rfc4556

PKINIT

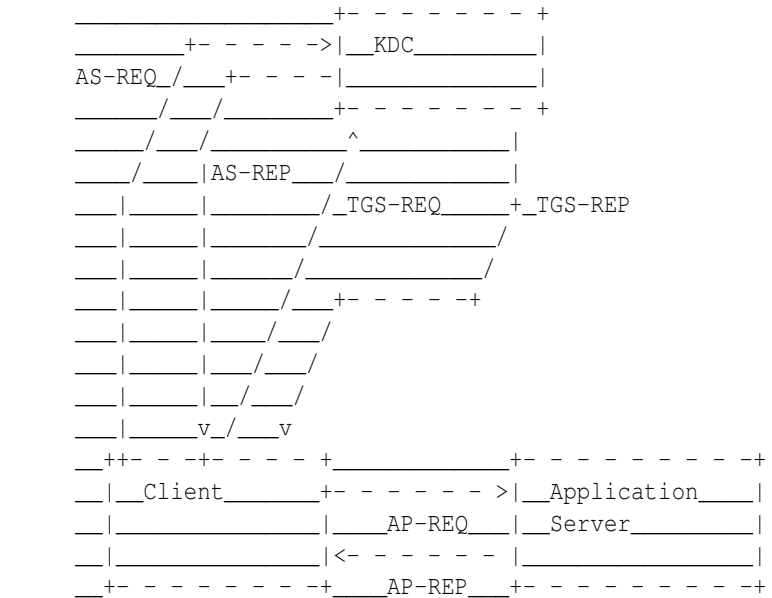
Ce document décrit des extensions au protocole Kerberos. Ces extensions fournissent une méthode pour intégrer le clé de chiffrement publique dans l'échange d'authentification initiale, en utilisant les clé asymétrique pour les algorithmes de chiffrement et/ou de signature dans les champs de donnée de pré-authentification.

La pierre angulaire de Kerberos 5 sont le ticket et l'authenticator. Un ticket encapsule une clé symétrique (la clé de session) dans un enveloppe (un message publique) à destination d'un service spécifique. Le contenu de ce ticket est chiffré avec une clé secrète partagée entre le principale de service et le KDC. La partie chiffrée du ticket contient le nom du principal du client, et d'autre éléments. Un authenticator est un enregistrement qui est généré en utilisant la clé de session du ticket associé. La clé de session est connue du client qui demande le ticket. Le contenu de l'authenticator est chiffré avec cette clé de session et contient un timestamp et le nom du principal du client.

Le protocole Kerberos 5 consiste des échanges de messages suivant, entre le client et le KDC, et le client et les services d'application :

- L'échange d'AS, où le client obtient un ticket initial, les TGT.
- L'échange TGT, où le client utilise sont TGT pour s'authentifier et demander des tickets de service.
- L'échange AP, où le client faite des demandes avec des messages consistant d'un ticket de service et d'un authenticator qui certifie que le client possède la clé de session du ticket. Le serveur peut optionnellement y répondre. Ces échange négocient généralement des clé de session symétriques.

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



Dans l'échange AS, la réponse du KDC contient la clé de session du ticket, avec d'autres éléments, qui est chiffré en utilisant une clé partagée entre le client et le KDC. La clé de réponse AS est généralement dérivée du mot de passe du client. Cependant, pour les personnes physiques, la force du protocole Kerberos est sujette à la force du mot de passe.

L'utilisation d'une clé asymétrique sous la forme d'un certificat X.509 permet de faciliter l'authentification de l'origine des données et la préservation du secret. Un PKI fournis une gestion et des mécanismes de distribution des clés qui peuvent être utilisé pour établir une authentification et une communication sûre.

L'avantage du TGT Kerberos est que le client expose sa clé à long-terme secrète seulement une fois. Le TGT est ses clé de session associés peuvent être utilisées pour d'autres demandes de ticket. Cela permet à toutes les autres authentification d'être indépendant de la méthode d'authentification initiale. En plus, l'utilisation de clé symétrique après l'authentification initial est préférable pour des raisons de performance.

Extensions

Cette section décris les extensions pour supporter le chiffrement à clé publique dans la requête initiale pour un ticket. Brièvement, cet document définis les extensions suivantes :

- Le client indique l'utilisation d'une authentification à clé publique en incluant un pré-authentifiant spécial dans la demande initiale. Ce pré-authentifiant contient la clé publique du client et une signature.
- Le KDC teste la demande du client avec sa stratégie d'authentification et les autorités de certification de confiance.
- Si le demande passe les tests de vérification, le KDC répond normalement, mais la réponse est chiffrée en utilisant soit :
  - Un clé générée avec l'échange DH avec le client, signé avec la clé de signature du KDC ; ou
  - Un clé de chiffrement symétrique, signée avec la clé de signature du KDC est chiffrée en utilisant la clé publique du client.
- Le client valide la signature du KDC, obtient la clé de chiffrement, déchiffre la réponse, et traite la suite normalement.

Algorithmes requis

Toutes les implémentations PKINIT doivent supporter les algorithmes suivant :

- l'enctype de la réponse AS : aes128-cts-hmac-sha1-96 and aes256-cts-hmac-sha1-96
- L'algorithme de signature : sha-1WithRSAEncryption
- La méthode de livraison de la clé de la réponse AS : La méthode DH.

En plus, les implémentations de cette spécification doivent être capable de traiter l'extension EKU et le id-pkinit-san otherName de l'extention SAN dans les certificat X.509.

Algorithmes recommandés

Toutes les implémentations PKINIT devraient supporter les algorithmes suivant :

- La méthode de livraison de la clé de la réponse AS : voir la section Utiliser le chiffrement à clé publique

Pour les implémentations qui supporte le chiffrement à clé publique pour les méthodes de livraison de clé, Les algorithmes suivant doivent être supportés :

- Les algorithmes de transport de clé identifiés dans le champ keyEncryptionAlgorithm du type KeyTransRecipientInfo pour le chiffrement de clé temporaire dans le champ encryptedKey avec un clé publique : rsaEncryption.
- Les algorithmes de chiffrement de contenu identifiés dans le champ contentEncryptionAlgorithm du type EncryptedContentInfo pour le chiffrement de la clé de réponse AS avec la clé temporaire contenue dans le champ encryptedKey du type KeyTransRecipientInfo : des-ede3-cbc

Messages définis et types de chiffrement

PKINIT utilise les types de pré-auhentification suivants :
PA_PK_AS_REQ 16
PA_PK_AS_REP 17
PKINIT utilisé les types de données d'autorisation suivants :
AD_INITIAL_VERIFIED_CAS 9
PKINIT introduit les nouveaux codes d'erreurs 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_MUST_BE_INCLUDED 79
KDC_ERR_DIGEST_IN_SIGNED_DATA_NOT_ACCEPTED 80
KDC_ERR_PUBLIC_KEY_ENCRYPTION_NOT_SUPPORTED 81
PKINIT utilise les type de données suivant pour les erreurs :
TD_TRUSTED_CERTIFIERS 104
TD_INVALID_CERTIFICATES 105
TD_DH_PARAMETERS 109

Type de chiffrement Kerberos pour CMS Algorithm Identifiers

PKINIT définis les numéros de type de chiffrement Kerberos, qui peut être utilisé dans le champ etype du message AS-REQ pour indiquer au KDC l'acceptation du client les algorithmes correspondant (incluant les algorithmes de transport de clé algorithmes de chiffrement de contenu, et algorithmes de signature). Les type de chiffrement dans le champ etype sont dans l'ordre décroissant de préférence du client. La présent de chacun de ces types de chiffrement dans le champ etype est équivalent à la présence du l'OID l'algorithme correspondant la le champ supportedCMSTypes, et l'ordre de préférence dans le champ supportedCMSTypes a précédence sur l'ordre de préférence dans le champ etype.

```
Kerberos Encryption Type Name____Num____Corresponding Algorithm OID
============================____===____==============================
id-dsa-with-sha1-CmsOID__________ 9_____id-dsa-with-sha1 [RFC3370]
md5WithRSAEncryption-CmsOID______10_____md5WithRSAEncryption [RFC3370]
sha-1WithRSAEncryption-CmsOID____11_____sha-1WithRSAEncryption [RFC3370]
rc2-cbc-EnvOID___________________12_____rc2-cbc [RFC3370]
rsaEncryption-EnvOID_____________13_____rsaEncryption [RFC3447][RFC3370]
id-RSAES-OAEP-EnvOID_____________14_____id-RSAES-OAEP [RFC3447][RFC3560]
des-ede3-cbc-EnvOID______________15_____des-ede3-cbc [RFC3370]
```

Les numéros de type de chiffrement ci-dessus sont seulement utilisé pour indiquer le support des algorithmes correspondant dans PKINIT ; ils ne correspondant pas aux types de chiffrement Kerberos actuels et ne doivent pas être utilisé dans le champ etype. L'assignation des numéros de type de chiffrement Kerberos pour indiquer le support pour les algorithmes CMS est déprécié, et les nouveaux numéros ne devraient pas être assignés dans ce but. À la place, le champ supportedCMSTypes devrait être utilisé pour identifier les algorithmes supportés par le client et l'ordre de préférence du client.

Pour une intéropérabilité maximiale, cependant, les clients PKINIT désirant indiquer au KDC le support pour un ou plusieur algorithmes listés ci-dessus devraient inclure les numéros dans le champ etype de l'AS-REQ.

Syntaxe et utilisation de la pré-authentification PKINIT

Cette section définie la syntaxe et l'utilisation des champs de pré-authentification utilisés par PKINIT.

Génération d'une demande client

La demande d'authentification initilale (AS-REQ) est envoyée avec un élément de pré-authentification, dont le padata-type est **PA PK_AS_REQ** et dont padata-value contient le DER du type **PA-PK-AS-REQ**.

```
PA-PK-AS-REQ ::= SEQUENCE {
 signedAuthPack [0] IMPLICIT OCTET STRING
 trustedCertifiers [1] SEQUENCE OF ExternalPrincipalIdentifier OPTIONAL
 kdcPkId [2] IMPLICIT OCTET STRING OPTIONAL
 ...
}
DHNonce ::= OCTET STRING
ExternalPrincipalIdentifier ::= SEQUENCE {
 subjectName [0] IMPLICIT OCTET STRING OPTIONAL
 issuerAndSerialNumber [1] IMPLICIT OCTET STRING OPTIONAL
 subjectKeyIdentifier [2] IMPLICIT OCTET STRING OPTIONAL
 ...
}
AuthPack ::= SEQUENCE {
 pkAuthenticator [0] PKAuthenticator,
 clientPublicValue [1] SubjectPublicKeyInfo OPTIONAL
 supportedCMSTypes [2] SEQUENCE OF AlgorithmIdentifier OPTIONAL
 clientDHNonce [3] DHNonce OPTIONAL
 ...
}
PKAuthenticator ::= SEQUENCE {
 cusec [0] INTEGER (0..999999)
 ctime [1] KerberosTime
 nonce [2] INTEGER (0..4294967295)
 paChecksum [3] OCTET STRING OPTIONAL
 ...
}
```

signedAuthPack Contient un CMS type ContentInfo. Le champ contentType est id-signedData (1.2.840.113549.1.7.2), et le champ tontent est un 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 DER du type AuthPack

trustedCertifiers Liste de CA, trusté par le client qui peuvent être utilisés pour certifier le KDC.

kdcPkId Contient un CMS type SignerIdentifier. Identifie, si présent, une clé publique d'un KDC particulier que le client a déjà.

subjectName Conient un nom de type PKIX. Identifie le sujet du certificat par le nom distinct du sujet.

issuerAndSerialNumber Contient un type CMS IssuerAndSerialNumber. Identifie un certificat du sujet.

subjectKeyIdentifier Identifie la clé publique du sujet par une ID de clé.

clientPublicValue Spécifie les paramètres de domaine DH et la valeur de clé publique du client.

supportedCMSTypes Liste d'algorithmes CMS qui identifie les algorithmes de transport de clé ou de signature supportés par le client, par ordre de préférence.

clientDHNonce Présent seulement si le client indique qu'il souhaite ré-utiliser les clés DH ou autoriser le KDC à le faire.

paChecksum Contient le checksum SHA1.

eContentType ce champ pour le type SignedData est id-pkinit-authData : { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) authData(1) }

eContent Ce champ pour le type SignedData contient le DER du type AuthPack

signerInfos Ce champ pour le type SignedData contient un simple signerInfo, qui contient la signature du type AuthPack.

AuthPack Cette structure contient un PKAuthenticator, les informations de clé publique client, les type de chiffrement CMS supportés et un DHNonce.

pkAuthenticator certifie au KDC que le client a une connaissance récente de la clé de signature qui authentifie le client.

clientPublicValue spécifie les paramètre de domaine DH et la valeur de clé publique du client. La valeur de clé publique SH est encodée en BIT STRING.

clientPublicValue est présent seulement si le client désire utilisé DH.

supportedCMSTypes spécifie la liste des ID d'algorithmes CMS qui sont supportés par le client et peut être utilisé pour identifier un algorithme de signature ou un algorithme de transport de clé dans le champ keyEncryptionAlgorithm de type KeyTransRecipientInfo, ou un algorithme de chiffrement de contenu dans le champ contentEncryptionAlgorithm de type EncryptedContentInfo.

clientDHNonce est décrit plus bas

ctime dans la structure PKAuthenticator contient l'heure courante dans l'hôte du client, et le champ cusec est la partie micro-secondes. Ils sont utilisé ensemble pour spécifier un timestamp suffisamment précis. le champ nonce est choisi au hasard. le champ paChechsum doit être présent et contient un checksum SHA1. Pour faciliter les migrations futures de l'utilisation de SHA1, le champ paChecksum est optionnel : quand la demande est étendue à la négociation d'algorithmes de hash, le nouveau client qui souhaite ne pas utiliser SHA1 va envoyer la demande sans ce champ. Le KDC ce conformant à cette spécification doivent retourner KRB-ERROR avec le code KDC_ERR_PA_CHECKSUM_MUST_BE_INCLUDED. Cela permet à un nouveau client de retenter avec SHA1 si permis par la stratégie locale.

certificates de type SignedData, contient les certificats utilisés pour la construction du chemin de certification, pour que le KDC puisse vérifier la signature sur le type AuthPack. Cet certificats devraient être suffisants pour construire au moins chemin de certification depuis le certificat client. Le client doit être capable d'inclure un tel jeu de certificats s'il est configuré pour le faire. Le champ certificates ne doit pas contenir de certificat CA root.

clientPublicValue La valeur public Diffie-Hellman est inclues si le client veut utiliser la méthode DH. Les paramètres de domaine DH pour la clé public du client sont spécifiés dans le champ algorithm du type SubjectPublicKeyInfo, et la valeur de clé publique DH du client est mapppé à un ubjectPublicKey. En utilisant la méthode d'agrément de clé DH, les implémentations doivent supporter Oakley 1024-bits Modular Exponential (MODP) group 2 et OakleOakley 2048-bits Modular Exponential (MODP) group 14 et devraient supporter OakleOakley 4096-bits Modular Exponential (MODP) group 16.

La structure ExternalPrincipalIdentifier est utilisée dans ce document pour identifier la clé publique du sujet. Cette structure est construite comme suit :

subjectName Contient un nom de type PKIX encodé en accord avec la rfc3280. Ce champ identifie le sujet du certificat par un nom de sujet distinct. Ce champ est requis quand il y a un distinguished subject name présent dans le certificat utilisé.

issuerAndSerialNumber Contient un type CMS encodé. Ce champ identifie un certificat du sujet. Il est requis pour TD-INVALID-CERTIFICATES et TD-TRUSTED-CERTIFIERS

subjectKeyIdentifier Identifie la clé publique du sujet par une key identifier. Quand un certificat X.509 est référencé, cette Key Identifier matche la valeur d'extention subjectKeyIdentifier. Quand d'autres fomats de certificat sont référencés, les documents qui spécifient le format de certificat et leur utilisation avec le CMS doivent inclurent les détails du match de key identifier au champ de certificat approprié.

Le champ trustedCertifiers du type PA-PK-AS-REQ contient une liste de CA, trustés par le client, qui peut être utilisé pour certifier le KDC. Chaque ExternalPrincipalIdentifier identifie une CA ou un certificat CA. Le champ kdcPkId du type PA-PK-AS-REQ contient un type CMS SignerIdentifier encodé. Ce champ identifie, si présent, une clé publique d'un KDC particulier que le client possède.

Réception de la demande client

Une fois reçu la demande du client, le KDC la valide. Le KDC vérifie la signature du client dans le champ signedAuthPack. Si, pendant la validation du certificat du client, le KDC ne peut pas construire un chemin de validation, il retourne un KRB-ERROR avec le code KDC_ERR_CANT_VERIFY_CERTIFICATE. Le e-data qui l'accompagne est un TYPED-DATA qui contient un élément dont data-type est TD_TRUSTED_CERTIFIERS, et contient le DER du type TD-TRUSTED-CERTIFIERS :

```
TD-TRUSTED-CERTIFIERS ::= SEQUENCE OF
 ExternalPrincipalIdentifier
 - Identifies une liste de CA trustés par le KDC.
 - Chaque ExternalPrincipalIdentifier identifies une CA
 - ou un certificat CA
```

Une fois ce message d'erreur reçu, le client devrait retenter seulement s'il a un jeu de certificats différent qui forment un chemin de certification acceptable pour le KDC.

Si, pendant le traitement du chemin de certification, de KDC détermine qu'une signature dans le champ signedAuthPack est invalide, il retourne KRB-ERROR avec le code KDC_ERR_INVALID_CERTIFICATE. Le e-data qui l'accompagne est un TYPED-DATA qui contient un élément dont data-type est TD_INVALID_CERTIFICATES, et contient le DER du type TD-INVALID-CERTIFICATES :

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

Si plus d'une signature est invalide, le KDC peut inclure un IssuerAndSerialNumber par signature invalide dans le TD-INVALID-CERTIFICATES. Le certificat X.509 du client est validé en accord avec la rfc3280.

En fonction de la stratégie locale, le KDC peut également vérifier si un certificat X.509 dans le chemin de certification a été révoqué. Si c'est le cas, le KDC doit retourner un message d'erreur avec le code KDC_ERR_REVOKED_CERTIFICATE. Si le KDC tente de déterminer le statut de révocation mais n'est pas en mesure 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 comme pour le code d'erreur KDC_ERR_INVALID_CERTIFICATE.

Noter que les données d'erreur TD_INVALID_CERTIFICATES est uniquement utilisé pour identifier les certificats invalides envoyés par le client dans le demande.

La clé publique du client est ainsi utilisée pour vérifier la signature. Si la vérification de la 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 pour ce message d'erreur.

En plus de la validation de la signature du client, le KDC doit également vérifier que la clé publique du client utilisée pour vérifier la signature est liée au principal du client comme suit :

1. Si le KDC a sa propre liaison entre la clé publique du client ou le certificat du client et le nom du principal du client, il utilise cette liaison.
2. Sinon, si le certificat X.509 du client contient un SAN avec un KRB5PrincipalName dans le champ otherName de type GeneralName, il lie le certificat du client à ce nom.

Le type de champ otherName est AnotherName. Le champ type-id du type AnotherName est id-pkinit-san :
```
id-pkinit-san OBJECT IDENTIFIER ::=
 { 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 du client dans le AS-REQ ne matche pas un nom lié par le KDC, ou s'il n'y a aucune liaison trouvée 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 pour ce type d'erreur.

Même si le chemin de certification est validée et que le certificat est mappé au nom principal du client, le KDC peut décider de ne pas accepter le certificat du client, en fonction de la stratégie locale.

Le KDC peut imposer la présence du EKU KeyPurposeId id-pkinit-KDClientAuth dans le champ extensions du certificat du client :

```
id-pkinit-KPClientAuth OBJECT IDENTIFIER ::=
 { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2)
 pkinit(3) keyPurposeClientAuth(4) }
 - PKINIT client authentication.
 - Key usage bits that MUST be consistent:
 - digitalSignature.
```

Le bit d'utilisation de clé digitalSignature doit être présent quand le but du certificat du client est restreins avec le id-pkinit-KDClientAuth.

Si ce EKU KeyPusposeId est requis mais n'est pas présent, ou si le certificat du client est restreins à ne pas l'utiliser pour PKINIT, le KDC doit retourner un message d'erreur avec le code KDC_ERR_INCONSISTENT_KEY_PURPOSE. Il n'y a pas de e-data pour cette erreur. Les KDC implémentant ce requis devraient également accepter le EKU KeyPurposeId id-ms-kp-sc-logon (1.3.6.1.4.1.311.20.2.2), vu que de nombreux certificats déployés pour PKINIT ont ce EKU.

En conséquence de la stratégie locale, le KDC peut décider de rejeter les demandes sur la base de l'absence ou la présence d'autres EKU spécifiques.

Si l'algorithme digest utilisé pour générer la signature CA pour la clé publique dans un certificat de la demande n'est pas acceptable, le KDC doit retourner un KRB-ERROR avec le code KDC_ERR_DIGEST_IN_CERT_NOT_ACCEPTED. Le e-data doit être un TYPED-DATA encodé, bien que rien n'est définis.

Si la clé publique du client n'est pas acceptée pour des raisons autre que ceux spécifiés, le KDC retourne un message KRB-ERROR avec le code KDC_ERR_CLIENT_NOT_TRUSTED. Il n'y a pas de e-data dans ce message.

Le KDC doit vérifier le timestamp pour s'assurer que la demande n'est pas rejouée, et que la plage de temps est dans la limite acceptable. Si la vérification échoue, le KDC doit retourner un code d'erreur KRB_AP_ERR_REPEAT ou KRB_AP_ERR_SKEW, respectivement.

Si clientPublicValue est remplis, indiquant que le client désire utiliser la méthode d'agrément de clé DH, le KDC devrait vérifier si les paramètres satisfont sa stratégie. Si ce n'est pas le cas, il doit retourner un message d'erreur avec le code KDC_ERR_DH_KEY_PARAMETERS_NOT_ACCEPTED. Le e-data qui l'accompagne est un TYPED-DATA qui contient un élément dont le data-type est TD-DH–PARAMETERS :

```
TD-DH-PARAMETERS ::= SEQUENCE OF AlgorithmIdentifier
 - Chaque AlgorithmIdentifier spécifie un jeu de
 - paramètres de domaine Diifie-Hellman. Cette liste
 - est dans l'ordre de préférence décroissante.
```

La structure AlgorithmIdentifier est définie dans la rfc3280 et est renseigné en accord avec la rfc3279. Si le client inclus un champ kdcPkId dans le PA-PK-AS-REQ et que le KDC ne possède pas la clé correspondante, le KDC doit ignorer le kdcPkId comme si le client ne l'avait pas inclus.

Si l'algorithme digest utilisé par le id-pkinit-authData n'est pas acceptable par le KDC, le KDC doit retourner un KRB-ERROR avec le code KDC_ERR_DIGEST_IN_SIGNED_DATA_NOT_ACCEPTED. Le e-data qui l'accompagne doit être encodé en TYPED-DATA, bien que rien n'est définis.

Génération de la réponse KDC

Si le paChechsum dans la demande n'est pas présente, le KDC se conformant a cette spécification doivent retourner un KRB-ERROR avec le code KDC_ERR_PA_CHECKSUM_MUST_BE_INCLUDED. Le e-data qui l'accompagne doit être encodé dans un TYPED-DATA.

En assumant que la demande du client a été validée, le KDC procède comme dans la rfc4120, excepté ce qui suit.

Le KDC doit mettre le flag initial et inclure un élément de donnée d'autorisation de ad-type AD_INITIAL_VERIFIED_CAS dans le ticket fournis. Le champ ad-data contient le DER du type AD-INITIAL-VERIFIED-CAS :

```
AD-INITIAL-VERIFIED-CAS ::= SEQUENCE OF ExternalPrincipalIdentifier
 - Identifie le chemin de certification avec lequel le certificat
 - client a été validé. Chaque ExternalPrincipalIdentifier
 - identifie une CA ou un certificat CA
```

Noter que la syntaxe pour les données d'autorisation AD-INITIAL-VERIFIED-CAS ne permet pas d'encoder des SEQUENCES vides. De tels séquences vides peuvent uniquement être utilisée si le KDC garantis lui-même le certificat du client.

l'AS enveloppe toute donnée AD-INITIAL-VERIFIED-CAS dans des conteneur AD-IF-RELEVANT si la liste des CA satisfait la stratégie locale de l'AS. De plus, tout TGS doit copier de telles données d'autorisation depuis les tickets utilisé dans un PA-TGS-REQ du TGS-REQ dans le ticket résultant. Si la liste des CA satisfait la stratégie de domaine du KDC, le TGS peut envelopper les données dans le AD-IF-RELEVANT ; sinon, il peut sortir les données d'autorisation du conteneur AD-IF-RELEVANT.

Les serveurs d'application qui comprennent ce type de données d'autorisation devraient appliquer la stratégie locale pour déterminer si un ticket donné comportant un tel type non contenu dans un conteneur AD-IF-RELEVANT est acceptable. (Cela correspond au serveur AP qui vérifie le champ transited quand le flag TRANSITED-POLICY-CHECKED n'est pas mis). Si un tel type de donnée est contenu dans un conteneur AD-IF-RELEVANT, les serveurs AP peuvent appliquer la stratégie locale pour déterminer si les données d'autorisation sont acceptables.

Un élément de donnée de pré-authentification, dont padata-type est PA_PK_AS_REP et dont padata-value contient le DER du type PA-PK-AS-REP, est inclus dans le AS-REP.

```
PA-PK-AS-REP ::= CHOICE {
 dhInfo [0] DHRepInfo,
 - Sélectionné quand l'échange de clé DH est utilisé
 encKeyPack [1] IMPLICIT OCTET STRING,
 - Sélectionné quand le chiffrement par clé publique est utilisé
 - Contient un type CMS ContentInfo encodé
 - 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 SignedData est id-pkinit-rkeyData (1.3.6.1.5.2.3.3)
 - et le champ eContent contient le DER du type ReplyKeyPack
}
KDCDHKeyInfo ::= SEQUENCE {
 subjectPublicKey [0] BIT STRING,
 - La clé publique DH du KDC
 - la clé publique DH est encodée en BIT STRING
 nonce [1] INTEGER (0..4294967295),
 - Contient le nonce dans le champ pkAuthenticator dans le demande si les clés DH ne sont pas réutilisées. 0
sinon.
 dhKeyExpiration [2] KerberosTime OPTIONAL,
 - Temp d'expiration pour la paire de clé du KDC, présent si est seulement si les clé DH sont réutilisées
 - Si présent, la clé publique DH du KDC ne doit pas être utilisé passé cette expiration
 - Si ce champ est omis, le champ serverDHNonce doit également être utilisé.
}
```

Le contenu de AS-REP est inchangé sinon. Le KDC chiffre la réponse normalement, mais pas avec la clé long-terme du client. À la place, il le chiffre avec soit une clé partagée dérivée d'un échange DH, ou une clé de chiffrement générée. Le contenu de PA-PK-AS-REP indique quelle méthode de livraison de clé est utilisé.

Si le client ne souhaite pas utiliser la méthode DH (le champ clientPublicValue n'est pas présent dans la demande) et que le KDC ne supporte pas 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 accompagnant ce message d'erreur.

En plus, la durée de vie du ticket retourné par le KDC ne doit pas excéder celle de la paire de clé du client.

Utiliser l'échange de clé Diffie-Hellman

Dans ce cas, le PA-PK-AS-REP contient une structure SHRepInfo. La structure ContentInfo pour le champ shSignedData est remplis comme suis :

1. Le champ contentType du type ContentInfo est id-signedData, et le champ content est un SignedData
2. Le champ eContentType pour le type SignedData est l'OID pour id-pkinit-DHKeyData : { iso(1) org(3) dod(6) internet(1) security(5) kerberosv5(2) pkinit(3) DHKeyData(2) }.
3. Le champ eContent pour le type SignedData contient le DER du type KDCDHKeyInfo.
4. La structure KDCDHKeyInfo contient la clé publique du KDC, un nonce, et optionnellement l'expiration de la clé DH du KDC qui est réutilisée. Le champ subjectPublicKey du type KDCDHKeyInfo identifie la clé publique DH du KDC. Ce clé est encodée en BIT STRING. Le champ nonce contient le nonce dans le champ pkAuthenticator dans le demande si les clés DH ne sont pas réutilisées. La valeur de ce champ nonce est 0 si les clés DH sont réutilisées. Le champ dhKeyExpiration est prisent si et seulement si les clé DH sont réutilisées. Si le champ dhKeyExpiration est présent, la clé publique DH du KDC dans cette structure KDCDHKeyInfo ne doit pas être utilisée passé ce temps. Si ce champ est omis, le champ serverDHNonce doit également être omis.
5. Le champ signerInfo du type SignedData contient un simple signerInfo, qui contient la signature pour le type KDCDHKeyInfo.
6. Le champ certificat du type SignedData contient les certificats prévus pour faciliter la construction du chemin de certification, pour que le client puisse vérifier la signature du KDC pour le type KDCDHKeyInfo. Les informations contenus dans le trustedCertifiers dans la demande devraient être utilisés par le KDC pour l'aider dans sa sélection d'une chaîne de certificats approprié à retourner au client. Ce champ peut être vide si la clé publique du KDC spécifiée par le champ kdcPkId dans le PA-PK-AS-REQ a été utilisé pour signer. Le KDC doit être capable d'inclure un tel jeu de certificat s'il est configuré pour le faire. Ce champ ne doit pas contenir de certificat CA root.
7. Si le client inclus le champ clientDHNonce, le KDC peut choisir de réutiliser ses clés DH, et doit inclure un temps d'expiration. Quand le serveur réutilise les clé DH, il doit inclure un serverDHNonce au moins aussi long que la longueur des clé pour le système de chiffrement symétrique utilisé pour chiffrer la réponse AS. Noter qu'inclure le serverDHNonce change la manière dont le client et le serveur calculent le clé à utiliser pour chiffrer la réponse. Le KDC ne dervait pas réutiliser les clé DH sauf si clientDHNonce est présent dans la demande.

La réponse AS est dérivée comme suit :

1. Le KDC et le client calculent la clé partagée comme suit : Quand MODP Diffie-Hellman est utilisé, les DHSharedSecret être la valeur de clé secrète. DHSharedSecret est la valeur ZZ, et est remplis avec des 0 pour la taille en octets soit la même que le modulo, et représente une chaîne d'octet dans l'ordre big-endian.
2. Laisse K être la source de génération de clé de la réponse AS.
3. Définie la fonction octetstring2key() comme suis :

```
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 de concaténation, 0x00, etc. sont représentés comme octet simple, random-to-key() est une opération qui génère une clé de protocole depuis un bitstring de longueur K ; et K-truncate tronque son entrée au premiers K bits.

4. Quand les clé DH sont réutilisées, laisse n_c être le clientDHNonce et n_k être le serverDHNonce, sinon ils sont vides.
5. La clé de réponse AS est : **k = octetstring2key(DHSharedSecret | n_c | n_k)**

Utiliser le chiffrement à clé publique

Dans ce cas, le PA-PK-AS-REP contient le champ encKeyPack où la clé de réponse AS est chiffrée. La structure ContentInfo pour le champ encKeyPack est construit comme suit :

1. Le champ contentType du type ContentInfo est id-envelopedData, et le champ content est un EnvelopedData
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 SignedData (quand déchiffré depuis le 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) }.
4. Le champ eContent pour le type SignedData contient le DER du type ReplyKeyPack
5. Le champ signerInfos du type SignedData contient un simple signerInfo, qui contient la signature pour le type ReplyKeyPack.
6. Le champ certificates pour le type SignedData contient les certificats pour construire le chemin de certification, pour que le client puisse vérifier la signature du KDC pou le type ReplyKeyPack. Les informations contenues dans le trustedCertifiers dans la demande devraient être utilisés par le KDC comme guide dans sa sélection d'une chaîne de certification appropriée. Ce champ peut être vide si la clé publique du KDC spécifiée par le champ kdcPkId dans le PA-PK-AS-REQ a été utilisé pour signer. Sinon, pour la validation du chemin, ces certificats devraient être suffisant pour construire au moins un chemin de certification depuis le certificat du KDC. Le KDC doit être capable d'inclure un tel jeu s'il est configuré pour le faire. Ce champ ne doit pas contenir de certificat CA root.
7. Le champ recipientInfos du type EnvelopedData est un jeu qui doit contenir exactement un membre du type KeyTransRecipientInfo. Le encryptedKey de ce membre contient la clé temporaire qui est chiffrée en utilisant la clé publique du client.
8. Les champs unprotectedAttrs ou originatorInfos du type EnvelopedData peut être présent.

S'il y a un champ supportedCMSTypes dans le AuthPack, le KDC doit vérifier s'il supporte un des types listés. S'il supporte plus d'un type, le KDC devrait utilisé le premiers qui listé. S'il n'en supporte aucun, le KDC doit retourner un message d'erreur avec le code KDC_ERR_ETYPE_NOSUPP.

Le KDC calcule le checksum du AS-REQ dans la demande du client. Ce checksum est effectué sur le type AS-REQ, et la clé de protocole de l'opération checksum est le replyKey, et le numéro d'utilisation de clé est 6. Si le enctype de replyKey est "newer", l'opération checksum est l'opération checksum requise de ce enctype :

```
ReplyKeyPack ::= SEQUENCE {
 replyKey [0] EncryptionKey,
 - contient la clé de session utilisé pour chiffrer le
 - champ enc-part dans le AS-REP. par ex: la clé de réponse AS
 asChecksum [1] Checksum,
 - Contient le checksum du AS-REQ correspondant au AS-REP
 - Le checksum est effectué sur le type AS-REQ.
 ...
 }
```

Les implémentations de cette méthode de livraison de clé de chiffrement RSA devraient supporter les clé RSA d'au moins 2048-bits.

Réception de la réponse du KDC

Une fois la réponse du KDC reçus, le client procède comme suit. Si le PA-PK-AS-REP contient le champ dhSignedData, le client dérive le clé de réponse AS en utilisant la même procédure utilisée par le KDC. Sinon, le message contient le champ encKeyPack, et le client

déchiffre de extrait la clé temporaire dans le champ encryptedKey du membre KeyTransRecipientInfo et l'utilise comme clé de réponse AS. Si la méthode de chiffrement à clé publique est utilisé, le client doit vérifier le asChecksum dans le ReplyKeyPack.

Dans tous les cas, le client doit vérifier la signature dans le SignedData. En plus, sauf si le client peut vérifier que la clé publique utilisée pour vérifier la signature du KDC est liée au KDC du domaine cible, le certificat du KDC doit contenir un SAN ayant un AnotherName dont type-id est id-pkinit-san et dont la valeur est un KRB5PrincipalName qui correspond au nom du TGS du domaine cible.

À moins que le client connaisse par d'autres moyens que le certificat du KDC est prévu pour un KDC, le client doit imposer que le certificat du KDC contienne le EKU KeyPurposeId id-pkinit-KPKdc :

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

Le bit d'utilisation de clé digitalSignature doit être mis quand le but du certificat du KDC est restreins avec le EKU id-pkinit-KPKdc. Si le certificat du KDC contient le nom du TGS encodé en SAN id-pkinit-san, ce certificat est certifié par la CA qui l'a délivré, et donc l'EKU id-pkinit-KPKdc n'est pas requis. Si toutes les vérifications applicables sont satisfaites, le client déchiffre le champ enc-part du KDC-REP dans le AS-REP, utilisant la clé de réponse AS, puis procède normalement.

Prérequis d'intéropérabilité

Le client doit être capable d'envoyer un jer de certificats suffisant pour permettre au KDC de construire un chemin de certification pour le certificat du client, si le jeu correct de certificat est fournis via la configuration ou la stratégie. Si le client envoie tous les certificats X.509 dans un chemin de certification en une chaîne acceptable pour le KDC, et si le KDC ne peut par vérifier la clé publique du client, le KDC doit être capable de traiter la validation pour le certificat du client basé sur les certificats dans le demande.

Indication du support PKINIT

Si la pré-authentification est requise mais n'est pas présente dans la requête, un message d'erreur avec le code KDC_ERR_PREAUTH_FAILED est retourné, et un objet METHOD-DATA sera présent dans le champ e-data du message KRB-ERROR pour spécifier quels mécanismes de pré-authentification sont acceptables. Le KDC peut ainsi indiquer le support de PKINIT en incluant un élément vide dont le padata-type est PA_PK_AS_REQ dans l'objet METHOD-DATA.

Sinon s'il est requis par la stratégie local du KDC que le client doive être pré-authentifié en utilisant le mécanisme de pré-authentification spécifié dans ce document, mais qu'aucune pré-authentification PKINIT n'est présent dans la requête, une message d'erreur avec le code d'erreur KDC_ERR_PREAUTH_FAILED devrait être retourné.

Le KDC doit laisser le champ padata-value de l'élément PA_PK_AS_REQ dans le METHOD-DATA du KRB-ERROR vide, et les clients doivent l'ignorer. De futures extensions de ce protocole peuvent spécifier d'autres données à envoyer au lieu d'un OCTET STRING vide.