the type ContentInfo is id-envelopedData (as defined in [RFC3852]), and the content field is an EnvelopedData (as defined in [RFC3852]).

2. The contentType field for the type EnvelopedData is id-signedData: { iso (1) member-body (2) us (840) rsadsi (113549) pkcs (1) pkcs7 (7) signedData (2) }.

3. The eContentType field for the inner type SignedData (when decrypted from the encryptedContent field for the type EnvelopedData) is id-pkrkeydata: { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) pkrkeydata(3) }.

4. The eContent field for the inner type SignedData contains the DER encoding of the type ReplyKeyPack.

5. The signerInfos field of the inner type SignedData contains a single signerInfo, which contains the signature over the type ReplyKeyPack.

6. The certificates field of the inner type SignedData contains certificates intended to facilitate certification path

construction, so that the client can verify the KDC's signature over the type ReplyKeyPack. This field may only be left empty if the KDC public key specified by the kdcPkId field in the PA-PK-AS-REQ was used for signing. Otherwise, for path validation, these certificates SHOULD be sufficient to construct at least one certification path from the KDC certificate to one trust anchor acceptable by the client [CAPATH]. The certificates field MUST NOT contain "root" CA certificates.

7. The recipientInfos field of the type EnvelopedData is a SET which MUST contain exactly one member of type KeyTransRecipientInfo. The encryptedKey of this member contains the temporary key which is encrypted using the client's public key.



8. The unprotectedAttrs or originatorInfo fields of the type EnvelopedData MAY be present.

### 3.2.4 Receipt of KDC Reply

Upon receipt of the KDC's reply, the client proceeds as follows. If the PA-PK-AS-REP contains the dhSignedData field, the client derives the KDC AS reply key using the same procedure used by the KDC as defined in Section 3.2.3.1. Otherwise, the message contains the encKeyPack field, and the client decrypts and extracts the temporary key in the encryptedKey field of the member KeyTransRecipientInfo, and then uses that as the KDC AS reply key.

In either case, the client MUST verify the signature in the SignedData according to [RFC3852]. Unless the client can otherwise prove that the public key used to verify the KDC's signature is bound to the target KDC, the KDC's X.509 certificate MUST satisfy at least one of the follow two requirements:

1. The certificate contains a Subject Alternative Name (SAN) extension carrying a dNSName and that name value matches the following name format:

   _Service._Proto.Realm

   Where the Service name is the string literal "kerberos", the Proto can be "udp" or "tcp", and the Realm is the domain style Kerberos realm name [CLAR] of the target KDC. This name format is identical to the owner label format used in the DNS SRV records for specifying the KDC location as described in [CLAR]. For example, the X.509 certificate for the KDC of the Kerberos

realm "EXAMPLE.COM" would contain a dNSName value of "_kerberos._tcp.EXAMPLE.COM" or "_kerberos._udp.EXAMPLE.COM".

2. The certificate contains the EKU KeyPurposeId [RFC3280] id-pkkdcekuoid (defined below) and an SAN extension [RFC3280] carrying an AnotherName whose type-id is id-pksan (as defined in Section 3.2.2) and whose value contains a KRB5PrincipalName name, and the realm name of that KRB5PrincipalName matches the realm name of the target KDC.

```
id-pkkdcekuoid OBJECT IDENTIFIER ::=
  { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2)
    pkinit(3) pkkdcekuoid(5) }
       -- Signing KDC responses.
       -- Key usage bits that MUST be consistent:
       -- digitalSignature.
```

If no SAN id-pksan extension is present (but the id-pkkdcekuoid EKU is) in the KDC's X.509 certificate, and the client has a





priori knowledge of the KDC's hostname (or IP address), the client SHOULD accept the KDC's X.509 certificate if that certificate contains an SAN extension carrying a dNSName and the dNSName value matches the hostname (or the IP address) of the KDC with which the client believes it is communicating.

Matching rules used for the dNSName value are specified in [RFC3280].

If all applicable checks are satisfied, the client then decrypts the enc-part field of the KDC-REP in the AS-REP using the KDC AS reply key, and then proceeds as described in [CLAR].

Implementation note: CAs issuing KDC certificates SHOULD place all "short" and "fully-qualified" Kerberos realm names of the KDC (one per SAN extension) into the KDC certificate to allow maximum flexibility.

## 3.3 Interoperability Requirements

The client MUST be capable of sending a set of certificates sufficient to allow the KDC to construct a certification path for the client's certificate, if the correct set of certificates is provided

through configuration or policy.

If the client sends all the X.509 certificates on a certification path to a trust anchor acceptable by the KDC, and the KDC can not verify the client's public key otherwise, the KDC MUST be able to process path validation for the client's certificate based on the certificates in the request.

The KDC MUST be capable of sending a set of certificates sufficient to allow the client to construct a certification path for the KDC's certificate, if the correct set of certificates is provided through configuration or policy.

If the KDC sends all the X.509 certificates on a certification path to a trust anchor acceptable by the client, and the client can not verify the KDC's public key otherwise, the client MUST be able to process path validation for the KDC's certificate based on the certificates in the reply.

## 3.4 KDC Indication of PKINIT Support

If pre-authentication is required, but was not present in the request, per [CLAR] an error message with the code KDC_ERR_PREAUTH_FAILED is returned and a METHOD-DATA object will be stored in the e-data field of the KRB-ERROR message to specify which pre-authentication mechanisms are acceptable. The KDC can then





indicate the support of PKINIT by including an empty element whose padata-type is PA_PK_AS_REQ in that METHOD-DATA object.

Otherwise if it is required by the KDC's local policy that the client must be pre-authenticated using the pre-authentication mechanism specified in this document, but no PKINIT pre-authentication was present in the request, an error message with the code KDC_ERR_PREAUTH_FAILED SHOULD be returned.

KDCs MUST leave the padata-value field of the PA_PK_AS_REQ element in the KRB-ERROR's METHOD-DATA empty (i.e., send a zero-length OCTET STRING), and clients MUST ignore this and any other value. Future extensions to this protocol may specify other data to send instead of an empty OCTET STRING.

## 4. Security Considerations

PKINIT raises certain security considerations beyond those that can be regulated strictly in protocol definitions. We will address them in this section.

PKINIT extends the cross-realm model to the public-key infrastructure. Users of PKINIT must understand security policies and procedures appropriate to the use of Public Key Infrastructures [RFC3280].

Standard Kerberos allows the possibility of interactions between cryptosystems of varying strengths; this document adds interactions with public-key cryptosystems to Kerberos. Some administrative policies may allow the use of relatively weak public keys. Using such keys to wrap data encrypted under stronger conventional cryptosystems may be inappropriate.

PKINIT requires keys for symmetric cryptosystems to be generated. Some such systems contain "weak" keys. For recommendations regarding these weak keys, see [CLAR].

PKINIT allows the use of the same RSA key pair for encryption and signing when doing RSA encryption based key delivery. This is not recommended usage of RSA keys [RFC3447], by using DH based key delivery this is avoided.

Care should be taken in how certificates are chosen for the purposes of authentication using PKINIT. Some local policies may require that key escrow be used for certain certificate types. Deployers of PKINIT should be aware of the implications of using certificates that have escrowed keys for the purposes of authentication. Because signing only certificates are normally not escrowed, by using DH





based key delivery this is avoided.

PKINIT does not provide for a "return routability" test to prevent attackers from mounting a denial-of-service attack on the KDC by causing it to perform unnecessary and expensive public-key operations. Strictly speaking, this is also true of standard Kerberos, although the potential cost is not as great, because standard Kerberos does not make use of public-key cryptography. By

using DH based key delivery and reusing DH keys, the necessary crypto processing cost per request can be minimized.

The syntax for the AD-INITIAL-VERIFIED-CAS authorization data does permit empty SEQUENCEs to be encoded. Such empty sequences may only be used if the KDC itself vouches for the user's certificate.

## 5. Acknowledgements

The following people have made significant contributions to this draft: Paul Leach, Kristin Lauter, Sam Hartman, Love Hornquist Astrand, Ken Raeburn, Nicolas Williams, John Wray, Jonathan Trostle, Tom Yu, Jeffrey Hutzelman, David Cross, Dan Simon and Karthik Jaganathan.

Special thanks to Clifford Neuman, Matthew Hur and Sasha Medvinsky who wrote earlier versions of this document.

The authors are indebt to the Kerberos working group chair Jeffrey Hutzelman who kept track of various issues and was enormously helpful during the creation of this document.

Some of the ideas on which this document is based arose during discussions over several years between members of the SAAG, the IETF CAT working group, and the PSRG, regarding integration of Kerberos and SPX. Some ideas have also been drawn from the DASS system. These changes are by no means endorsed by these groups. This is an attempt to revive some of the goals of those groups, and this document approaches those goals primarily from the Kerberos perspective.

Lastly, comments from groups working on similar ideas in DCE have been invaluable.

## 6. IANA Considerations

This document has no actions for IANA.





## 7. References

## 7.1 Normative References


[CLAR] RFC-Editor: To be replaced by RFC number for draft-ietf-krb-wg-kerberos-clarifications. Work in Progress.

[IEEE1363] IEEE, "Standard Specifications for Public Key Cryptography", IEEE 1363, 2000.

[KCRYPTO] RFC-Editor: To be replaced by RFC number for draft-ietf-krb-wg-crypto. Work in Progress.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2412] Orman, H., "The OAKLEY Key Determination Protocol", RFC 2412, November 1998.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

[RFC3279] Bassham, L., Polk, W. and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.

[RFC3280] Housley, R., Polk, W., Ford, W. and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[RFC3370] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC3565] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, July 2003.

[RFC3852] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[X.509-97] ITU-T. Recommendation X.509: The Directory - Authentication Framework. 1997.

[X690] ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), ITU-T Recommendation X.690 (1997) | ISO/IEC International Standard 8825-1:1998.

## 7.2 Informative References

[CAPATH] RFC-Editor: To be replaced by RFC number for draft-ietf-pkix-certpathbuild. Work in Progress.

[LENSTRA] Lenstra, A. and E. Verheul, "Selecting Cryptographic Key Sizes", Journal of Cryptology 14 (2001) 255-293.

[ODL99] Odlyzko, A., "Discrete logarithms: The past and the future, Designs, Codes, and Cryptography (1999)".

## Authors' Addresses

Brian Tung
USC Information Sciences Institute
4676 Admiralty Way Suite 1001, Marina del Rey CA
Marina del Rey, CA 90292
US

Email: brian@isi.edu

Larry Zhu
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
US

Email: lzhu@microsoft.com

## Appendix A. PKINIT ASN.1 Module

```
KerberosV5-PK-INIT-SPEC {
        iso(1) identified-organization(3) dod(6) internet(1)
        security(5) kerberosV5(2) modules(4) pkinit(5)
} DEFINITIONS EXPLICIT TAGS ::= BEGIN

IMPORTS
    SubjectPublicKeyInfo, AlgorithmIdentifier
        FROM PKIX1Explicit88 { iso (1)
          identified-organization (3) dod (6) internet (1)
          security (5) mechanisms (5) pkix (7) id-mod (0)
          id-pkix1-explicit (18) }
          -- As defined in RFC 3280.

    DomainParameters, EcpkParameters
        FROM PKIX1Algorithms88 { iso(1)
          identified-organization(3) dod(6)
          internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
          id-mod-pkix1-algorithms(17) }
          -- As defined in RFC 3279.

    KerberosTime, TYPED-DATA, PrincipalName, Realm, EncryptionKey
        FROM KerberosV5Spec2 { iso(1) identified-organization(3)
          dod(6) internet(1) security(5) kerberosV5(2)
          modules(4) krb5spec2(2) } ;

id-pkinit OBJECT IDENTIFIER ::=
  { iso (1) org (3) dod (6) internet (1) security (5)
    kerberosv5 (2) pkinit (3) }

id-pkauthdata  OBJECT IDENTIFIER  ::= { id-pkinit 1 }
id-pkdhkeydata OBJECT IDENTIFIER  ::= { id-pkinit 2 }
id-pkrkeydata  OBJECT IDENTIFIER  ::= { id-pkinit 3 }
id-pkekuoid    OBJECT IDENTIFIER  ::= { id-pkinit 4 }
id-pkkdcekuoid OBJECT IDENTIFIER  ::= { id-pkinit 5 }

pa-pk-as-req INTEGER ::=                  16
pa-pk-as-rep INTEGER ::=                  17

ad-initial-verified-cas INTEGER ::=        9

td-trusted-certifiers INTEGER ::=        104
td-invalid-certificates INTEGER ::=      105
td-dh-parameters INTEGER ::=             109

PA-PK-AS-REQ ::= SEQUENCE {
   signedAuthPack          [0] IMPLICIT OCTET STRING,
```

```
                -- Contains a CMS type ContentInfo encoded
                -- according to [RFC3852].
```





```
                -- The contentType field of the type ContentInfo
                -- is id-signedData (1.2.840.113549.1.7.2),
                -- and the content field is a SignedData.
                -- The eContentType field for the type SignedData is
                -- id-pkauthdata (1.3.6.1.5.2.3.1), and the
                -- eContent field contains the DER encoding of the
                -- type AuthPack.
                -- AuthPack is defined below.
        trustedCertifiers       [1] SEQUENCE OF TrustedCA OPTIONAL,
                -- A list of CAs, trusted by the client, that can
                -- be used as the trust anchor to validate the KDC's
                -- signature.
                -- Each TrustedCA identifies a CA or a CA
                -- certificate (thereby its public key).
        kdcPkId                 [2] IMPLICIT OCTET STRING
                                    OPTIONAL,
                -- Contains a CMS type SignerIdentifier encoded
                -- according to [RFC3852].
                -- Identifies, if present, a particular KDC
                -- public key that the client already has.
        ...
    }

    DHNonce ::= OCTET STRING

    TrustedCA ::= SEQUENCE {
        caName                  [0] IMPLICIT OCTET STRING,
                -- Contains a PKIX type Name encoded according to
                -- [RFC3280].
                -- Identifies a CA.
                -- Prefer the sid field below if that is available.
        certificateSerialNumber [1] INTEGER OPTIONAL,
                -- Specifies the certificate serial number.
                -- The defintion of the certificate serial number
                -- is taken from X.509 [X.509-97].
        subjectKeyIdentifier    [2] OCTET STRING OPTIONAL,
                -- Identifies the CA's public key by a key
                -- identifier.  When an X.509 certificate is
                -- referenced, this key identifier matches the X.509
```

```
                -- subjectKeyIdentifier extension value.  When other
                -- certificate formats are referenced, the documents
                -- that specify the certificate format and their use
                -- with the CMS must include details on matching the
                -- key identifier to the appropriate certificate
                -- field.
        ...
    }
```





```
    AuthPack ::= SEQUENCE {
        pkAuthenticator         [0] PKAuthenticator,
        clientPublicValue       [1] SubjectPublicKeyInfo OPTIONAL,
                -- Defined in [RFC3280].
                -- The pubic key value (the subjectPublicKey field
                -- of the type SubjectPublicKeyInfo) MUST be encoded
                -- according to [RFC3279].
                -- Present only if the client wishes to use the
                -- Diffie-Hellman key agreement method.
        supportedCMSTypes       [2] SEQUENCE OF AlgorithmIdentifier
                                    OPTIONAL,
                -- List of CMS encryption types supported by
                -- client in order of (decreasing) preference.
        clientDHNonce           [3] DHNonce OPTIONAL,
                -- Present only if the client indicates that it
                -- wishes to reuse DH keys or to allow the KDC to
                -- do so.
        ...
    }

    PKAuthenticator ::= SEQUENCE {
        cusec                   [0] INTEGER (0..999999),
        ctime                   [1] KerberosTime,
                -- cusec and ctime are used as in [CLAR], for replay
                -- prevention.
        nonce                   [2] INTEGER (0..4294967295),
                -- Chosen randomly;  This nonce does not need to
                -- match with the nonce in the KDC-REQ-BODY.
        paChecksum              [3] OCTET STRING,
                -- Contains the SHA1 checksum, performed over
                -- KDC-REQ-BODY.
        ...
```

```
}

TD-TRUSTED-CERTIFIERS ::= SEQUENCE OF TrustedCA
            -- Identifies a list of CAs trusted by the KDC.
            -- Each TrustedCA identifies a CA or a CA
            -- certificate (thereby its public key).

TD-INVALID-CERTIFICATES ::= SEQUENCE OF OCTET STRING
            -- Each OCTET STRING contains a CMS type
            -- IssuerAndSerialNumber encoded according to
            -- [RFC3852].
            -- Each IssuerAndSerialNumber indentifies a
            -- certificate (sent by the client) with an invalid
            -- signature.

KRB5PrincipalName ::= SEQUENCE {
```





```
    realm                   [0] Realm,
    principalName           [1] PrincipalName
}

AD-INITIAL-VERIFIED-CAS ::= SEQUENCE OF TrustedCA
            -- Identifies the certification path based on which
            -- the client certificate was validated.
            -- Each TrustedCA identifies a CA or a CA
            -- certificate (thereby its public key).

PA-PK-AS-REP ::= CHOICE {
   dhInfo                   [0] DHRepInfo,
            -- Selected when Diffie-Hellman key exchange is
            -- used.
   encKeyPack               [1] IMPLICIT OCTET STRING,
            -- Selected when public key encryption is used.
            -- Contains a CMS type ContentInfo encoded
            -- according to [RFC3852].
            -- The contentType field of the type ContentInfo is
            -- id-envelopedData (1.2.840.113549.1.7.3).
            -- The content field is an EnvelopedData.
            -- The contentType field for the type EnvelopedData
            -- is id-signedData (1.2.840.113549.1.7.2).
            -- The eContentType field for the inner type
            -- SignedData (when unencrypted) is id-pkrkeydata
```

```
                -- (1.2.840.113549.1.7.3) and the eContent field
                -- contains the DER encoding of the type
                -- ReplyKeyPack.
                -- ReplyKeyPack is defined below.
        ...
    }

    DHRepInfo ::= SEQUENCE {
        dhSignedData            [0] IMPLICIT OCTET STRING,
                -- Contains a CMS type ContentInfo encoded according
                -- to [RFC3852].
                -- The contentType field of the type ContentInfo is
                -- id-signedData (1.2.840.113549.1.7.2), and the
                -- content field is a SignedData.
                -- The eContentType field for the type SignedData is
                -- id-pkdhkeydata (1.3.6.1.5.2.3.2), and the
                -- eContent field contains the DER encoding of the
                -- type KDCDHKeyInfo.
                -- KDCDHKeyInfo is defined below.
        serverDHNonce           [1] DHNonce OPTIONAL
                -- Present if and only if dhKeyExpiration is
                -- present.
    }
```





```
    KDCDHKeyInfo ::= SEQUENCE {
        subjectPublicKey        [0] BIT STRING,
                -- KDC's DH public key.
                -- The DH pubic key value is mapped to a BIT STRING
                -- according to [RFC3279].
        nonce                   [1] INTEGER (0..4294967295),
                -- Contains the nonce in the PKAuthenticator of the
                -- request if DH keys are NOT reused,
                -- 0 otherwise.
        dhKeyExpiration         [2] KerberosTime OPTIONAL,
                -- Expiration time for KDC's key pair,
                -- present if and only if DH keys are reused. If
                -- this field is omitted then the serverDHNonce
                -- field MUST also be omitted.
        ...
    }

    ReplyKeyPack ::= SEQUENCE {
```

```
    replyKey                [0] EncryptionKey,
                -- Contains the session key used to encrypt the
                -- enc-part field in the AS-REP.
    nonce                   [1] INTEGER (0..4294967295),
                -- Contains the nonce in the PKAuthenticator of the
                -- request.
    ...
}

TD-DH-PARAMETERS ::= SEQUENCE OF DHDomainParameters
                -- Contains a list of Diffie-Hellman domain
                -- parameters in decreasing preference order.

DHDomainParameters ::= CHOICE {
    modp                    [0] DomainParameters,
                -- Type DomainParameters is defined in [RFC3279].
    ec                      [1] EcpkParameters,
                -- Type EcpkParameters is defined in [RFC3279].
    ...
}
END
```





## Intellectual Property Statement

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

## Disclaimer of Validity



## Copyright Statement



## Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.