Groupe de travail Réseau
Request for Comments : 4556
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

L. Zhu, Microsoft Corporation
B. Tung, Aerospace Corporation

juin 2006

Cryptographie à clé publique pour authentification initiale dans Kerberos (PKINIT)

Résumé
Le présent document décrit les extensions de protocole (ci après appelées PKINIT) à la spécification du protocole Kerberos. Ces extensions fournissent une méthode pour intégrer la cryptographie à clé publique dans l'échange initial d'authentification, en utilisant une signature à clé asymétrique et/ou des algorithmes de chiffrement dans les champs de données de pré-authentification.

Table des matières

1. Introduction

Le protocole Kerberos v5 [RFC4120] implique l'utilisation d'un tiers de confiance appelé centre de distribution de clé (KDC, *Key Distribution Center*) pour négocier les clés de session partagées entre les clients et les services et assure une authentification mutuelle entre eux.

Les pierres angulaires de Kerberos v5 sont le ticket et l'authentifiant. Un ticket encapsule une clé symétrique (la clé de session de ticket) dans une enveloppe (un message public) destinée à un service spécifique. Le contenu du ticket est chiffré avec une clé symétrique partagée entre le service principal et le KDC producteur. La partie chiffrée du ticket contient le nom principal du client, entre autres choses. Un authentifiant est un enregistrement qui peut être montré comme ayant été généré récemment en utilisant la clé de session de ticket dans le ticket associé. La clé de session de ticket est connue du client qui a demandé le ticket. Le contenu de l'authentifiant est chiffré avec la clé de session de ticket associée. La partie chiffrée d'un authentifiant contient un horodatage et le nom principal du client, entre autres choses.

Comme le montre la Figure 1 ci-dessous, le protocole Kerberos V5 consiste en les échanges de messages suivants entre le client et le KDC, et entre le client et le service d'application :

- Échange de service d'authentification (AS, *Authentication Service*)

Le client obtient un ticket "initial" du serveur d'authentification Kerberos, normalement un ticket distributeur de tickets (TGT, *Ticket Granting Ticket*). Le message AS-REQ et le message AS-REP sont respectivement le message de demande et le message de réponse, entre le client et l'AS.

- Échange de service de distribution de ticket (TGS, *Ticket Granting Service*)

Le client utilise ensuite le TGT pour s'authentifier et demander un ticket de service pour un service particulier, à partir du serveur de distribution de tickets Kerberos. Le message TGS-REQ et le message TGS-REP sont respectivement le message de demande et le message de réponse entre le client et le TGS.

- Échange client/serveur de protocole d'authentification (AP)

Le client fait ensuite une demande avec un message AP-REQ, consistant en un ticket de service et un authentifiant qui certifie la possession par le client de la clé de session de ticket. Le serveur peut facultativement répondre avec un message AP-REP. Les échanges AP négocient normalement des clés symétriques spécifiques de la session.

Généralement, l'AS et le TGS sont intégrés dans un seul appareil appelé le KDC.

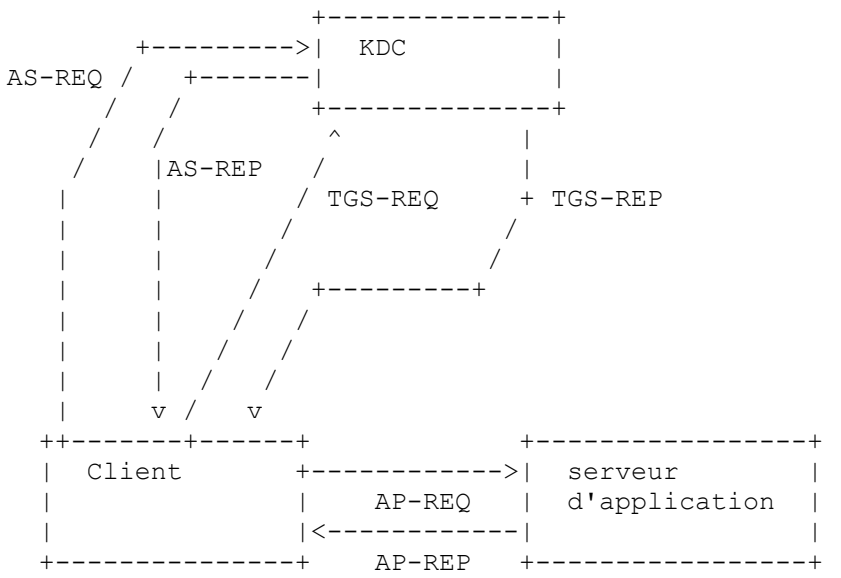

Figure 1 : échanges de messages dans le protocole Kerberos v5

Dans l'échange d'AS, la réponse de KDC contient la clé de session de ticket, entre autres choses, qui est chiffrée en utilisant une clé (la clé de réponse d'AS) partagée entre le client et le KDC. La clé de réponse d'AS est normalement déduite du mot de passe du client pour l'utilisateur humain. Donc, pour l'utilisateur humain, la force de résistance aux attaques du protocole Kerberos est n'est pas supérieure à la force de leurs mots de passe.

L'utilisation d'une cryptographie asymétrique sous la forme de certificats X.509 [RFC3280] est courante pour faciliter l'authentification de l'origine des données et le secret parfait. Une infrastructure de clé publique (PKI, *Public Key Infrastructure*) établie fournit des mécanismes de gestion de clé et de distribution de clés qui peuvent être utilisés pour établir l'authentification et sécuriser la communication. L'ajout de la cryptographie à clé publique à Kerberos apporte une certaine congruence aux protocoles de clé publique, allège pour l'utilisateur humain le fardeau de la gestion de mots de passe forts, et permet aux applications kerbérisées de tirer parti des services de clé et de gestion d'identité existants.

L'avantage du TGT Kerberos est que le client expose une seule fois ses secrets à long terme. Le TGT et sa clé de session associée peuvent alors être utilisés pour toute demande ultérieure de ticket de service. Il en résulte que toute authentification ultérieure est indépendante de la méthode par laquelle a été effectuée l'authentification initiale. Par conséquent, l'authentification initiale fournit un endroit convenable pour intégrer la cryptographie à clé publique dans l'authentification Kerberos. De plus, l'utilisation de la cryptographie symétrique après l'échange initial est préférable pour les performances.

Le présent document décrit les méthodes et formats de données que le client et le KDC peuvent utiliser dans les paires de clés publique et privée pour s'authentifier mutuellement dans l'échange d'AS et négocier la clé de réponse d'AS, connue seulement du client et du KDC, pour chiffrer la AS-REP envoyée par le KDC.

2. Conventions utilisées dans ce document

Les mots-clés "DOIT", "NE DOIT PAS", "EXIGÉ", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDÉ", "PEUT", et "FACULTATIF" dans le présent document sont à interpréter comme décrit dans la [RFC2119].

Dans ce protocole, le client et le KDC ont une paire de clés publique-privée afin de prouver leur identité l'un à l'autre sur le réseau ouvert. Le terme de "clé de signature" est utilisé pour se référer à la clé privée de la paire de clés utilisée.

La clé de chiffrement utilisée pour chiffrer le champ enc-part de la KDC-REP dans la AS-REP [RFC4120] est appelée la clé de réponse d'AS.

Une séquence vide dans le champ facultatif peut être incluse ou omise : les deux codages sont permis et considérés équivalents.

Le terme "Diffie-Hellman modulaire exponentiel" est utilisé pour se référer à l'échange de clé Diffie-Hellman, comme décrit dans la [RFC2631], afin de le différencier des autres représentations équivalentes du même algorithme d'accord de clés.

3. Extensions

Cette section décrit les extensions à la [RFC4120] pour prendre en charge l'utilisation de la cryptographie à clé publique dans la demande initiale de ticket.

En bref, le présent document définit les extensions suivantes à la [RFC4120] :

1. Le client indique l'utilisation de l'authentification par clé publique en incluant un pré authentifiant spécial dans la demande initiale. Ce pré authentifiant contient les données de clé publique du client et une signature.

2. Le KDC vérifie la demande du client par rapport à sa politique d'authentification et les autorités de certification (CA, *Certification Authorities*) de confiance.

3. Si la demande passe les essais de vérification, les réponses de KDC sont celles habituelles, mais la réponse est chiffrée en utilisant :
   a. soit une clé générée par un échange de clé Diffie-Hellman (DH) [RFC2631] [IEEE1363] avec le client, signée en utilisant la clé de signature du KDC ;
   b. soit une clé de chiffrement symétrique, signée en utilisant la clé de signature du KDC et chiffrée en utilisant la clé publique du client.

Tout le matériel de chiffrement requis par le client pour obtenir la clé de chiffrement pour déchiffrer la réponse du KDC est retourné dans un champ de pré authentification qui accompagne la réponse usuelle.

4. Le client valide la signature du KDC, obtient la clé de chiffrement, déchiffre la réponse, et procède ensuite comme d'habitude.

Le paragraphe 3.1 du présent document énumère les algorithmes requis et les types de message d'extension nécessaires. Le paragraphe 3.2 décrit les messages d'extension plus en détails.

3.1 Définitions, exigences, et constantes

3.1.1 Algorithmes exigés

Toutes les mises en œuvre de PIKNIT DOIVENT prendre en charge les algorithmes suivants :
- types de chiffrement de clé de réponse d'AS : aes128-cts-hmac-sha1-96 et aes256-cts-hmac-sha1-96 [RFC3962].
- algorithme de signature : sha-1WithRSAEncryption [RFC3370].
- méthode de livraison de clé de réponse d'AS : méthode de livraison de clé Diffie-Hellman, comme décrite au paragraphe 3.2.3.1.

De plus, les mises en œuvre de la présente spécification DOIVENT être capables de traiter l'extension d'usage de clé étendu (EKU, *Extended Key Usage*) et le otherName de id-pkinit-san (comme défini au paragraphe 3.2.2) de l'extension de nom de remplacement de sujet (SAN, *Subject Alternative Name*) dans les certificats X.509 [RFC3280].

3.1.2 Algorithmes recommandés

Toutes les mises en œuvre de PIKNIT DEVRAIENT prendre en charge l'algorithme suivant :

o méthode de livraison de clé de réponse d'AS : méthode de livraison de clé de chiffrement de clé publique, comme décrite au paragraphe 3.2.3.2.

Pour les mises en œuvre qui prennent en charge la méthode de livraison de clé de chiffrement de clé publique, les algorithmes suivants DOIVENT être pris en charge :

a) Les algorithme de transport de clés identifiés dans le champ keyEncryptionAlgorithm de type KeyTransRecipientInfo [RFC3852] pour chiffrer la clé temporaire dans le champ encryptedKey [RFC3852] avec une clé publique, comme décrit au paragraphe 3.2.3.2 : rsaEncryption (c'est le schéma de chiffrement RSAES-PKCS1-v1_5) [RFC3370] [RFC3447].

b) les algorithmes de chiffrement de contenu identifiés dans le champ contentEncryptionAlgorithm du type EncryptedContentInfo [RFC3852] pour chiffrer la clé de réponse d'AS avec la clé temporaire contenue dans le champ encryptedKey du type KeyTransRecipientInfo [RFC3852], comme décrit au paragraphe 3.2.3.2 : des-ede3-cbc (3DES à trois clés, en mode CBC) [RFC3370].

3.1.3 Types de message et de chiffrement définis

PKINIT utilise les nouveaux types de pré authentification suivants :

PA_PK_AS_REQ : 16
PA_PK_AS_REP : 17

PKINIT utilise aussi le nouveau type de données d'autorisation suivant :

AD_INITIAL_VERIFIED_CAS : 9

PKINIT introduit les nouveaux codes d'erreur suivants :

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 : 77
KDC_ERR_DIGEST_IN_CERT_NOT_ACCEPTED : 78
KDC_ERR_PA_CHECKSUM_DOIT_BE_INCLUDED : 79
KDC_ERR_DIGEST_IN_SIGNED_DATA_NOT_ACCEPTED : 80
KDC_ERR_PUBLIC_KEY_ENCRYPTION_NOT_SUPPORTED : 81

PKINIT utilise les types de données typées suivants pour les erreurs :

TD_TRUSTED_CERTIFIERS : 104
TD_INVALID_CERTIFICATES : 105
TD_DH_PARAMETERS : 109

Le module ASN.1 pour toutes les structures définies dans le présent document (plus les déclarations IMPORTE pour toutes les structures importées) est donné à l'Appendice A.

Toutes les structures définies dans le présent document ou importées DOIVENT être codées en utilisant les règles de codage distinctif (DER, *Distinguished Encoding Rules*) [X680], [X690] (sauf notation contraire). Toutes les structures de données portées dans des CHAÎNES D'OCTETS DOIVENT être codées conformément aux règles spécifiées dans les spécifications qui

définissent chaque structure de données ; une référence à la spécification appropriée est fournie pour chaque structure de données.

Note d'interopérabilité : certaines mises en œuvre peuvent n'être pas capables de décoder les objets enveloppés de syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*) [RFC3852] codés avec BER ; précisément, elle peuvent n'être pas capables de décoder des codages de longueur indéfinie. Pour maximiser l'interopérabilité, les mises en œuvre DEVRAIENT coder les objets de CMS utilisés dans PKINIT avec DER.

3.1.4 Types de chiffrement Kerberos définis pour les identifiants d'algorithme de CMS

PKINIT définit les numéros de type de chiffrement Kerberos[RFC3961] suivants, qui peuvent être utilisés dans le champ etype du message AS-REQ [RFC4120] pour indiquer au KDC l'acceptation du client des algorithmes correspondants (incluant les algorithmes de transport de clé [RFC3370], les algorithmes de chiffrement de contenu [RFC3370], et les algorithmes de signature) à utiliser avec la syntaxe de message cryptographique (CMS) [RFC3852], [RFC3370].

Selon la [RFC4120], les types de chiffrement dans le champ etype sont dans l'ordre de de préférence décroissante du client. Noter qu'il n'y a pas de signification à l'ordre relatif entre deux types différents d'algorithmes : les algorithmes de transport de clé, les algorithmes de chiffrement de contenu, et les algorithmes de signature.

La présence de chacun de ces types de chiffrement dans le champ etype est équivalente à la présence de l'identifiant d'objet (OID, *Object Identifier*) de l'algorithme correspondant dans le champ supportedCMSTypes comme décrit au paragraphe 3.2.1. Et l'ordre de préférence exprimé dans le champ supportedCMSTypes va écraser l'ordre de préférence mentionné dans le champ etype.

Numéro	Nom de type de chiffrement Kerberos	OID d'algorithme correspondant
9	id-dsa-with-sha1-CmsOID	id-dsa-with-sha1 [RFC3370]
10	md5WithRSAEncryption-CmsOID	md5WithRSAEncryption [RFC3370]
11	sha-1WithRSAEncryption-CmsOID	sha-1WithRSAEncryption [RFC3370]
12	rc2-cbc-EnvOID	rc2-cbc [RFC3370]
13	rsaEncryption-EnvOID	rsaEncryption [RFC3447][RFC3370]
14	id-RSAES-OAEP-EnvOID	id-RSAES-OAEP [RFC3447], [RFC3560]
15	des-ede3-cbc-EnvOID	des-ede3-cbc [RFC3370]

Les numéros de type de chiffrement ci-dessus ne sont utilisés que pour indiquer la prise en charge de l'utilisation des algorithmes correspondants dans PKINIT ; ils ne correspondent pas aux types de chiffrement Kerberos réels [RFC3961] et NE DOIVENT PAS être utilisés dans le champ etype du type EncryptedData Kerberos [RFC4120]. La pratique d'allocation de numéros de type de chiffrement Kerberos pour indiquer la prise en charge des algorithmes de CMS est considérée comme déconseillée, et de nouveaux numéros ne devraient pas être alloués à cette fin. À la place le champ supportedCMSTypes devrait être utilisé pour identifier les algorithmes pris en charge par le client et l'ordre de préférence du client.

Cependant, pour maximiser l'interopérabilité, les clients PKINIT qui souhaitent indiquer au KDC la prise en charge d'un ou plusieurs des algorithmes mentionnés ci-dessus DEVRAIENT inclure le ou les numéros de type de chiffrement correspondant dans le champ etype de la AS-REQ.

3.2 Syntaxe et usage de la pré authentification PKINIT

Ce paragraphe définit la syntaxe et l'usage des divers champs de pré authentification employés par PKINIT.

3.2.1 Génération de la demande du client

La demande initiale d'authentification (AS-REQ) est envoyée selon la [RFC4120] ; de plus, un élément de données de pré authentification, dont le padata-type est PA_PK_AS_REQ et dont la valeur de padata contient le codage en DER du type PA-PK-AS-REQ, est inclus.

```
PA-PK-AS-REQ ::= SEQUENCE {
        signedAuthPack      [0] CHAINE D'OCTETS IMPLICITE,
-- Contient un type CMS ContentInfo codé selon la [RFC3852]. Le champ contentType du type ContentInfo est id-signedData
   (1.2.840.113549.1.7.2), et le champ Content est SignedData. Le champ eContentType pour le type SignedData est id-
   pkinit-authData (1.3.6.1.5.2.3.1), et le champ eContent contient le codage en DER du type AuthPack. AuthPack est défini
   plus loin.
        trustedCertifiers      [1] SEQUENCE DE ExternalPrincipalIdentifier FACULTATIF,
```

-- Contient une liste de CA, de confiance pour le client, qui peuvent être utilisées pour certifier le KDC. Chaque ExternalPrincipalIdentifier identifie une CA ou un certificat de CA (et donc sa clé publique). Les informations contenues dans les trustedCertifiers DEVRAIENT être utilisées par le KDC comme indications pour guider son choix d'une chaîne de certificats appropriée à retourner au client.
kdcPkId [2] CHAINE D'OCTETS IMPLICITE FACULTATIF,
-- Contient un type CMS SignerIdentifier codé conformément à la [RFC3852]. Identifie, si il est présent, une clé publique de KDC particulière que le client a déjà.
...
}

DHNonce ::= CHAINE D'OCTETS

ExternalPrincipalIdentifier ::= SEQUENCE {
subjectName [0] CHAINE D'OCTETS IMPLICITE FACULTATIF,
-- Contient un nom de type PKIX codé selon la [RFC3280]. Identifie le sujet du certificat par le nom de sujet distinctif. EXIGÉ quand un nom de sujet distinctif est présent dans le certificat.
issuerAndSerialNumber [1] CHAINE D'OCTETS IMPLICITE FACULTATIF,
-- Contient un type CMS IssuerAndSerialNumber codé selon la [RFC3852]. Identifie un certificat du sujet. EXIGÉ pour TD-INVALID-CERTIFICATES et TD-TRUSTED-CERTIFIERS.
subjectKeyIdentifier [2] CHAINE D'OCTETS IMPLICITE FACULTATIF,
-- Identifie la clé publique du sujet par un identifiant de clé. Quand un certificat X.509 est référencé, cet identifiant de clé correspond à la valeur d'extension X.509 de subjectKeyIdentifier. Quand d'autres formats de certificat sont référencés, les documents qui spécifient le format de certificat et son utilisation avec la CMS doivent inclure des détails sur la correspondance entre l'identifiant de clé et le champ approprié de certificat. RECOMMANDÉ pour TD-TRUSTED-CERTIFIERS.
...
}

AuthPack ::= SEQUENCE {
pkAuthenticator [0] PKAuthenticator,
clientPublicValue [1] SubjectPublicKeyInfo FACULTATIF,
-- Le type SubjectPublicKeyInfo est défini dans la [RFC3280]. Spécifie les paramètres de domaine Diffie-Hellman et la valeur de la clé publique du client [IEEE1363]. La valeur de la clé publique DH est codée comme une chaîne de bits conformément à la [RFC3279]. Ce champ n'est présent que si le client souhaite utiliser la méthode d'accord de clé Diffie-Hellman.
supportedCMSTypes [2] SEQUENCE DE AlgorithmIdentifier FACULTATIF,
-- Le type AlgorithmIdentifier est défini dans la [RFC3280]. Il fait la liste des identifiants d'algorithme de CMS [RFC3370] qui identifient les algorithmes de transport de clé, ou les algorithmes de chiffrement de contenu, ou les algorithmes de signature pris en charge par le client en ordre de préférence décroissante.
clientDHNonce [3] DHNonce FACULTATIF,
-- N'est présent que si le client indique qu'il souhaite réutiliser les clés DH ou permettre au KDC de le faire (voir le paragraphe 3.2.3.1).
...
}

PKAuthenticator ::= SEQUENCE {
cusec [0] ENTIER (0..999999),
ctime [1] KerberosTime,
-- cusec et ctime sont utilisés comme dans la [RFC4120], pour la prévention des répétitions.
nonce [2] ENTIER (0..4294967295),
-- Choisi au hasard ; ce nom occasionnel n'a pas besoin de correspondre au nom occasionnel dans le KDC-REQ-BODY.
paChecksum [3] CHAINE D'OCTETS FACULTATIF,
-- DOIT être présent. Contient la somme de contrôle SHA1, effectuée sur KDC-REQ-BODY.
...
}

La structure ContentInfo [RFC3852] contenue dans le champ signedAuthPack du type PA-PK-AS-REQ est codée conformément à la [RFC3852] et est remplie comme suit :

1. Le champ contentType de type ContentInfo est id-signedData (comme défini dans la [RFC3852]) et le champ "content" est SignedData (comme défini dans la [RFC3852]).

2. Le champ eContentType pour le type SignedData est id-pkinit-authData : { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) authData(1) }. Note pour les mises en œuvre de CMS : l'attribut signé content-type DOIT être présent dans cette instance de SignedData, et sa valeur est id-pkinit-authData conformément à la [RFC3852].

3. Le champ eContent pour le type SignedData contient le codage en DER du type AuthPack.

4. Le champ signerInfos du type SignedData contient une seule signerInfo, qui contient la signature sur le type AuthPack.

5. La structure AuthPack contient un PKAuthenticator, les informations de clé publique du client, les types de chiffrement de CMS pris en charge par le client, et un DHNonce. Le champ pkAuthenticator certifie au KDC que le client a une connaissance récente de la clé de signature qui authentifie le client. Le champ clientPublicValue spécifie les paramètres de domaine Diffie-Hellman et la valeur de la clé publique du client. La valeur de la clé publique DH est codée comme CHAINE DE BITS conformément à la [RFC3279]. Le champ clientPublicValue n'est présent que si le client souhaite utiliser la méthode d'accord de clé Diffie-Hellman. Le champ supportedCMSTypes spécifie la liste des identifiants d'algorithme de CMS qui sont pris en charge par le client en ordre de préférence décroissante et peuvent être utilisés pour identifier un algorithme de signature ou un algorithme de transport de clés [RFC3370] dans le champ keyEncryptionAlgorithm du type KeyTransRecipientInfo, ou un algorithme de chiffrement de contenu [RFC3370] dans le champ contentEncryptionAlgorithm de type EncryptedContentInfo [RFC3852] lors du chiffrement de clé de réponse d'AS comme décrit au paragraphe 3.2.3.2. Cependant, il n'y a pas de signification à l'ordre relatif entre deux différents types d'algorithmes : algorithmes de transport de clés, algorithmes de chiffrement de contenu, et algorithmes de signature. Le champ clientDHNonce est décrit plus loin dans ce paragraphe.

6. Le champ ctime dans la structure PKAuthenticator contient l'heure courante chez l'hôte du client, et le champ cusec contient la partie microsecondes de l'horodatage du client. Les champs ctime et cusec sont utilisés ensemble pour spécifier un horodatage raisonnablement précis [RFC4120]. Le champ nonce est choisi au hasard. Le champ paChecksum DOIT être présent et il contient une somme de contrôle SHA1 qui est effectuée sur le KDC-REQ-BODY [RFC4120]. Afin de faciliter une future migration de l'utilisation de SHA1, le champ paChecksum est rendu syntaxiquement facultatif : quand la demande est étendue pour négocier des algorithmes de hachage, le nouveau client qui ne souhaite pas utiliser SHA1 va envoyer la demande dans la syntaxe étendue de message sans le champ paChecksum. Le KDC conforme à la présente spécification DOIT retourner un message KRB-ERROR [RFC4120] avec le code KDC_ERR_PA_CHECKSUM_MUST_BE_INCLUDED (voir au paragraphe 3.2.3). Cela va permettre à un nouveau client de réessayer avec SHA1 si c'est permis par la politique locale.

7. Le champ certificates du type SignedData contient les certificats destinés à faciliter la construction du chemin de certification, afin que le KDC puisse vérifier la signature sur le type AuthPack. Pour la validation du chemin, ces certificats DEVRAIENT être suffisants pour construire au moins un chemin de certification depuis le certificat du client jusqu'à une ancre de confiance acceptable pour le KDC [RFC4158]. Le client DOIT être capable d'inclure un tel ensemble de certificats si il est configuré à le faire. Le champ certificates NE DOIT PAS contenir de certificats de CA "racine".

8. La valeur publique Diffie-Hellman du client (clientPublicValue) est incluse si et seulement si le client souhaite utiliser la méthode d'accord de clé Diffie-Hellman. Les paramètres de domaine Diffie-Hellman [IEEE1363] pour la clé publique du client sont spécifiés dans le champ algorithm du type SubjectPublicKeyInfo [RFC3279], et la valeur de clé publique Diffie-Hellman du client est transposée en une subjectPublicKey (une CHAINE DE BITS) conformément à la [RFC3279]. Quand elles utilisent la méthode d'accord de clé Diffie-Hellman, les mises en œuvre DOIVENT prendre en charge le groupe 2 Oakley bien connu de1024 bits modulaire exponentiel (MODP) [RFC2412] et le groupe 14 Oakley bien connu de 2048 bits MODP [RFC3526] et DEVRAIENT prendre en charge le groupe 16 Oakley bien connu de 4096 bits MODP [RFC3526].

   La taille du champ Diffie-Hellman devrait être choisie de façon à fournir une sécurité cryptographique suffisante [RFC3766].

   Quand Diffie-Hellman MODP est utilisé, les exposants devraient avoir au moins deux fois autant de bits que les clés symétriques qui vont en être déduites [ODL99].

9. Le client peut souhaiter réutiliser des clés DH ou permettre au KDC de le faire (voir le paragraphe 3.2.3.1). Si il en est ainsi, le client va alors inclure le champ clientDHNonce. Cette chaîne de nom occasionnel DOIT être aussi longue que la plus grande longueur de clé des types de clé symétrique que le client accepte. Ce nom occasionnel DOIT être choisi au hasard.

La structure ExternalPrincipalIdentifier est utilisée dans le présent document pour identifier la clé publique du sujet devenant par là le principal du sujet. Cette structure est remplie comme suit :

1. Le champ subjectName contient un nom de type PKIX codé conformément à la [RFC3280]. Ce champ identifie le sujet du certificat par le nom de sujet distingué. Ce champ est EXIGÉ quand un nom de sujet distingué est présent dans le certificat utilisé.

2. Le champ issuerAndSerialNumber contient un type CMS IssuerAndSerialNumber codé conformément à la [RFC3852]. Ce champ identifie un certificat du sujet. Ce champ est EXIGÉ pour TD-INVALID-CERTIFICATES et TD-TRUSTED-CERTIFIERS (les deux structures sont définies au paragraphe 3.2.2).

3. Le champ subjectKeyIdentifier [RFC3852] identifie la clé publique du sujet par un identifiant de clé. Quand un certificat X.509 est référencé, cet identifiant de clé correspond à la valeur d'extension X.509 subjectKeyIdentifier. Quand d'autres formats de certificat sont référencés, les documents qui spécifient le format de certificat et leur utilisation avec la CMS doivent inclure des détails sur la correspondance de l'identifiant de clé avec le champ de certificat approprié . Ce champ est RECOMMANDÉ pour TD-TRUSTED-CERTIFIERS (comme défini au paragraphe 3.2.2).

Le champ trustedCertifiers du type PA-PK-AS-REQ contient une liste des autorités de certification, de confiance pour le client, qui peuvent être utilisées pour certifier le KDC. Chaque ExternalPrincipalIdentifier identifie une CA ou un certificat de CA (et donc sa clé publique).

Le champ kdcPkId du type PA-PK-AS-REQ contient un type CMS SignerIdentifier codé conformément à la [RFC3852]. Ce champ identifie, si il est présent, une clé publique particulière de KDC que le client a déjà.

3.2.2 Réception de la demande du client

À réception de la demande du client, le KDC la valide. Ce paragraphe décrit les étapes que le KDC DOIT (sauf mention contraire) parcourir pour valider la demande.

Le KDC vérifie la signature du client dans le champ signedAuthPack conformément à la [RFC3852].

Si, lors de la validation du certificat X.509 du client [RFC3280], le KDC ne peut pas construire un chemin de certification pour valider le certificat du client, il renvoie un message KRB-ERROR [RFC4120] avec le code KDC_ERR_CANT_VERIFY_CERTIFICATE. Les e-data d'accompagnement de ce message d'erreur sont un TYPED-DATA (comme défini dans la [RFC4120]) qui contient un élément dont le type de données est TD_TRUSTED_CERTIFIERS, et dont la valeur des données contient le codage en DER du type TD-TRUSTED-CERTIFIERS :

```
TD-TRUSTED-CERTIFIERS ::= SEQUENCE DE ExternalPrincipalIdentifier
-- Identifie une liste de CA de confiance pour le KDC. Chaque ExternalPrincipalIdentifier identifie une CA ou un certificat de
   CA (et donc sa clé publique).
```

Chaque ExternalPrincipalIdentifier (comme défini au paragraphe 3.2.1) dans la structure TD-TRUSTED-CERTIFIERS identifie une CA ou certificat de CA (et donc sa clé publique) de confiance pour le KDC.

À réception de ce message d'erreur, le client ne DEVRAIT réessayer que si il a un ensemble différent de certificats (provenant de ses demandes précédentes) qui forme un chemin de certification (ou chemin partiel) à partir des ancres de confiance acceptables par le KDC jusqu'à son propre certificat.

Si, lors du traitement du chemin de certification, le KDC détermine que la signature sur un des certificats dans le champ signedAuthPack est invalide, il retourne un message KRB-ERROR [RFC4120] avec le code KDC_ERR_INVALID_CERTIFICATE. Les e-data d'accompagnement de ce message d'erreur sont un TYPED-DATA qui contient un élément dont le data-type est TD_INVALID_CERTIFICATES, et dont la data-value contient le codage en DER du type TD-INVALID-CERTIFICATES :

```
TD-INVALID-CERTIFICATES ::= SEQUENCE DE ExternalPrincipalIdentifier
-- Chaque ExternalPrincipalIdentifier identifie un certificat (envoyé par le client) avec une signature invalide.
```

Chaque ExternalPrincipalIdentifier (comme défini au paragraphe 3.2.1) dans la structure TD-INVALID-CERTIFICATES identifie un certificat (qui a été envoyé par le client) avec une signature invalide.

Si plus d'une signature de certificat X.509 est invalide, le KDC PEUT inclure un IssuerAndSerialNumber par signature invalide dans le TD-INVALID-CERTIFICATES.

Le certificat X.509 du client est validé conformément à la [RFC3280].

Selon sa politique locale, le KDC peut aussi vérifier si des certificats X.509 dans le chemin de certification qui valide le certificat du client ont été révoqués. Si l'un d'eux a été révoqué, le KDC DOIT retourner un message d'erreur avec le code KDC_ERR_REVOKED_CERTIFICATE ; si le KDC tente de déterminer l'état de révocation mais est dans l'incapacité de le faire, il DEVRAIT retourner un message d'erreur avec le code KDC_ERR_REVOCATION_STATUS_UNKNOWN. Le ou les certificats affectés sont identifiés exactement comme pour le code d'erreur KDC_ERR_INVALID_CERTIFICATE (voir ci-dessus).

Noter que les données d'erreur TD_INVALID_CERTIFICATES ne sont utilisées que pour identifier des certificats invalides envoyés par le client dans la demande.

La clé publique du client est alors utilisée pour vérifier la signature. Si la vérification de signature échoue, le KDC DOIT retourner un message d'erreur avec le code KDC_ERR_INVALID_SIG. Il n'y a pas de e-data d'accompagnement pour ce message d'erreur.

En plus de la validation de la signature du client, le KDC DOIT aussi vérifier que la clé publique du client utilisée pour vérifier la signature du client est liée au nom principal du client spécifié dans la AS-REQ comme suit :

1. Si le KDC a son propre lien entre la clé publique de signature-vérification du client ou le certificat du client et le nom principal Kerberos du client, il utilise ce lien.

2. Autrement, si le certificat X.509 du client contient une extension "nom de remplacement de sujet" (SAN, *Subject Alternative Name*) portant un KRB5PrincipalName (défini ci-dessous) dans le champ otherName du type GeneralName [RFC3280], il lie le certificat X.509 du client à ce nom.

Le type du champ otherName est AnotherName. Le champ type-id du type AnotherName est id-pkinit-san :

id-pkinit-san IDENTIFIANT D'OBJET ::= { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) x509SanAN (2) }

Et le champ Valeur du type AnotherName est un KRB5PrincipalName.

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

Si le nom de client Kerberos dans la AS-REQ ne correspond pas à un nom lié par le KDC (le lien peut être dans le certificat, par exemple, comme décrit ci-dessus) ou si il n'est pas trouvé de lien par le KDC, le KDC DOIT retourner un message d'erreur avec le code KDC_ERR_CLIENT_NAME_MISMATCH. Il n'y a pas de e-data d'accompagnement pour ce message d'erreur.

Même si le chemin de certification est validé et si le certificat est transposé en le nom principal du client, le KDC peut décider de ne pas accepter le certificat du client, selon sa politique locale.

Le KDC PEUT exiger la présence d'un KeyPurposeId [RFC3280] d'usage de clé étendu (EKU, *Extended Key Usage*) id-pkinit-KPClientAuth dans le champ extensions du certificat X.509 du client :

```
id-pkinit-KPClientAuth IDENTIFIANT D'OBJET ::= iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3)
                                                keyPurposeClientAuth(4) }
-- Authentification du client PKINIT. Les bits d'usage de clé qui DOIVENT être cohérents : digitalSignature.
```

Le bit d'usage de clé digitalSignature [RFC3280] DOIT être établi quand l'intention du certificat X.509 du client est restreinte par l'EKU id-pkinit-KPClientAuth.

Si cet EKU KeyPurposeId est exigé mais n'est pas présent, ou si le certificat de client est restreint à ne pas être utilisé pour l'authentification de client PKINIT selon le paragraphe 4.2.1.13 de la [RFC3280], le KDC DOIT retourner un message d'erreur de code KDC_ERR_INCONSISTENT_KEY_PURPOSE. Il n'y a pas de e-data d'accompagnement pour ce message d'erreur. Les KDC qui mettent en œuvre cette exigence DEVRAIENT aussi accepter le EKU KeyPurposeId id-ms-kp-sc-logon (1.3.6.1.4.1.311.20.2.2) comme satisfaisant l'exigence, car il y a un grand nombre de certificats de client X.509 déployés pour être utilisés avec PKINIT qui ont cet EKU.

Au titre de la politique locale, le KDC PEUT décider de rejeter les demandes sur la base de l'absence ou présence d'autres OID spécifiques de EKU.

Si l'algorithme de résumé utilisé pour générer la signature de la CA pour la clé publique dans un certificat de la demande n'est pas acceptable par le KDC, le KDC DOIT retourner un message KRB-ERROR [RFC4120] avec le code KDC_ERR_DIGEST_IN_CERT_NOT_ACCEPTED. Les e-data d'accompagnement DOIVENT être codées en TYPED-DATA, bien qu'aucune ne soit définie pour le moment.

Si la clé publique du client n'est pas acceptée pour des raisons autres que celles spécifiées ci-dessus, le KDC retourne un message KRB-ERROR [RFC4120] avec le code KDC_ERR_CLIENT_NOT_TRUSTED. Il n'y a pas de e-data d'accompagnement actuellement définies pour ce message d'erreur.

Le KDC DOIT vérifier l'horodatage pour s'assurer que la demande n'est pas une répétition, et que le biais horaire tombe dans les limites acceptables. Les recommandations de biais d'horloge de la [RFC4120] s'appliquent ici. Si la vérification échoue, le KDC DOIT retourner le code d'erreur KRB_AP_ERR_REPEAT ou KRB_AP_ERR_SKEW, selon le cas.

Si la clientPublicValue est remplie, indiquant que le client souhaite utiliser la méthode d'accord de clé Diffie-Hellman, le KDC DEVRAIT vérifier si les paramètres de clé satisfont à sa politique. Si ce n'est pas le cas, il DOIT retourner un message d'erreur avec le code KDC_ERR_DH_KEY_PARAMETERS_NOT_ACCEPTED. Les e-data d'accompagnement sont des TYPED-DATA qui contiennent un élément dont le type de données est TD_DH_PARAMETERS, et dont la valeur de données contient le codage en DER du type TD-DH-PARAMETERS :

TD-DH-PARAMETERS ::= SEQUENCE DE AlgorithmIdentifier
-- Chaque AlgorithmIdentifier spécifie un ensemble de paramètres de domaine Diffie-Hellman [IEEE1363]. Cette liste est en ordre de préférence décroissante.

TD-DH-PARAMETERS contient une liste de paramètres de domaine Diffie-Hellman que le KDC prend en charge en ordre de préférence décroissante, à partir de laquelle le client DEVRAIT en prendre une pour réessayer la demande.

La structure de AlgorithmIdentifier est définie dans la [RFC3280] et est remplie conformément à la [RFC3279]. Plus précisément, le paragraphe 2.3.3 de la [RFC3279] décrit comment remplir la structure AlgorithmIdentifier dans le cas où l'échange de clé MODP Diffie-Hellman est utilisé.

Si le client a inclus un champ kdcPkId dans la PA-PK-AS-REQ et si le KDC ne possède pas la clé correspondante, le KDC DOIT ignorer le champ kdcPkId comme si le client n'en avait pas inclus.

Si l'algorithme de résumé utilisé par les id-pkinit-authData n'est pas acceptable pour le KDC, le KDC DOIT retourner un message KRB-ERROR [RFC4120] avec le code KDC_ERR_DIGEST_IN_SIGNED_DATA_NOT_ACCEPTED. Les e-data d'accompagnement DOIVENT être codées en TYPED-DATA, bien qu'aucune ne soit définie pour l'instant.

3.2.3 Génération de la réponse du KDC

Si la aChecksum remplie dans la demande n'est pas présente, le KDC qui se conforme à la présente spécification DOIT retourner un message KRB-ERROR [RFC4120] avec le code KDC_ERR_PA_CHECKSUM_MUST_BE_INCLUDED. Les e-data d'accompagnement DOIVENT être codées en TYPED-DATA (aucune donnée d'erreur n'est définie dans cette spécification).

En supposant que la demande du client a été correctement validée, le KDC procède comme d'après la [RFC4120], sauf comme suit.

Le KDC DOIT établir le fanion initial et inclure un élément de données d'autorisation de ad-type [RFC4120] AD_INITIAL_VERIFIED_CAS dans le ticket produit. Le champ ad-data [RFC4120] contient le codage en DER du type AD-INITIAL-VERIFIED-CAS :

AD-INITIAL-VERIFIED-CAS ::= SEQUENCE DE ExternalPrincipalIdentifier
-- Identifie le chemin de certification avec lequel le certificat de client a été validé. Chaque ExternalPrincipalIdentifier identifie une CA ou un certificat de CA (et donc sa clé publique).

La structure AD-INITIAL-VERIFIED-CAS identifie le chemin de certification avec lequel le certificat de client a été validé. Chaque ExternalPrincipalIdentifier (comme défini au paragraphe 3.2.1) dans la structure AD-INITIAL-VERIFIED-CAS identifie une CA ou un certificat de CA (et donc sa clé publique).

Noter que la syntaxe pour les données d'autorisation AD-INITIAL-VERIFIED-CAS permet de coder des SEQUENCE vides. De telles séquences vides ne peuvent être utilisées que si le KDC lui-même se porte garant du certificat de l'utilisateur.

L'AS enveloppe toutes les données de AD-INITIAL-VERIFIED-CAS dans des conteneurs AD-IF-RELEVANT si la liste des CA satisfait à la politique locale du domaine de l'AS (cela correspond au fanion de ticket TRANSITED-POLICY-CHECKED [RFC4120]). De plus, tout TGS DOIT copier de telles données d'autorisation des tickets utilisés au sein d'une PA-TGS-REQ de la TGS-REQ dans le ticket résultant. Si la liste des CA satisfait à la politique du domaine du KDC local, le TGS PEUT envelopper les données dans le conteneur AD-IF-RELEVANT ; autrement, il PEUT désenvelopper les données d'autorisation du conteneur AD-IF-RELEVANT.

Les serveurs d'application qui comprennent ce type de données d'autorisation DEVRAIENT appliquer leur politique locale pour déterminer si un certain ticket portant un tel type *non* contenu dans un conteneur AD-IF-RELEVANT est acceptable. (Cela correspond à la vérification par le serveur d'AP du champ de transit quand le fanion TRANSITED-POLICY-CHECKED n'a pas été établi [RFC4120].) Si un tel type de données est contenu dans un conteneur AD-IF-RELEVANT, les serveurs d'AP PEUVENT appliquer leur politique locale pour déterminer si les données d'autorisation sont acceptables.

Un élément de données de pré authentification, dont le padata-type est PA_PK_AS_REP et dont la padata-value contient le codage en DER du type PA-PK-AS-REP (défini ci-dessous) est inclus dans la AS-REP [RFC4120].

```
PA-PK-AS-REP ::= CHOIX {
        dhInfo              [0] DHRepInfo,
-- Choisi quand l'échange de clé Diffie-Hellman est utilisé.
        encKeyPack          [1] CHAINE D'OCTETS IMPLICITE,
-- Choisi quand le chiffrement à clé publique est utilisé. Contient un type CMS ContentInfo codé conformément à la
    [RFC3852]. Le champ contentType de type ContentInfo est id-envelopedData (1.2.840.113549.1.7.3). Le champ content
    est un EnvelopedData. Le champ contentType pour le type EnvelopedData est id-signedData (1.2.840.113549.1.7.2). Le
    champ eContentType pour le type interne SignedData (quand il n'est pas chiffré) est id-pkinit-rkeyData (1.3.6.1.5.2.3.3) et
    le champ eContent contient le codage en DER du type ReplyKeyPack. ReplyKeyPack est défini au paragraphe 3.2.3.2.
        ...
     }

DHRepInfo ::= SEQUENCE {
        dhSignedData        [0] CHAINE D'OCTETS IMPLICITE,
-- Contient un type CMS ContentInfo codé conformément à la [RFC3852]. Le champ contentType de type ContentInfo est id-
    signedData (1.2.840.113549.1.7.2), et le champ content est un SignedData. Le champ eContentType pour le type
    SignedData est id-pkinit-DHKeyData (1.3.6.1.5.2.3.2), et le champ eContent contient le codage en DER du type
    KDCDHKeyInfo. KDCDHKeyInfo est défini ci-dessous.
        serverDHNonce       [1] DHNonce FACULTATIF,
-- Présent si et seulement si dhKeyExpiration est présent dans le KDCDHKeyInfo.
        ...
     }

KDCDHKeyInfo ::= SEQUENCE {
        subjectPublicKey    [0] CHAINE DE BITS,
-- Clé publique DH du KDC. La valeur de la clé publique DH est codée comme CHAINE DE BITS conformément à la
    [RFC3279].
        nonce               [1] ENTIER (0 à 4 294 967 295),
 -- Contient le nom occasionnel dans le champ pkAuthenticator de la demande si les clés DH ne sont PAS réutilisées, 0
    autrement.
        dhKeyExpiration     [2] KerberosTime FACULTATIF,
-- Heure d'expiration de la paire de clés du KDC, présent si et seulement si les clés DH sont réutilisées. Si il est présent, la clé
    publique DH du KDC NE DOIT PAS être utilisée après l'heure d'expiration. Si ce champ est omis, le champ
    serverDHNonce DOIT alors être omis aussi.
        ...
     }
```

Le contenu de l'AS-REP est autrement inchangé par rapport à la [RFC4120]. Le KDC chiffre la réponse comme d'habitude, mais pas avec la clé à long terme du client. Il la chiffre plutôt avec une clé partagée déduite d'un échange Diffie-Hellman ou avec une clé de chiffrement générée. Le contenu de la PA-PK-AS-REP indique quelle méthode de livraison de clé est utilisée.

Si le client ne souhaite pas utiliser la méthode de livraison de clé Diffie-Hellman (le champ clientPublicValue n'est pas présent dans la demande) et si le KDC ne prend pas en charge la méthode de livraison de clé de chiffrement à clé publique, le KDC DOIT retourner un message d'erreur avec le code KDC_ERR_PUBLIC_KEY_ENCRYPTION_NOT_SUPPORTED. Il n'y a pas de e-data d'accompagnement pour ce message d'erreur.

De plus, la duré de vie du ticket retourné par le KDC NE DOIT PAS excéder celle de la paire de clés publique-privée du client. Cependant la durée de vie du ticket peut être plus courte que celle de la paire de clés publique-privée du client. Pour les mises en œuvre de la présente spécification, la durée de vie de la paire de clés publique-privée du client est la période de validité dans les certificats X.509 [RFC3280], sauf configurée autrement.

3.2.3.1 Utilisation de l'échange de clé Diffie-Hellman

Dans ce cas, la PA-PK-AS-REP contient une structure DHRepInfo.

La structure ContentInfo [RFC3852] pour le champ dhSignedData est remplie comme suit :

1. Le champ contentType de type ContentInfo est id-signedData (comme défini dans la [RFC3852]), et le champ content est une SignedData (comme défini dans la [RFC3852]).

2. Le champ eContentType pour le type SignedData est la valeur d'OID pour id-pkinit-DHKeyData: { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) DHKeyData(2) }. Note pour les mises en œuvre de CMS : le type de contenu d'attribut signé DOIT être présent dans cette instance de SignedData, et sa valeur est id-pkinit-DHKeyData conformément à la [RFC3852].

3. Le champ eContent pour le type SignedData contient le codage en DER du type KDCDHKeyInfo.

4. La structure KDCDHKeyInfo contient la clé publique du KDC, un nom occasionnel *(nonce)* et, facultativement, l'heure d'expiration de la clé DH du KDC réutilisée. Le champ subjectPublicKey de type KDCDHKeyInfo identifie la clé publique DH du KDC. Cette valeur de la clé publique DH est codée comme CHAINE DE BITS conformément à la [RFC3279]. Le champ nonce contient le nom occasionnel dans le champ pkAuthenticator de la demande si les clés DH ne sont PAS réutilisées. La valeur de ce nom occasionnel est 0 si les clés DH sont réutilisées. Le champ dhKeyExpiration n'est présent que si et seulement si les clés DH sont réutilisées. Si le champ dhKeyExpiration est présent, la clé publique du KDC dans cette structure KDCDHKeyInfo NE DOIT PAS être utilisé après cette heure d'expiration. Si ce champ est omis, le champ serverDHNonce DOIT alors être omis aussi.

5. Le champ signerInfos du type SignedData contient un seul signerInfo, qui contient la signature sur le type KDCDHKeyInfo.

6. Le champ certificates de type SignedData contient les certificats destinés à faciliter la construction du chemin de certification, afin que le client puisse vérifier la signature du KDC sur le type KDCDHKeyInfo. Les informations contenues dans le trustedCertifiers de la demande DEVRAIENT être utilisées par le KDC comme indications pour guider son choix d'une chaîne de certificats appropriée à retourner au client. Ce champ peut être laissé vide si la clé publique de KDC spécifiée par le champ kdcPkId dans la PA-PK-AS-REQ a été utilisée pour signer. Autrement, pour la validation de chemin, ces certificats DEVRAIENT être suffisants pour construire au moins un chemin de certification allant du certificat de KDC à une ancre de confiance acceptable par le client [RFC4158]. Le KDC DOIT être capable d'inclure un tel ensemble de certificats si il est configuré à le faire. Le champ certificats NE DOIT PAS contenir de certificat de CA "racine".

7. Si le client a inclus le champ clientDHNonce, le KDC peut alors choisir de réutiliser ses clés DH. Si le serveur réutilise ses clés DH, il DOIT alors inclure une heure d'expiration dans le champ dhKeyExpiration. Passée cette heure d'expiration, la signature sur le type DHRepInfo est considérée comme expirée/invalide. Quand le serveur réutilise des clés DH, il DOIT alors inclure un serverDHNonce au moins aussi long que les clés pour le système de chiffrement symétrique utilisé pour chiffrer la réponse d'AS. Noter qu'inclure le serverDHNonce change le calcul par le client et le serveur de la clé à utiliser pour chiffrer la réponse ; voir ci-dessous les détails. Le KDC NE DEVRAIT PAS réutiliser les clés DH à moins que le champ clientDHNonce soit présent dans la demande.

La clé de réponse d'AS est déduite comme suit :

1. Le KDC et le client calculent tous deux la valeur de secret partagé comme suit :
   a) Quand MODP Diffie-Hellman est utilisé, soit DHSharedSecret la valeur de secret partagé. DHSharedSecret est la valeur ZZ, comme décrit au paragraphe 2.1.1 de la [RFC2631].

      DHSharedSecret est d'abord bourré avec des zéros en tête afin que la taille de DHSharedSecret en octets soit la même que celle du module, puis représentée comme une chaîne d'octets en ordre gros boutien.

Note de mise en œuvre : le client et le KDC peuvent tous deux mettre en antémémoire le triplet {ya, yb, DHSharedSecret} où ya est la clé publique du client et yb est la clé publique du KDC. Si ya et yb sont tous deux les mêmes dans un échange ultérieur, le DHSharedSecret de l'antémémoire peut être utilisé.

2. Soit K la longueur du germe de génération de clé [RFC3961] de la clé de réponse d'AS dont le enctype est choisi conformément à la [RFC4120].

3. On définit la fonction octetstring2key() comme suit :

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

   où x est une chaîne d'octets ; | est l'opérateur d'enchaînement ; 0x00, 0x01, 0x02, etc. sont chacun représenté comme un seul octet ; random-to-key() est une opération qui génère une clé de protocole à partir d'une chaîne de bits de longueur K, et K-truncate tronque son entrée aux K premiers bits. K et random-to-key() sont tous deux comme défini dans le profil kcrypto [RFC3961] pour le enctype de la clé de réponse d'AS.

4. Quand les clés DH sont réutilisées, soit n_c le clientDHNonce et n_k le serverDHNonce ; autrement, soient n_c et n_k des chaînes d'octets vides.

5. La clé de réponse d'AS k est : k = octetstring2key(DHSharedSecret | n_c | n_k)

3.2.3.2 Utilisation du chiffrement à clé publique

Dans ce cas, la PA-PK-AS-REP contient le champ encKeyPack où la clé de réponse d'AS est chiffrée.

La structure ContentInfo [RFC3852] pour le champ encKeyPack est remplie comme suit :

1. Le champ contentType du type ContentInfo est id-envelopedData (comme défini dans la [RFC3852]), et le champ content est une EnvelopedData (comme défini dans la [RFC3852]).

2. Le champ contentType pour le type EnvelopedData est id-signedData : { iso (1) member-body (2) us (840) rsadsi (113549) pkcs (1) pkcs7 (7) signedData (2) }.

3. Le champ eContentType pour le type interne SignedData (quand il est déchiffré du champ encryptedContent pour le type EnvelopedData) est id-pkinit-rkeyData : { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) rkeyData(3) }. Note pour les mises en œuvre de CMS : le type de contenu d'attribut signé DOIT être présent dans cette instance de SignedData, et sa valeur est id-pkinit-rkeyData conformément à la [RFC3852].

4. Le champ eContent pour le type interne SignedData contient le codage en DER du type ReplyKeyPack (comme décrit ci-dessous).

5. Le champ signerInfos du type interne SignedData contient une seule signerInfo, qui contient la signature pour le type ReplyKeyPack.

6. Le champ certificates du type interne SignedData contient les certificats destinés à faciliter la construction du chemin de certification, afin que le client puisse vérifier la signature du KDC pour le type ReplyKeyPack. Les informations contenues dans le trustedCertifiers dans la demande DEVRAIENT être utilisées par le KDC comme indications pour guider son choix d'une chaîne de certificats appropriée à retourner au client. Ce champ peut être laissé vide si la clé publique du KDC spécifiée par le champ kdcPkId dans la PA-PK-AS-REQ a été utilisée pour signer. Autrement, pour la validation de chemin, ces certificats DEVRAIENT être suffisants pour construire au moins un chemin de certification du certificat de KDC à une ancre de confiance acceptable par le client [RFC4158]. Le KDC DOIT être capable d'inclure un tel ensemble de certificats si il est configuré à le faire. Le champ certificates NE DOIT PAS contenir de certificat de CA "racine".

7. Le champ recipientInfos du type EnvelopedData est un ensemble qui DOIT contenir exactement un membre du type KeyTransRecipientInfo. La encryptedKey de ce membre contient la clé temporaire qui est chiffrée en utilisant la clé publique du client.

8. Les champs unprotectedAttrs ou originatorInfo du type EnvelopedData PEUVENT être présents.

Si il y a un champ supportedCMSTypes dans le AuthPack, le KDC doit vérifier si il prend en charge un des types mentionnés. Si il prend en charge plus d'un des types, le KDC DEVRAIT utiliser celui qui figure en premier. Si il ne prend en charge aucun d'eux, il DOIT retourner un message d'erreur avec le code KDC_ERR_ETYPE_NOSUPP [RFC4120].

De plus, le KDC calcule la somme de contrôle de la AS-REQ dans la demande du client. Cette somme de contrôle est effectuée sur le type AS-REQ, et la clé de protocole [RFC3961] de l'opération de somme de contrôle est la replyKey, et le numéro d'usage de clé est 6. Si le type de chiffrement de la replyKey est "plus récent" [RFC4120], [RFC4121], l'opération de somme de contrôle est l'opération de somme de contrôle exigée [RFC3961] de ce enctype.

```
ReplyKeyPack ::= SEQUENCE {
        replyKey            [0] EncryptionKey,
-- Contient la clé de session utilisée pour chiffrer le champ enc-part dans la AS-REP, c'est-à-dire, la clé de réponse d'AS.
        asChecksum          [1] Checksum,
-- Contient la somme de contrôle de l'AS-REQ correspondant à l'AS-REP conteneuse. La somme de contrôle est effectuée sur
    le type AS-REQ. La clé de protocole [RFC3961] de la somme de contrôle est la replyKey et le numéro d'usage de clé est 6.
    Si le type de chiffrement de la replyKey est "plus récent" [RFC4120], [RFC4121], la somme de contrôle est l'opération de
    somme de contrôle exigée [RFC3961] pour ce enctype. Le client DOIT vérifier cette somme de contrôle dès la réception de
    l'AS-REP.
        ...
    }
```

Il est RECOMMANDÉ que les mises en œuvre de cette méthode de livraison de clé de chiffrement RSA prennent en charge des clés RSA d'au moins 2048 bits.

3.2.4 Réception de la réponse du KDC

À réception de la réponse du KDC, le client procède comme suit. Si la PA-PK-AS-REP contient le champ dhSignedData, le client déduit la clé de réponse d'AS en utilisant la même procédure qu'utilisée par le KDC, comme défini au paragraphe 3.2.3.1. Autrement, le message contient le champ encKeyPack, et le client déchiffre et extrait la clé temporaire dans le champ encryptedKey des KeyTransRecipientInfo membres et utilise ensuite cela comme clé de réponse d'AS.

Si la méthode de chiffrement à clé publique est utilisée, le client DOIT vérifier la asChecksum contenue dans le ReplyKeyPack.

Dans l'un et l'autre cas, le client DOIT vérifier la signature dans les SignedData conformément à la [RFC3852]. Le certificat X.509 du KDC DOIT être validé conformément à la [RFC3280]. De plus, sauf si le client peut vérifier autrement que la clé publique utilisée pour vérifier la signature du KDC est liée au KDC du domaine cible, le certificat X.509 du KDC DOIT contenir une extension SAN [RFC3280] portant un AnotherName dont le type-id est id-pkinit-san (comme défini au paragraphe 3.2.2) et dont la valeur est un KRB5PrincipalName qui correspond au nom du TGS du domaine cible (comme défini au paragraphe 7.3 de la [RFC4120]).

Sauf si le client connaît par d'autres moyens que le certificat de KDC est destiné à un KDC Kerberos, le client DOIT exiger que le certificat de KDC contienne le EKU KeyPurposeId [RFC3280] id-pkinit-KPKdc :

```
id-pkinit-KPKdc IDENTIFIANT D'OBJET ::= { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3)
                                          keyPurposeKdc(5) }
-- Réponses du KDC signataire. Bits d'usage de clé qui DOIVENT être cohérents : digitalSignature.
```

Le bit d'usage de clé digitalSignature [RFC3280] DOIT être établi quand l'objet du certificat X.509 du KDC est restreint par le EKU id-pkinit-KPKdc.

Si le certificat de KDC contient le nom de TGS Kerberos codé comme SAN id-pkinit-san, ce certificat est certifié par la CA productrice comme certificat de KDC, donc le EKU id-pkinit-KPKdc n'est pas exigé.

Si toutes les vérifications applicables sont satisfaites, le client déchiffre alors le champ enc-part de la KDC-REP dans la AS-REP, en utilisant la clé de réponse d'AS, et ensuite en poursuivant comme décrit dans la [RFC4120].

3.3 Exigences d'interopérabilité

Le client DOIT être capable d'envoyer un ensemble de certificats suffisant pour permettre au KDC de construire un chemin de certification pour le certificat du client, si l'ensemble correct de certificats est fourni par la configuration ou la politique.

Si le client envoie tous les certificats X.509 sur un chemin de certification à une ancre de confiance acceptable pour le KDC, et si le KDC ne peut pas vérifier la clé publique du client autrement, le KDC DOIT être capable de traiter la validation de chemin pour le certificat du client sur la base des certificats dans la demande.

Le KDC DOIT être capable d'envoyer un ensemble de certificats suffisant pour permettre au client de construire un chemin de certification pour le certificat du KDC, si l'ensemble correct de certificats est fourni par la configuration ou la politique.

Si le KDC envoie tous les certificats X.509 sur un chemin de certification à une ancre de confiance acceptable par le client, et si le client ne peut pas vérifier autrement la clé publique du KDC, le client DOIT être capable de traiter la validation de chemin pour le certificat du KDC sur la base des certificats de la réponse.

3.4 Indication par le KDC de la prise en charge de PKINIT

Si la pré authentification est exigée mais n'était pas présente dans la demande, selon la [RFC4120] un message d'erreur avec le code KDC_ERR_PREAUTH_FAILED est retourné, et un objet METHOD-DATA va être mémorisé dans le champ e-data du message KRB-ERROR pour spécifier quels mécanismes de pré authentification sont acceptables. Le KDC peut alors indiquer la prise en charge de PKINIT en incluant un élément vide dont le padata-type est PA_PK_AS_REQ dans cet objet METHOD-DATA.

Autrement, si il est exigé par la politique locale du KDC que le client doive être pré authentifié en utilisant le mécanisme de pré authentification spécifié dans le présent document, mais qu'aucune pré authentification PKINIT n'est présente dans la demande, un message d'erreur avec le code KDC_ERR_PREAUTH_FAILED DEVRAIT être retourné.

Les KDC DOIVENT laisser vide le champ padata-value de l'élément PA_PK_AS_REQ dans les METHOD-DATA du KRB-ERROR (c'est-à-dire, envoyer une CHAÎNE D'OCTETS de longueur zéro) et les clients DOIVENT ignorer cela et toute autre valeur. De futures extensions de ce protocole pourront spécifier d'autres données à envoyer à la place d'une CHAÎNE D'OCTETS vide.

4. Considérations pour la sécurité

La sécurité des algorithmes de chiffrement dépend de la quantité de secret générée [RFC4086]. Le nombre de bits vraiment aléatoires est extrêmement important pour déterminer la force de résistance aux attaques contre le système de chiffrement ; par exemple, les exposants secrets Diffie-Hellman doivent être choisis sur la base de bits vraiment aléatoires (où n est l'exigence de sécurité du système). La sécurité du système global est significativement affaiblie par l'utilisation d'entrées insuffisamment aléatoires : un attaquant sophistiqué peut trouver plus facile de reproduire l'environnement qui a produit les quantités secrètes et de chercher dans le petit ensemble de possibilités résultante que de localiser les quantités dans l'espace entier de nombres potentiels.

Les messages d'erreur Kerberos ne sont pas protégés en intégrité ; par suite, les paramètres de domaine envoyés par le KDC comme TD-DH-PARAMETERS peuvent être altérés par un attaquant de sorte que l'ensemble des paramètres de domaine choisi pourrait être plus faible ou non mutuellement préféré. La politique locale peut configurer des ensembles de paramètres de domaine acceptables localement, ou interdire la négociation des paramètres de domaine DH.

La taille de la clé symétrique de réponse et la taille de champ Diffie-Hellman ou la taille du module RSA devraient être choisies de façon à fournir une sécurité cryptographique suffisante [RFC3766].

Quand MODP Diffie-Hellman est utilisé, les exposants devraient avoir au moins deux fois autant de bits que les clés symétriques qui vont en être déduites [ODL99].

PKINIT soulève certains problèmes de sécurité au delà de ce qui peut être réglé strictement dans les définitions de protocole. On les aborde dans cette section.

PKINIT étend le modèle inter domaines à l'infrastructure de clé publique. Les utilisateurs de PKINIT doivent comprendre les politiques et procédures de sécurité appropriées à l'utilisation des infrastructures de clé publique [RFC3280].

Pour faire confiance à un certificat de KDC qui est certifié par une CA comme certificat de KDC pour un domaine cible (par exemple, en affirmant le nom de TGS de ce domaine Kerberos comme un SAN id-pkinit-san et/ou en restreignant l'usage du certificat par l'utilisation de l'EKU id-pkinit-KPKdc, comme décrit au paragraphe 3.2.4) le client DOIT vérifier que la CA qui produit le certificat de KDC est autorisée à produire des certificats de KDC pour ce domaine cible. Autrement, le lien entre le certificat de KDC et le KDC du domaine cible n'est pas établi.

Comment valider cette autorisation est une affaire de politique locale. Une façon de faire cela est la configuration d'ensembles spécifiques de CA et ancres de confiance intermédiaires, dont une doit être sur le chemin de certification du certificat de KDC [RFC3280], et, pour chaque CA ou ancre de confiance, sur les domaines pour lesquels elle est autorisée à produire des certificats.

De plus, si une CA qui est de confiance pour produire des certificats de KDC peut aussi produire d'autres sortes de certificats, la politique locale doit être capable de distinguer entre eux ; par exemple, elle pourrait exiger que les certificats de KDC contiennent le EKU id-pkinit-KPKdc ou que le domaine soit spécifié avec le SAN id-pkinit-san.

Il est de la responsabilité des administrateurs PKI d'une organisation de s'assurer que les certificats de KDC sont produits seulement pour les KDC, et que les clients peuvent s'en assurer en utilisant leur politique locale.

Kerberos standard donne la possibilité d'interactions entre cryptosystèmes de diverses forces ; le présent document ajoute à Kerberos des interactions avec les systèmes de chiffrement à clé publique. Certaines politiques administratives peuvent permettre l'utilisation de clés publiques relativement faibles. Quand on utilise de telles clés asymétriques faibles pour protéger/échanger des clés symétriques plus fortes, la force de résistance à l'attaque du système global n'est pas meilleure que celle de ces clés faibles [RFC3766].

PKINIT exige que les clés pour les systèmes de chiffrement symétrique soient générées. Certains de ces systèmes contiennent des clés "faibles". Pour des recommandations concernant ces clés faibles, voir la [RFC4120].

PKINIT permet l'utilisation de la même paire de clés RSA pour le chiffrement et la signature lors de la livraison de clés RSA fondée sur le chiffrement. Ce n'est pas l'usage recommandé des clés RSA [RFC3447] ; on évite cela en utilisant la livraison de clé fondée sur DH.

On devrait veiller à la façon dont les certificats sont choisis pour les besoins de l'authentification qui utilise PKINIT. Certaines politiques locales peuvent exiger qu'un tiers de confiance soit utilisé pour certains types de certificats. Les déploiements de PKINIT devraient être conscients des implications de l'utilisation de certificats qui ont des clés fournies par un tiers de confiance pour les besoins de l'authentification. Comme les certificats seulement de signature ne sont normalement pas fournis par un tiers de confiance, ceci est évité en utilisant la livraison de clé fondée sur DH.

PKINIT ne fournit pas d'essais "d'acheminement de retour" pour empêcher des attaquants de monter une attaque de déni de service sur le KDC en l'amenant à effectuer des opérations de clé publique inutiles et coûteuses. Strictement parlant, ceci est aussi vrai du Kerberos standard, bien que le coût potentiel soit faible, parce que le Kerberos standard n'utilise pas la cryptographie à clé publique. En utilisant la livraison de clés fondée sur DH et en réutilisant les clés DH, le coût nécessaire de traitement de chiffrement par demande peut être minimisé.

Quand la méthode d'échange de clé Diffie-Hellman est utilisée, des données de pré authentification supplémentaires [RFC4120] (en plus de la PA_PK_AS_REQ, comme défini dans la présente spécification) ne sont pas liées à la AS_REQ par les mécanismes discutés dans cette spécification (ce qui signifie qu'elles peuvent être abandonnées ou ajoutées par des attaquants sans être détectées par le client ou le KDC). Les concepteurs de données de pré authentification supplémentaires devraient prendre en considération si de telles données supplémentaires de pré authentification peuvent être utilisées en conjonction avec la PA_PK_AS_REQ. Les futurs travaux du groupe de travail Kerberos devraient mettre à jour les algorithmes de hachage spécifiés dans le présent document et fournir un mécanisme générique pour lier les données de pré authentification supplémentaires à l'AS_REQ qui les accompagne.

Le numéro d'usage de clé 6 utilisé par le champ asChecksum est aussi utilisé pour la somme de contrôle d'authentifiant (champ cksum de la AP-REQ) contenue dans les données de pré authentification de PA-TGS-REQ contenues dans une TGS-REQ [RFC4120]. Ce conflit est présent pour des raisons historiques ; la réutilisation des numéros d'usage de clé est fortement déconseillée.

5. Remerciements

Les personnes suivantes ont fait des contributions significatives au présent document : Paul Leach, Stefan Santesson, Sam Hartman, Love Hornquist Astrand, Ken Raeburn, Nicolas Williams, John Wray, Tom Yu, Jeffrey Hutzelman, David Cross, Dan Simon, Karthik Jaganathan, Chaskiel M Grundman, et Jeffrey Altman.

Andre Scedrov, Aaron D. Jaggard, Iliano Cervesato, Joe-Kai Tsay, et Chris Walstad ont découvert un problème de lien entre AS-REQ et AS- REP dans le projet n° 26. Il en est résulté l'ajout du champ asChecksum.

Des remerciements particuliers à Clifford Neuman, Matthew Hur, Ari Medvinsky, Sasha Medvinsky, et Jonathan Trostle qui ont rédigé les premières versions de ce document.

Les auteurs sont redevables au président du groupe de travail Kerberos, Jeffrey Hutzelman, qui a suivi divers problèmes et a été d'une aide considérable durant la création de ce document.

Certaines des idées qui fondent ce document sont apparues durant des discussions échelonnées sur plusieurs années entre les membres du SAAG, du groupe de travail CAT de l'IETF, et le PSRG, concernant l'intégration de Kerberos et SPX. Certaines idées ont aussi été tirées du système DASS. Ces changements ne sont aucunement reprises par ces groupes. C'est une tentative de faire revivre certains des objectifs de ces groupes, et le présent document envisage ces objectifs dans la seule perspective de Kerberos.

Enfin, les commentaires des groupes qui travaillent sur des idées similaires dans DCE ont été précieuses.

6. Références

6.1 Références normatives

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

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2412] H. Orman, "Protocole OAKLEY de détermination de clés", novembre 1998. *(Information)*

[RFC2631] E. Rescorla, "Méthode d'accord de clé Diffie-Hellman", juin 1999. *(P.S.)*

[RFC3279] L. Bassham, W. Polk et R. Housley, "Algorithmes et identifiants pour le profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002.

[RFC3280] R. Housley, W. Polk, W. Ford et D. Solo, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002. *(Obsolète, voir* RFC5280)

[RFC3370] R. Housley, "Algorithmes de syntaxe de message cryptographique (CMS)", août 2002. *(P.S. ; MàJ par* RFC8702*)*

[RFC3447] J. Jonsson et B. Kaliski, "Normes de cryptographie à clés publiques (PKCS) n° 1 : Spécifications de la cryptographie RSA version 2.1", février 2003. *(Obsolète, remplacée par* RFC8017*) (Information)*

[RFC3526] T. Kivinen et M. Kojo, "Groupes supplémentaires d'exponentiation modulaire (MODP) Diffie-Hellman pour l'échange de clés Internet (IKE)", mai 2003.

[RFC3560] R. Housley, "Utilisation de l'algorithme de transport de clé RSAES-OAEP dans la syntaxe de message cryptographique (CMS)", juillet 2003. *(P.S.)*

[RFC3766] H. Orman, P. Hoffman, "Détermination de la force des clés publiques utilisées pour l'échange de clés symétriques", avril 2004. (BCP0086)

[RFC3852] R. Housley, "Syntaxe de message cryptographique (CMS)", juillet 2004. *(Obsolète, voir la* RFC5652)

[RFC3961] K. Raeburn, "Spécifications de chiffrement et de somme de contrôle pour Kerberos 5", février 2005. *(MàJ par* 8429

[RFC3962] K. Raeburn, "Chiffrement de la norme de chiffrement évolué (AES) pour Kerberos 5", février 2005.

[RFC4086] D. Eastlake 3rd, J. Schiller, S. Crocker, "Exigences d'aléa pour la sécurité", juin 2005. (*Remplace* RFC1750) (BCP0106)

[RFC4120] C. Neuman et autres, "Service Kerberos d'authentification de réseau (V5)", juillet 2005. (*MàJ par* RFC4537, 5021, 6649, 7751, 8062, 8129, 8429)

[X680] Recommandation UIT-T X.680 | ISO/CEI 8824-1:2002, "Technologies de l'information - Notation de syntaxe abstraite numéro un (ASN.1) : spécification de la notation de base". (2002)

[X690] Recommandation UIT-T X.690 | ISO/CEI 8825-1:2002, " Technologies de l'information - règles de codages ASN.1 : spécification des règles de codage de base (BER), des règles de codages canoniques (CER) et des règles de codage distinctives (DER)". (2002)

6.2 Références pour information

[ODL99] Odlyzko, A., "Discrete logarithms: The past et the future, Designs, Codes, et Cryptography", avril 1999.

[RFC4121] L. Zhu et autres, "Version 2 du mécanisme d'interface de programme d'application de service de sécurité générique (GSS-API) de Kerberos version 5", juillet 2005. (*MàJ* RFC1964) (*MàJ par les* RFC6542, 6649, 8062)) *(P.S.)*

[RFC4158] M. Cooper et autres, "Infrastructure de clés publiques X.509 pour l'Internet : construction du chemin de certification", septembre 2005. *(Information)*

Appendice A. Module ASN.1 pour PKINIT

```
KerberosV5-PK-INIT-SPEC { iso(1) identified-organization(3) dod(6) internet(1) security(5) kerberosV5(2) modules(4) pkinit(5)}
DEFINITIONS DES ÉTIQUETTES EXPLICITES ::= DÉBUT
IMPORTE
  SubjectPublicKeyInfo, AlgorithmIdentifier
     DE PKIX1Explicit88 { iso (1) identified-organization (3) dod (6) internet (1) security (5) mechanisms (5) pkix (7) id-mod
                          (0) id-pkix1-explicit (18) }
-- Comme défini dans la RFC 3280.

  KerberosTime, PrincipalName, Realm, EncryptionKey, Checksum
        DE KerberosV5Spec2 { iso(1) identified-organization(3) dod(6) internet(1) security(5) kerberosV5(2) modules(4)
                             krb5spec2(2) };
-- Comme défini dans la RFC 4120.

id-pkinit IDENTIFIANT D'OBJET ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit
                                    (3) }

id-pkinit-authData        IDENTIFIANT D'OBJET  ::= { id-pkinit 1 }
id-pkinit-DHKeyData       IDENTIFIANT D'OBJET  ::= { id-pkinit 2 }
id-pkinit-rkeyData        IDENTIFIANT D'OBJET  ::= { id-pkinit 3 }
id-pkinit-KPClientAuth IDENTIFIANT D'OBJET  ::= { id-pkinit 4 }
id-pkinit-KPKdc           IDENTIFIANT D'OBJET  ::= { id-pkinit 5 }
id-pkinit-san         IDENTIFIANT D'OBJET ::= { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) x509SanAN (2) }
pa-pk-as-req ENTIER ::=                16
pa-pk-as-rep ENTIER ::=                17
ad-initial-verified-cas ENTIER ::=      9
td-trusted-certifiers ENTIER ::=      104
td-invalid-certificates ENTIER ::=    105
td-dh-parameters ENTIER ::=           109

PA-PK-AS-REQ ::= SEQUENCE {
       signedAuthPack       [0] CHAINE D'OCTETS IMPLICITE,
-- Contient un type CMS ContentInfo codé conformément à la [RFC3852].
-- Le champ contentType du type ContentInfo est id-signedData (1.2.840.113549.1.7.2), et le champ content est SignedData.
-- Le champ eContentType pour le type SignedData est id-pkinit-authData (1.3.6.1.5.2.3.1), et le champ eContent contient le
     codage en DER du type AuthPack.
-- AuthPack est défini ci-dessous.
        trustedCertifiers     [1] SEQUENCE DE ExternalPrincipalIdentifier FACULTATIF,
-- Contient une liste des CA de confiance pour le client qui peuvent être utilisées pour certifier le KDC.
-- Chaque ExternalPrincipalIdentifier identifie une CA ou un certificat de CA (et donc sa clé publique).
-- Les informations contenues dans le trustedCertifiers DEVRAIENT être utilisées par le KDC comme conseils pour guider
     son choix d'une chaîne de certificats appropriée à retourner au client.
        kdcPkId               [2] CHAINE D'OCTETS IMPLICITE FACULTATIF,
-- Contient un type CMS SignerIdentifier codé conformément à la [RFC3852].
-- Identifie, si présent, une clé publique de KDC particulière que le client a déjà.
        ...
```

```
    }

DHNonce ::=  CHAINE D'OCTETS

ExternalPrincipalIdentifier ::= SEQUENCE {
        subjectName                 [0] CHAINE D'OCTETS IMPLICITE FACULTATIF,
-- Contient un nom de type PKIX codé conformément à la [RFC3280].
-- Identifie le sujet de certificat par le nom de sujet distingué.
-- EXIGÉ quans un nom de sujet distingué est présent dans le certificat.
        issuerAndSerialNumber   [1] CHAINE D'OCTETS IMPLICITE FACULTATIF,
-- Contient un type CMS IssuerAndSerialNumber codé conformément à la [RFC3852].
-- Identifie un certificat du sujet.
-- EXIGÉ pour TD-INVALID-CERTIFICATES et TD-TRUSTED-CERTIFIERS.
        subjectKeyIdentifier        [2] CHAINE D'OCTETS IMPLICITE FACULTATIF,
-- Identifie la clé publique du sujet par un identifiant de clé. Quand un certificat X.509 est référencé, cet identifiant de clé
   correspond à la valeur d'extension de subjectKeyIdentifier X.509. Quand d'autres formats de certificat sont référencés, les
   documents qui spécifient le format de certificat et son utilisation avec la CMS doivent inclure les détails sur la
   confrontation de l'identifiant de clé au champ de certificat approprié.
-- RECOMMANDÉ pour TD-TRUSTED-CERTIFIERS.
        ...
    }

AuthPack ::= SEQUENCE {
        pkAuthenticator         [0] PKAuthenticator,
        clientPublicValue       [1] SubjectPublicKeyInfo FACULTATIF,
-- Le tType SubjectPublicKeyInfo est défini dans la [RFC3280].
-- Spécifie les paramètres de domaine Diffie-Hellman et la valeur de la clé publique du client [IEEE1363].
-- La valeur de la clé pubique DH est codée comme CHAINE DE BITS conformément à la [RFC3279].
-- Ce champ n'est présent que si le client souhaite utiliser la méthode d'accord de clé Diffie-Hellman.
        supportedCMSTypes   [2] SEQUENCE DE AlgorithmIdentifier FACULTATIF,
-- Le type AlgorithmIdentifier est défini dans la [RFC3280].
-- Fait la liste des identifiants d'algorithme de CMS [RFC3370] qui identifient les algorithmes de transport de clés, ou les
   algorithmes de chiffrement de contenu, ou les algorithmes de signature pris en charge par le client en ordre de préférence
   décroissante.
        clientDHNonce           [3] DHNonce FACULTATIF,
-- Présent seulement si le client indique qu'il souhaite réutiliser les clés DH ou permettre au KDC de le faire.
        ...
    }

PKAuthenticator ::= SEQUENCE {
        cusec                   [0] ENTIER (0..999999),
        ctime                   [1] KerberosTime,
-- cusec et ctime sont utilisés comme dans la [RFC4120], pour la prévention des répétitions.
        nonce                   [2] ENTIER (0..4294967295),
-- Choisi au hasard ; ce nom occasionnel n'a pas besoin de correspondre au nom occasionnel de KDC-REQ-BODY.
        paChecksum          [3] CHAINE D'OCTETS FACULTATIF,
-- DOIT être present. Contient la somme de contrôle SHA1 effectuée sur KDC-REQ-BODY.
        ...
    }

TD-TRUSTED-CERTIFIERS ::= SEQUENCE DE ExternalPrincipalIdentifier
-- Identifie une liste des CA de confiance pour le KDC.
-- Chaque ExternalPrincipalIdentifier identifie une CA ou un certificat de CA (et donc sa clé publique).

TD-INVALID-CERTIFICATES ::= SEQUENCE DE ExternalPrincipalIdentifier
-- Chaque ExternalPrincipalIdentifier identifie un certificat (envoyé par le client) avec une signature invalide.

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

AD-INITIAL-VERIFIED-CAS ::= SEQUENCE DE ExternalPrincipalIdentifier
```

```
-- Identifie le chemin de certification sur la base duquel le certificat de client a été validé.
-- Chaque ExternalPrincipalIdentifier identifie une CA ou un certificat de CA (et donc sa clé publique).

PA-PK-AS-REP ::= CHOIX {
        dhInfo              [0] DHRepInfo,
-- Choisi quand l'échange de clés Diffie-Hellman est utilisé.
        encKeyPack          [1] CHAINE D'OCTETS IMPLICITE,
-- Choisi quand le chiffrement à clé publique est utilisé.
-- Contient un type CMS ContentInfo codé conformément à la [RFC3852].
-- Le champ contentType de type ContentInfo est id-envelopedData (1.2.840.113549.1.7.3).
-- Le champ content est une EnvelopedData.
-- Le champ contentType pour le type EnvelopedData est id-signedData (1.2.840.113549.1.7.2).
-- Le champ eContentType pour le type interne SignedData (quand il n'est pas chiffré) est id-pkinit-rkeyData (1.3.6.1.5.2.3.3)
     et le champ eContent contient le codage en DER du type ReplyKeyPack.
-- ReplyKeyPack est défini ci-dessous.
        ...
      }

DHRepInfo ::= SEQUENCE {
        dhSignedData        [0] CHAINE D'OCTETS IMPLICITE,
-- Contient un type CMS ContentInfo codé conformément à la [RFC3852].
-- Le champ contentType du type ContentInfo est id-signedData (1.2.840.113549.1.7.2), et le champ content est SignedData.
-- Le champ eContentType pour le type SignedData est id-pkinit-DHKeyData (1.3.6.1.5.2.3.2), et le champ eContent contient
     le codage en DER du type KDCDHKeyInfo.
-- KDCDHKeyInfo est défini ci-dessous.
        serverDHNonce       [1] DHNonce FACULTATIF,
-- Présent si et seulement si dhKeyExpiration est présent.
        ...
      }

KDCDHKeyInfo ::= SEQUENCE {
        subjectPublicKey    [0] CHAINE DE BITS,
-- Clé pubique DH du KDC.
-- La valeur de la clé pubique DH est codée comme CHAINE DE BITS conformément à la [RFC3279].
        nonce               [1] ENTIER (0..4294967295),
-- Contient le nom occasionnel dans le champ pkAuthenticator dans la demande si les clés DH ne sont PAS réutilisées ;
     0 autrement.
        dhKeyExpiration     [2] KerberosTime FACULTATIF,
-- Heure d'expiration pour la paire de clés du KDC, présent si et seulement si les clés DH sont réutilisées.
-- Si elle est présente, la clé pubique DH du KDC NE DOIT PAS être utilisée après cette heure d'expiration.
-- Si ce champ est omis, alors le champ serverDHNonce DOIT aussi être omis.
        ...
      }

ReplyKeyPack ::= SEQUENCE {
        replyKey            [0] EncryptionKey,
-- Contient la clé de session utilisée pour chiffrer le champ enc-part dans la AS-REP, c'est-à-dire, la clé de réponse d'AS.
        asChecksum          [1] Checksum,
-- Contient la somme de contrôle de l'AS-REQ correspondant à la AS-REP contenante.
-- La somme de contrôle est effectuée sur le type AS-REQ.
-- La clé de protocole [RFC3961] de la somme de contrôle est la replyKey et le numéro d'usage de clé est 6.
-- Si le enctype de la replyKey est "plus récent" [RFC4120], [RFC4121], la somme de contrôle est l'opération de somme de
     contrôle exigée [RFC3961] pour ce enctype.
-- Le client DOIT vérifier cette somme de contrôle dès la réception de l'AS-REP.
        ...
      }

TD-DH-PARAMETERS ::= SEQUENCE DE AlgorithmIdentifier
-- Chaque AlgorithmIdentifier spécifie un ensemble de paramètres de domaine Diffie-Hellman [IEEE1363].
-- Cette liste est en ordre de préférence décroissante.
FIN
```

Appendice B. Vecteurs d'essais

La fonction octetstring2key() est définie au paragraphe 3.2.3.1. Cette section décrit quelques vecteurs d'essais qui pourraient être utiles aux mises en œuvre de octetstring2key().

Ensemble 1 :
=====
La chaîne d'octets d'entrée x est :

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Résultat de K-truncate() quand la taille de clé est 32 octets :

```
5e e5 0d 67 5c 80 9f e5 9e 4a 77 62 c5 4b 65 83
75 47 ea fb 15 9b d8 cd c7 5f fc a5 91 1e 4c 41
```

Ensemble 2 :
=====
La chaîne d'octets d'entrée x est :

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Résultat de K-truncate() quand la taille de clé est 32 octets :

```
ac f7 70 7c 08 97 3d df db 27 cd 36 14 42 cc fb
a3 55 c8 88 4c b4 72 f3 7d a6 36 d0 7d 56 78 7e
```

Ensemble 3 :
======
La chaîne d'octets d'entrée x est :

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e
0f 10 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d
0e 0f 10 00 01 02 03 04 05 06 07 08 09 0a 0b 0c
0d 0e 0f 10 00 01 02 03 04 05 06 07 08 09 0a 0b
0c 0d 0e 0f 10 00 01 02 03 04 05 06 07 08 09 0a
0b 0c 0d 0e 0f 10 00 01 02 03 04 05 06 07 08 09
0a 0b 0c 0d 0e 0f 10 00 01 02 03 04 05 06 07 08
```

Résultat de K-truncate() quand la taille de clé est 32 octets :

```
c4 42 da 58 5f cb 80 e4 3b 47 94 6f 25 40 93 e3
73 29 d9 90 01 38 0d b7 83 71 db 3a cf 5c 79 7e
```

Ensemble 4 :

=====
La chaîne d'octets d'entrée x est :
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e
0f 10 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d
0e 0f 10 00 01 02 03 04 05 06 07 08 09 0a 0b 0c
0d 0e 0f 10 00 01 02 03 04 05 06 07 08

Résultat de K-truncate() quand la taille de clé est 32 octets :
00 53 95 3b 84 c8 96 f4 eb 38 5c 3f 2e 75 1c 4a
59 0e d6 ff ad ca 6f f6 4f 47 eb eb 8d 78 0f fc

Appendice C. Informations diverses sur les mises en œuvre PKINIT de Microsoft Windows

Des révisions antérieures de PKINIT I-D ont été mises en œuvre dans diverses livraisons de Microsoft Windows et déployées en très grand nombre. Pour permettre à la communauté de mieux interopérer avec les systèmes qui fonctionnent avec ces livraisons, les informations suivantes peuvent être utiles.

Les certificats de KDC produits par les CA Windows 2000 Enterprise contiennent un SAN dNSName avec le nom DNS de l'hôte qui gère le KDC, et l'EKU id-kp-serverAuth [RFC3280].

Les certificats de KDC produits par les CA Windows 2003 Enterprise contiennent un SAN dNSName avec le nom DNS de l'hôte qui gère le KDC, l'EKU id-kp-serverAuth, et l'EKU id-ms-kp-sc-logon.

Il est prévu que la prochaine livraison de Windows soit déjà trop avancée pour permettre de prendre en charge la production de certificats de KDC avec le SAN id- pkinit-san comme spécifié dans la présente RFC. À la place, il y aura un SAN dNSName contenant le nom de domaine du KDC, et l'objet de ces certificats de KDC va être restreint par la présence des EKU id-pkinit-KPKdc et id-kp-serverAuth.

De plus pour vérifier que ceux-ci sont présents dans un certificat de KDC, les clients Windows vérifient que le producteur du certificat de KDC est dans un ensemble de producteurs autorisés de tels certificats, de sorte que ceux qui souhaitent produire des certificats de KDC doivent configurer leurs clients Windows de façon appropriée.

Les certificats de client acceptés par les KDC de serveur Windows 2000 et Windows 2003 doivent contenir un SAN id-ms-san-sc-logon-upn (1.3.6.1.4.1.311.20.2.3) et l'EKU id-ms-kp-sc-logon. Le SAN id-ms-san-sc-logon-upn contient une chaîne codée en UTF8 dont la valeur est celle de l'attribut de service de répertoire UserPrincipalName de l'objet compte de client, et l'objet de l'inclusion du SAN id-ms-san-sc-logon-upn dans le certificat de client est de valider la transposition de client (en d'autres termes, la clé publique du client est liée au compte qui a cette valeur de UserPrincipalName).

On devrait noter que tous les noms de domaine Kerberos de Microsoft sont des noms de domaines de style domaine et strictement en majuscules. De plus, l'attribut UserPrincipalName est unique au monde dans Windows 2000 et Windows 2003.

Adresse des auteurs

Larry Zhu
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
US
mél : lzhu@microsoft.com

Brian Tung
Aerospace Corporation
2350 E. El Segundo Blvd.
El Segundo, CA 90245
US
mél : brian@aero.org

Déclaration de droits de reproduction

ENGINEERING TASK FORCE déclinent toutes garanties, exprimées ou implicites, y compris mais non limitées à toute garantie que l'utilisation des informations ci-encloses ne violent aucun droit ou aucune garantie implicite de commercialisation ou d'aptitude à un objet particulier.

Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourraient être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faits au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf- ipr@ietf.org.

Remerciement

Le financement de la fonction d'édition des RFC est actuellement fourni par l'activité de soutien administratif (IASA) de l'IETF.