Groupe de travail Réseau
Request for Comments : 4120
RFC rendue obsolète : 1510
Catégorie : En cours de normalisation

C. Neuman, USC-ISI
T. Yu, S. Hartman, K. Raeburn, MIT
juillet 2005
Traduction Claude Brière de L'Isle, septembre 2007

Le service Kerberos (V5) d'authentification de réseau

Résumé
Le présent document donne une vue d'ensemble et la spécification de la version 5 du protocole Kerberos, et il rend obsolète la RFC 1510 afin d'éclaircir certains aspects du protocole et de sa destination qui exigent des explications plus précises ou plus claires que ce que fournissait la RFC 1510. Le présent document est destiné à fournir une description détaillée du protocole, convenant pour sa mise en œuvre, ainsi qu'une description des utilisations appropriées des messages du protocole et des champs au sein de ces messages.

Table des matières

1 Introduction

Le présent document décrit les concepts et modèles sur lesquels est fondé le système Kerberos d'authentification de réseau. Il spécifie aussi la version 5 du protocole Kerberos. Les motivations, buts, hypothèses, et raisons qui sont derrière la plupart des concepts sont traités rapidement ; ils sont décrits plus en détail dans un article disponible dans les communications de l'IEEE [NT94] et antérieurement dans le chapitre Kerberos du plan technique Athena [MNSS87].

Le présent document n'est pas destiné à décrire Kerberos pour l'utilisateur final, l'administrateur de système, ou le développeur d'application. Des articles de plus haut niveau qui décrivent la version 5 du système Kerberos [NT94] et documentent la version 4 [SNS88] sont disponibles par ailleurs.

Le modèle Kerberos se fonde en partie sur le protocole d'authentification par tiers de confiance [NS78] de Needham et Schroeder et sur des modifications suggérées par Denning et Sacco [DS81]. Le concept original et la mise en œuvre des versions 1 à 4 de Kerberos a été le travail de deux anciens membres du personnel du projet Athena, Steve Miller de Digital Equipment Corporation et Clifford Neuman (maintenant à l'Institut des sciences de l'information de l'Université de Southern California), ainsi que de Jerome Saltzer, directeur technique du projet Athena, et Jeffrey Schiller, directeur du réseau du campus du MIT. De nombreux autres membres du projet Athena ont aussi contribué au travail sur Kerberos.

La Version 5 du protocole Kerberos (décrit dans le présent document) a évolué à cause de nouvelles exigences et du souhait de caractéristiques non disponibles dans la version 4. La conception de la version 5 a été dirigée par Clifford Neuman et John Kohl avec beaucoup d'apports de la part de la communauté. Le développement de la mise en œuvre de

référence du MIT a été conduit par John Kohl et Theodore Ts'o, avec l'aide et les contributions de nombreux autres. Depuis la publication de la RFC 1510, de nombreuses extensions et révisions du protocole ont été proposées. Le présent document reflète certaines de ces propositions. Lorsque de tels changements impliquent des efforts significatifs, le document cite la contribution du proposant.

Les mises en œuvre de référence des versions 4 et 5 de Kerberos sont disponibles au public, et des mises en œuvre commerciales ont été développées et sont largement utilisées. On trouvera le détail des différences entre les versions 4 et 5 dans [KNT94].

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

1.1 Le protocole Kerberos

Kerberos fournit un moyen de vérifier les identités des principaux, (par exemple, l'utilisateur d'une station de travail ou un serveur du réseau) sur un réseau ouvert (non protégé). Ceci se fait sans s'appuyer sur des assertions du système d'exploitation de l'hôte, sans fonder la confiance sur les adresses des hôtes, sans exiger de sécurité physique de la part de tous les hôtes du réseau, et avec l'hypothèse que les paquets qui voyagent le long du réseau peuvent être lus, modifiés, et insérés à volonté. Dans ces conditions, Kerberos effectue l'authentification comme un service d'authentification de tiers de confiance en utilisant la cryptographie conventionnelle (à clé secrète partagée). Les extensions à Kerberos (qui sortent du domaine d'application du présent document) peuvent répondre à l'utilisation de la cryptographie à clé publique durant certaines phases du protocole d'authentification. De telles extensions prennent en charge l'authentification Kerberos pour les utilisateurs enregistrés auprès des autorités de certification à clé publique et apportent certains avantages à la cryptographie à clé publique dans des situations où elle est nécessaire.

Le processus de base de l'authentification de Kerberos est le suivant : un client envoie une demande au serveur d'authentification (AS) pour des "accréditifs" auprès d'un serveur donné. L'AS répond avec ces accréditifs, chiffrés dans la clé du client. Les accréditifs consistent en un "ticket" pour le serveur et une clé de chiffrement temporaire (souvent appelée une "clé de session"). Le client transmet le ticket (qui contient l'identité du client et une copie de la clé de session, toutes deux chiffrées dans la clé du serveur) au serveur. La clé de session (maintenant partagée par le client et le serveur) est utilisée pour authentifier le client et peut facultativement être utilisée pour authentifier le serveur. Elle peut aussi être utilisée pour chiffrer les communications ultérieures entre les deux parties ou à échanger une sous-clé de session séparée à utiliser pour chiffrer les communications ultérieures. Noter que de nombreuses applications n'utilisent les fonctions de Kerberos que sur l'initiation d'une connexion réseau fondée sur le flux. Sauf si une application effectue le chiffrement ou la protection d'intégrité pour le flux de données, la vérification d'identité ne s'applique qu'à l'initiation de la connexion, et elle ne garantit pas que les messages ultérieurs sur la connexion proviennent du même principal.

La mise en œuvre du protocole de base consiste en un ou plusieurs serveurs d'authentification fonctionnant sur des hôtes sécurisés physiquement. Les serveurs d'authentification entretiennent une base de données des principaux (c'est-à-dire, utilisateurs et serveurs) et leur clés secrètes. Des bibliothèques de codes fournissent le chiffrement et mettent en œuvre le protocole Kerberos. Afin d'ajouter l'authentification à ses transactions, une application de réseau normale ajoute les appels à la bibliothèque Kerberos, soit directement soit par l'intermédiaire de l'interface générique de programmation des services de sécurité (GSS-API, *Generic Security Services Application Programming Interface*) décrite dans document distinct [RFC4121]. Il résulte de ces appels la transmission des messages nécessaires pour réaliser l'authentification.

Le protocole Kerberos consiste en plusieurs sous protocoles (ou échanges). Le client a deux méthodes de base pour demander des accréditifs à un serveur Kerberos. Dans la première approche, le client envoie en clair une demande de ticket pour le serveur désiré à l'AS. La réponse est envoyée chiffrée dans la clé secrète du client. Habituellement, cette demande est pour un ticket distributeur de tickets (TGT, *ticket-granting ticket*), qui peut ultérieurement être utilisé avec le serveur-distributeur de tickets (TGS, *ticket-granting serveur*). Dans la seconde méthode, le client envoie une demande au TGS. Le client utilise le TGT pour s'authentifier lui-même auprès du TGS de la même manière que si il était en train de contacter n'importe quel autre serveur d'application qui demanderait l'authentification de Kerberos. La réponse est chiffrée dans la clé de session provenant du TGT. Bien que la spécification du protocole décrive l'AS et le TGS comme des serveurs séparés, en pratique ils sont mis en œuvre comme des points d'entrée de protocole différents au sein d'un unique serveur Kerberos.

Une fois obtenus, les accréditifs peuvent être utilisés pour vérifier l'identité des principaux dans une transaction, pour

s'assurer de l'intégrité des messages échangés entre eux, ou pour préserver la confidentialité des messages. L'application a toute liberté pour choisir quelle protection est nécessaire.

Pour vérifier les identités des principaux dans une transaction, le client transmet le ticket au serveur d'application. Comme le ticket est envoyé "en clair" (des parties en sont chiffrées, mais ce cryptage ne déjoue pas la répétition) et pourrait être intercepté et réutilisé par un attaquant, des informations supplémentaires sont envoyées pour prouver que le message a bien pour origine le principal à qui le ticket a été fourni. Cette information (appelée l'authentifiant) est chiffrée dans la clé de session et inclut un horodatage. L'horodatage prouve que le message a été généré récemment et n'est pas une répétition. Le chiffrement de l'authentifiant dans la clé de session prouve qu'elle a été générée par un possesseur de la clé de session. Comme personne d'autre que le principal demandeur et le serveur ne connaît la clé de session (elle n'est jamais envoyée en clair sur le réseau), cela garantit l'identité du client.

L'intégrité des messages échangés entre les principaux peut aussi être garantie en utilisant la clé de session (passée dans le ticket et contenue dans les accréditifs). Cette approche permet la détection à la fois des attaques en répétition et des attaques de modification du flux des messages. Elle est accomplie en générant et transmettant une somme de contrôle à l'épreuve des collisions (appelée ailleurs fonction de hachage ou de résumé) du message du client, entrée avec la clé de session. La confidentialité et l'intégrité des messages échangés entre les principaux peut être sécurisée en chiffrant les données à passer en utilisant la clé de session contenue dans le ticket ou la sous-clé de session trouvée dans l'authentifiant.

Les échanges d'authentification mentionnés ci-dessus exigent un accès en lecture seule à la base de données Kerberos. Parfois cependant, les entrées dans la base de données doivent être modifiées, comme lors de l'ajout de nouveaux principaux ou du changement de la clé d'un principal. Ceci se fait à l'aide d'un protocole entre un client et un serveur Kerberos tiers, le serveur d'administration de Kerberos (KADM, *Kerberos Administration Server*). Il y a aussi un protocole pour conserver plusieurs copies de la base de données Kerberos. Aucun de ces protocoles n'est décrit dans le présent document.

1.2 Fonctionnement inter domaine

Le protocole Kerberos est conçu pour fonctionner à travers les frontières organisationnelles. Un client dans une organisation peut être authentifié auprès d'un serveur dans une autre organisation. Chaque organisation qui souhaite faire fonctionner un serveur Kerberos établit son propre "domaine". Le nom du domaine dans lequel un client est enregistré fait partie du nom du client et peut être utilisé par le service d'extrémité pour décider s'il va honorer une demande.

En établissant des clés "inter domaine", les administrateurs de deux domaines peuvent permettre à un client authentifié dans le domaine local de prouver son identité aux serveurs dans d'autres domaines. L'échange de clés inter domaines (une clé distincte peut être utilisée pour chaque direction) enregistre le service d'allocation de tickets de chaque domaine comme un principal dans l'autre domaine. Un client est alors capable d'obtenir un TGT pour le service d'allocation de tickets de la part de son domaine local. Lorsque ce TGT est utilisé, le service distant d'allocation de tickets utilise la clé inter domaines (qui diffère habituellement de sa propre clé TGS normale) pour déchiffre le TGT ; et donc il est certain que le ticket a été produit par le propre TGS du client. Les tickets produits par le service distant d'allocation de tickets indiqueront au service d'extrémité que le client a été authentifié à partir d'un autre domaine.

Sans fonctionnement inter domaine, et avec la permission appropriée, le client peut arranger l'enregistrement d'un principal avec une dénomination à part dans un domaine distant et engager des échanges normaux avec les services de ce domaine. Cependant, même pour de petits nombres de clients, cela devient lourd à gérer, et des méthodes plus automatiques comme celles décrites ici sont nécessaires.

Un domaine est dit communiquer avec un autre domaine si les deux domaines partagent une clé inter domaines, ou si le domaine local partage une clé inter domaines avec un domaine intermédiaire qui communique avec le domaine distant. Un chemin d'authentification est la séquence des domaines intermédiaires par où transitent les communications d'un domaine à l'autre.

Les domaines peuvent être organisé hiérarchiquement. Chaque domaine partage une clé avec son parent et une clé différente avec chaque enfant. Si une clé inter domaines n'est pas directement partagée par deux domaines, l'organisation hiérarchique permet de construire facilement un chemin d'authentification.

Si on n'utilise pas d'organisation hiérarchique, il peut être nécessaire de consulter une base de données afin de construire un chemin d'authentification entre les domaines.

Bien que les domaines soient normalement hiérarchisés, des domaines intermédiaires peut être court-circuités pour réaliser une authentification inter domaines à travers des chemins d'authentification de remplacement. (Ils peuvent être établis pour rendre plus efficace les communications entre deux domaines.) Il est important pour le service d'extrémité de savoir quels domaines ont été traversés lors de la prise de décision sur la foi à accorder au processus d'authentification. Pour faciliter cette décision, un champ de chaque ticket contient les noms des domaines qui ont été impliqués dans l'authentification du client.

Le serveur d'application est responsable en dernier ressort de l'acceptation ou du rejet de l'authentification et DEVRAIT vérifier les champs de transit. Le serveur d'application peut choisir de s'appuyer sur le centre de distribution des clés (KDC, *Key Distribution Center*) du domaine du serveur d'application pour vérifier les champs de transit. Le KDC du serveur d'application va dans ce cas établir le fanion TRANSITED-POLICY-CHECKED. Les KDC des domaines intermédiaires peuvent aussi vérifier les champs de transit lorsqu'ils produisent les TGT pour d'autres domaines, mais il est conseillé qu'ils ne le fassent pas. Un client peut demander que les KDC ne vérifient pas le champ de transit en établissant le fanion DISABLE-TRANSITED-CHECK. Les KDC DEVRAIENT respecter ce fanion.

1.3 Choisir un principal avec lequel communiquer

Le protocole Kerberos donne les moyens de vérifier (sous réserve des hypothèses du paragraphe 1.6) que l'entité avec laquelle on va communiquer est la même que celle qui a été enregistrée auprès du KDC en utilisant l'identité revendiquée (comme nom de principal). Il est encore nécessaire de déterminer si cette identité correspond à l'entité avec laquelle on a l'intention de communiquer.

Lorsque des données appropriées ont été échangées à l'avance, l'application peut effectuer cette détermination de façon syntaxique sur la base de la spécification du protocole de l'application, des informations fournies par l'utilisateur, et sur les fichiers de configuration. Par exemple, le nom principal du serveur (y compris le domaine) pour un serveur telnet pourrait être déduit du nom d'hôte spécifié par l'utilisateur (d'après la ligne de commande telnet), le préfixe "host/" spécifié dans la spécification du protocole de l'application, et une transposition en une domaine Kerberos déduit syntaxiquement de la partie domaine du nom d'hôte spécifié et des informations tirées de la base de données Kerberos locale.

On peut aussi s'appuyer sur des tiers de confiance pour faire cette détermination, mais seulement lorsque les données obtenues du tiers sont convenablement protégées en intégrité lorsqu'elles résident sur le serveur du tiers de confiance et lors de leur transmission. Et donc, par exemple, on ne devrait pas se fier à un enregistrement DNS non protégé pour transposer un alias d'hôte en nom principal d'un serveur, acceptant le nom principal comme étant celui de la partie qu'on veut contacter, car un attaquant peut modifier la transposition et se faire passer pour cette partie.

Les mises en œuvre de Kerberos et les protocoles fondés sur Kerberos NE DOIIVENT PAS utiliser des interrogations DNS non sécurisées pour canoniser les composants de nom d'hôte des noms du principal du service (c'est-à-dire qu'elles NE DOIVENT PAS utiliser des interrogations de DNS non sécurisées pour transposer un nom en un autre pour déterminer la partie hôte du nom de principal avec lequel on veut communiquer). Dans un environnement sans service de nom sécurisé, les auteurs d'applications PEUVENT ajouter un nom de domaine configuré de façon statique à des noms d'hôte non qualifiés avant de passer le nom aux mécanismes de sécurité, mais ils ne devraient pas faire plus que cela. Les facilités de service de nom sécurisées, si elles sont disponibles, pourraient être crues pour la canonisation de nom d'hôte, mais une telle canonisation par le client NE DEVRAIT PAS être exigée par les mises en œuvre KDC.

Note de mise en œuvre : De nombreuses mises en œuvre courantes font à un certain degré la canonisation du nom de service fourni, souvent en utilisant DNS même si cela crée des problèmes de sécurité. Cependant, il n'y a pas de cohérence entre les mises en œuvre sur le point de savoir si le nom de service est ramené en minuscules ou si la résolution inverse est utilisée. Pour maximiser l'interopérabilité et la sécurité, les applications DEVRAIENT fournir des mécanismes de sécurité avec des noms qui résultent de l'alignement du nom entré par l'utilisateur en minuscules sans effectuer d'autre modification ou canonisation.

1.4 Autorisation

Comme service d'authentification, Kerberos donne le moyen de vérifier l'identité des principaux sur un réseau. L'authentification est habituellement utile principalement comme première étape dans le processus d'autorisation, en déterminant si un client peut utiliser un service, à quels objets le client est autorisé à accéder, et le type d'accès permis pour chacun. Kerberos ne fournit pas d'autorisation par lui-même. La possession d'un ticket client pour un service ne

joue que pour l'authentification du client auprès de ce service, et e l'absence d'une procédure séparée d'autorisation, une application ne devrait pas le considérer comme autorisant l'utilisation de ce service.

Des méthodes d'autorisation séparées PEUVENT être mises en œuvre comme fonctions d'accès spécifiques de l'application et peuvent utiliser des fichiers sur le serveur d'application, sur des accréditifs d'autorisation produits séparément comme ceux fondés sur des mandataires [Neu93], ou sur d'autres services d'autorisation. Les accréditifs d'autorisation authentifiés séparément PEUVENT être incorporés dans les données d'autorisation d'un ticket lorsque ils sont encapsulés par l'élément de données d'autorisation produit par le KDC.

Les applications ne devraient pas accepter la simple production d'un ticket de service par le serveur Kerberos (même par un serveur Kerberos modifié) comme accordant l'autorisation d'utiliser le service, car de telles applications peuvent devenir vulnérables au détournement de cette vérification d'autorisation dans un environnement où sont fournies d'autres options pour l'authentification d'application, ou si elles interopèrent avec d'autres KDC.

1.5 Extension de Kerberos sans rupture d'interopérabilité

Avec la croissance de la base de mises en œuvre déployées de Kerberos, l'extension de Kerberos devient plus importante. Malheureusement, certaines extensions du protocole Kerberos existant créent des problèmes d'interopérabilité à cause d'incertitudes au sujet du traitement de certaines options d'extensibilité par certaines mises en œuvre. Ce paragraphe comporte des lignes directrices qui permettront aux futures mises en œuvre de conserver l'interopérabilité.

Kerberos fournit un mécanisme général pour l'extensibilité de protocole. Certains messages de protocole contiennent des tous typés – sous messages qui contiennent une chaîne d'octets avec un entier qui définit comment interpréter la chaîne d'octets. Les types d'entiers sont enregistrés centralement, mais ils peuvent être utilisés à la fois pour les extensions de fabricant et pour les extensions normalisées par l'IETF.

Dans le présent document, le mot "extension" se réfère aux extensions en définissant un nouveau type à insérer dans un trou typé existant dans un message de protocole. Il ne se réfère pas à l'extension par addition de nouveaux champs aux types ASN.1, sauf si le texte le mentionne explicitement.

1.5.1 Compatibilité avec la RFC 1510

Noter que les formats de message Kerberos existants ne sont pas prêts à être étendus par l'ajout de champs aux types ASN.1. L'envoi de champs supplémentaires résulte souvent en l'élimination de la totalité du message sans indication d'erreur. Les versions futures de la présente spécification donneront des lignes directrices pour s'assurer que les champs ASN.1 peuvent être ajoutés sans créer de problème d'interopérabilité.

En attendant, toute mise en œuvre nouvelle ou modifiée de Kerberos qui reçoit une extension de message inconnue DEVRAIT préserver le codage de l'extension mais ignore par ailleurs sa présence. Les receveurs NE DOIIVENT PAS refuser une demande simplement à cause de la présence d'une extension.

Il y a une exception à cette règle. Si un type d'élément de données d'autorisation inconnu est reçu par un serveur autre que le service d'allocation de tickets soit dans une AP-REQ soit dans un ticket contenu dans une AP-REQ, l'authentification DOIT alors échouer. Une des principales utilisations des données d'autorisation est de restreindre l'utilisation du ticket. Si le service ne peut pas déterminer si la restriction s'applique à ce service, il peut en résulter une daille dans la sécurité si le ticket peut être utilisé pour ce service. Les éléments d'autorisation qui sont facultatifs DEVRAIENT être inclus dans l'élément AD-IF-RELEVANT.

Le service d'allocation de tickets DOIT ignorer mais propager sur des tickets dérivés tous les types de données d'autorisation inconnus, sauf si ces types de données sont incorporés dans un élément MANDATORY-FOR-KDC, auquel cas la demande sera rejetée. Ce comportement est approprié parce que demander que le service d'allocation de tickets comprenne les types de données d'autorisation inconnus exigerait que le logiciel du KDC soit mis à jour pour comprendre les nouvelles restrictions de niveau application avant que les applications n'utilisent ces restrictions, rabaissant d'utilité des données d'autorisation au niveau d'un mécanisme de restriction de l'utilisation des tickets. Aucun problème de sécurité n'est créé parce que les services auxquels les tickets sont produits vont vérifier les données d'autorisation.

Note de mise en œuvre : De nombreuses mises en œuvre de la RFC 1510 ignorent les éléments de données

d'autorisation inconnus. Dépendre de la mise en œuvre pour respecter les restrictions sur les données d'autorisation peut créer une faiblesse pour la sécurité.

1.5.2 Envoi de messages extensibles

Il faut veiller à s'assurer que les vieilles mises en œuvre puissent comprendre les messages qui leur sont envoyés, même si elles ne comprennent pas une extension utilisée. Sauf si l'envoyeur sait qu'une extension est prise en charge, l'extension ne peut pas changer la sémantique du cœur de message ou d'extensions définies précédemment.

Par exemple, une extension incluant les informations de clé nécessaires pour déchiffre la partie chiffrée d'une KDC-REP ne pourrait être utilisée que dans des situations où le receveur est connu comme prenant en charge l'extension. Et donc, lors de la conception de telles extensions, il est important de donner les moyens au receveur de notifier à l'envoyeur la prise en charge de l'extension. Par exemple, dans le cas d'une extension qui change la clé de réponse de KDC-REP, le client pourrait indiquer la prise en charge de l'extension en incluant un élément padata dans la séquence AS-REQ. Le KDC ne devrait utiliser l'extension que si cet élément padata est présent dans la AS-REQ. Même si la politique exige l'utilisation de l'extension, il est préférable de retourner une erreur en indiquant que l'extension est exigée que d'utiliser l'extension alors que le receveur peut ne pas la prendre en charge. Corriger le logiciel des mises en œuvre qui n'interopèrent pas est plus facile quand des signalisations d'erreur sont retournées.

1.6 Hypothèses sur l'environnement

Kerberos impose quelques hypothèses sur l'environnement dans lequel il peut fonctionner de façon appropriée, qui sont les suivantes :

* Les attaques de "déni de service" ne sont pas résolues avec Kerberos. Il y a des endroits dans les protocoles où un intrus peut empêcher une application de participer aux étapes d'authentification appropriées. La détection et la solution à de telles attaques (dont certaines peuvent apparaître comme étant des modes d'échec "normal" pour le système) sont habituellement laissées à l'appréciation des administrateurs et utilisateurs humains.

* Les principaux DOIVENT garder secrètes leurs clés secrètes. Si un intrus s'empare d'une façon ou d'une autre de la clé d'un principal, il sera capable de se faire passer pour se principal ou de se déguiser en n'importe quel serveur du principal légitime.

* Les attaques de "mot de passe deviné" ne sont pas résolues par Kerberos. Si un utilisateur choisit un mot de passe faible, il est possible à un agresseur de monter avec succès une attaque de dictionnaire hors ligne en essayant avec persévérance de déchiffrer, avec des entrées successives à partir d'un dictionnaire, les messages obtenus qui sont chiffrés avec une clé dérivée du mot de passe de l'utilisateur.

* Chaque hôte sur le réseau DOIT avoir une horloge qui est "synchronisée approximativement" à l'heure des autres hôtes ; cette synchronisation est utilisée pour réduire les besoins d'enregistrement des serveurs d'application lorsqu'ils refont effectivement la détection. Le degré "d'approximation" peut être configuré serveur par serveur, mais il est normalement de l'ordre de 5 minutes. Si les horloges sont synchronisées par le réseau, le protocole de synchronisation d'horloge DOIT lui-même être sécurisé contre les attaquants du réseau.

* Les identifiants de principal ne sont pas recyclés à court terme. Un contrôle de mode d'accès normal va utiliser des listes de contrôle d'accès (ACL) pour accorder les permissions à des principaux particuliers. Si une entrée d'ACL périmée subsiste pour un principal supprimé et si l'identifiant du principal est réutilisé, le nouveau principal va hériter des droits spécifiés dans l'entrée d'ACL périmée. On supprime le danger d'un accès par inadvertance en ne réutilisant pas les identifiants de principal.

1.7 Glossaire

Ci-dessous figure une liste des termes utilisés dans le présent document.

Authentification
Vérifier l'identité revendiquée par un principal.

En-tête d'authentification
Enregistrement qui contient un ticket et un authentifiant à présenter à un serveur au titre du processus

d'authentification.

Chemin d'authentification
Séquence de domaines intermédiaires traversés dans le processus d'authentification lors de la communication d'un domaine à l'autre.

Authentifiant
Enregistrement contenant des informations dont on peut montrer qu'elles ont été générées récemment en utilisant la clé de session connue seulement du client et du serveur.

Autorisation
Processus pour déterminer si un client a la permission d'utiliser un service, à quels objets le client a la permission d'accès et le type d'accès permis pour chacun.

Capacité
Jeton qui accorde au porteur la permission d'accéder à un objet ou service. Dans Kerberos, ce peut être un ticket dont l'utilisation est restreinte par le contenu du champ de données d'autorisation, mais qui ne donne pas d'adresse réseau, avec la clé de session nécessaire pour utiliser le ticket.

Texte chiffré
Résultat d'une fonction de chiffrement. Le chiffrement transforme le texte clair en texte chiffré.

Client
Processus qui utilise un service réseau au nom d'un utilisateur. Noter que dans certains cas, un serveur peut être lui-même un client de quelque autre serveur (par exemple, un serveur d'impression peut être un client d'un serveur de fichiers).

Accréditifs
Un ticket plus la clé de session secrète nécessaire pour utiliser ce ticket avec succès dans un échange d'authentification.

Type de chiffrement (etype)
Associé à des données chiffrées; un type de chiffrement identifie l'algorithme utilisé pour chiffrer les données et est utilisé pour choisir l'algorithme approprié pour déchiffrer les données. Les étiquettes de type de chiffrement sont communiquées dans d'autres messages pour énumérer les algorithmes qui sont souhaités, pris en charge, préférés, ou permis en utilisation pour le chiffrement des données entre les parties. Cette préférence est combinée aux informations et politiques locales pour choisir l'algorithme à utiliser.

KDC *(Key Distribution Center)*
Centre de distribution de clés. Service réseau qui fournit les tickets et les clés de sessions temporaires ; ou instance de ce service ou l'hôte sur lequel il fonctionne. Le KDC sert à la fois le ticket initial et les demandes de ticket d'allocation de ticket. La portion de ticket initial est parfois appelée serveur (ou service) d'authentification. La portion de ticket d'allocation de ticket est parfois appelée serveur (ou service) d'allocation de ticket.

Kerberos
Nom donné au service d'authentification du Projet Athéna, protocole utilisé par ce service, ou code utilisé pour mettre en œuvre le service d'authentification. Le nom vient de celui du chien à trois têtes qui garde l'Hadès.

Numéro de version de clé (kvno, *Key Version Number*)
Étiquette associée aux données chiffrées qui identifie quelle clé a été utilisée pour le chiffrement lorsque une clé à longue durée associée à un principal change au fil du temps. Il est utilisé durant la transition vers une nouvelle clé de sorte que la partie qui déchiffre un message puisse dire si les données ont été chiffrées avec la vieille ou la nouvelle clé.

Texte en clair
Entrée dans une fonction de chiffrement ou sortie d'une fonction de déchiffrement. Le déchiffrement transforme le texte chiffré en texte en clair.

Principal
Entité client ou serveur nommée qui participe à une communication réseau, avec un nom qui est considéré comme canonique.

Identifiant de principal
Nom canonique utilisé pour identifier de façon univoque chaque différent principal.

Sceau
Pour chiffrer un enregistrement contenant plusieurs champs de telle sorte que les champs ne puissent pas être remplacés individuellement sans connaissance de la clé de chiffrement ou sans laisser de traces d'altération.

Clé secrète
Clé de chiffrement partagée par un principal et le KDC, distribuée en dehors des limites du système, avec une durée de vie longue. Dans le cas du principal d'un utilisateur humain, la clé secrète PEUT être déduite d'un mot de passe.

Serveur
Principal particulier qui fournit une ressource aux clients réseau. Le serveur est parfois appelé serveur d'application.

Service
Ressource fournie aux clients réseau ; souvent fournie par plus d'un serveur (par exemple, un service de fichier distant).

Clé de session
Clé de chiffrement temporaire utilisée entre deux principaux, avec une durée de vie limitée à la durée d'une seule "session" de connexion. Dans le système Kerberos, une clé de session est générée par le KDC. La clé de session est distincte de la sous-clé de session, décrite ci-après.

Sous-clé de session
Clé de chiffrement temporaire utilisée entre deux principaux, choisie et échangée par les principaux en utilisant la clé de session, et avec une durée de vie limitée à la durée d'une seule association. La sous-clé de session est aussi appelée sous-clé.

Ticket
Enregistrement qui aide un client à s'authentifier auprès d'un serveur ; il contient l'identité du client, une clé de session, un horodatage, et d'autres informations, toutes scellées en utilisant la clé secrète du serveur. Il ne sert qu'à authentifier un client lorsqu'il est présenté avec un authentifiant frais.

2 Utilisations et demandes de fanion de ticket

Chaque ticket Kerberos contient un ensemble de fanions qui sont utilisés pour indiquer les attributs de ce ticket. La plupart des fanions peuvent être demandés par un client lorsque le ticket est obtenu ; certains sont automatiquement activés et désactivés en tant que de besoin par un serveur Kerberos. Les paragraphes qui suivent explique la signification des divers fanions et donnent des exemples des raisons de leur utilisation. À l'exception du fanion INVALID, les clients DOIVENT ignorer les fanions de ticket qu'ils ne reconnaissent pas. Les KDC DOIVENT ignorer les options KDC qui ne sont pas reconnues. Certaines mises en œuvre de la RFC 1510 sont connues pour rejeter les options de KDC inconnues, de sorte que les clients peuvent avoir besoin de renvoyer une demande sans nouvelles options KDC si la demande a été rejetée alors qu'elle avait été envoyée avec des options ajoutées depuis la RFC 1510. Comme les nouveaux KDC ignoreront les options inconnues, les clients DOIVENT confirmer que le ticket retourné par le KDC satisfait leurs besoins.

Noter qu'il n'est en général pas possible de déterminer si une option n'a pas été satisfaite parce qu'elle n'a pas été comprise ou si elle a été rejetée à cause de la configuration ou de la politique. Lors de l'ajout d'une nouvelle option au protocole Kerberos, les concepteurs devraient examiner si la distinction est importante pour leur option. Si elle l'est, il faut fournir un mécanisme pour que le KDC retourne une indication comme quoi l'option a été comprise mais rejetée, dans la spécification de l'option. Souvent dans de tels cas, le mécanisme doit être suffisamment tolérant pour permettre qu'une erreur ou cause soit retournée.

2.1 Tickets initial, pré-authentifié, et matériel authentifié

Le fanion INITIAL indique qu'un ticket a été produit en utilisant le protocole d'AS, plutôt que produit sur la base d'un TGT. Les serveurs d'application qui veulent demander la preuve de la connaissance de la clé secrète d'un client (par exemple, un programme à changement de mot de passe) peut insister pour que ce fanion soit établi dans tous les tickets qu'il accepte, et peut donc être assuré que la clé du client a été récemment présentée au serveur d'authentification.

Les fanions PRE-AUTHENT *(pré-authentifié)* et HW-AUTHENT *(matériel-authentifié)* donnent des informations supplémentaires sur l'authentification initiale, que le ticket en cours ait été produit directement (auquel cas le fanion INITIAL sera établi aussi) ou qu'il soit produit sur la base d'un TGT (auquel cas le fanion INITIAL n'est pas mis, mais les fanions PRE-AUTHENT et HW-AUTHENT sont ramenés du TGT).

2.2 Tickets invalides

Le fanion INVALID indique qu'un ticket est invalide. Les serveurs d'application DOIVENT rejeter les tickets qui ont ce fanion. Un ticket postdaté sera produit dans cette forme. Les tickets invalides DOIVENT être validés par le KDC avant utilisation, en étant présentés au KDC dans une demande TGS avec l'option VALIDATE spécifiée. Le KDC ne validera le ticket qu'après que son heure de début soit passée. La validation est exigée de sorte que les tickets postdatés qui ont été volés avant leur heure de début puissent être rendus invalides de façon permanente (grâce à un mécanisme de liste noire) (voir au paragraphe 3.3.3.1).

2.3 Tickets renouvelables

Les applications peuvent désirer détenir des tickets qui soient valides pour de longues durées. Cependant, cela peut exposer leurs accréditifs à des menaces de vol pour des durées également longues, et ces accréditifs volés seraient valides jusqu'à l'heure d'expiration du ou des tickets. Utiliser simplement des tickets à courte durée de vie et en obtenir périodiquement de nouveaux exige que le client ait un accès à long terme à sa clé secrète, ce qui présente un risque encore plus grand. Les tickets renouvelables peuvent être utilisés pour atténuer les conséquences d'un vol. Les tickets renouvelables ont deux "heure d'expiration" : la première est quand l'instance actuelle du ticket expire, et la seconde est la valeur permissible la plus tardive pour une heure d'expiration individuelle. Un client d'application doit périodiquement (c'est-à-dire, avant son expiration) présenter au KDC un ticket renouvelable, avec l'option RENEW établie dans la demande au KDC. Le KDC va produire un nouveau ticket avec une nouvelle clé de session et une heure d'expiration plus tardive. Tous les autres champs du ticket sont laissés inchangés par le processus de renouvellement. Lorsque arrive l'heure d'expiration la plus tardive permissible, le ticket est expiré de façon permanente. À chaque renouvellement, le KDC PEUT consulter une liste noire pour déterminer si le ticket a été noté volé depuis son dernier renouvellement ; il refusera de renouveler les tickets volés, et donc la durée de vie utilisable des tickets volés est réduite.

Le fanion RENEWABLE dans un ticket n'est normalement interprété que par le service d'allocation de tickets (exposé au paragraphe 3.3). Il peut habituellement être ignoré par les serveurs d'application. Cependant, certains serveurs d'application particulièrement soigneux PEUVENT interdire les tickets renouvelables.

Si un ticket renouvelable n'est pas renouvelé à son heure d'expiration, le KDC ne renouvellera pas le ticket. Le fanion RENEWABLE est rétabli par défaut, mais un client PEUT demander qu'il soit établi en mettant l'option RENEWABLE dans le message KRB_AS_REQ. Si il est établi, le champ renew-till *(renouveler jusqu'à)* dans le ticket contient l'heure après laquelle le ticket ne peut plus être renouvelé.

2.4 Tickets postdatés

Les applications peuvent à l'occasion avoir besoin d'obtenir des tickets à utiliser beaucoup plus tard ; par exemple, un système de soumission par lots aura besoin de tickets valides au moment où le lot est à traiter. Cependant, il est dangereux de détenir des tickets valides dans une file d'attente de lots, car ils seront plus longtemps en ligne et plus enclins à se faire voler. Les tickets postdatés fournissent le moyen d'obtenir ces tickets du KDC au moment de la soumission de la tâche, mais de les laisser "dormants" jusqu'à ce qu'ils soient activés et validés par une demande ultérieure du KDC. Si un vol de ticket devait être rapporté dans l'intervalle, le KDC refuserait de valider le ticket, et le voleur sera floué.

Le fanion MAY-POSTDATE *(peut postdater)* dans un ticket n'est normalement interprété que par le service d'allocation de tickets. Il peut être ignoré par les serveurs d'application. Ce fanion DOIT être établi dans un TGT afin de produire un ticket postdaté sur la base du ticket présenté. Il est rétabli par défaut ; un client PEUT le demander en établissant l'option ALLOW-POSTDATE *(permettre postdate)* dans le message KRB_AS_REQ. Ce fanion ne permet pas à un client d'obtenir un TGT postdaté ; les TGT postdatés ne peuvent être obtenus qu'en demandant le postdatage dans le message KRB_AS_REQ. La durée de vie (heure de fin – heure de début) d'un ticket postdaté sera la durée de vie restante du TGT au moment de la demande, sauf si l'option RENEWABLE *(renouvelable)* est aussi établie, auquel cas elle peut être la durée de vie complète (heure de fin – heure de début) du TGT. Le KDC PEUT limiter de combien

un ticket peut être postdaté.

Le fanion POSTDATED indique qu'un ticket a été postdaté. Le serveur d'application peut vérifier le champ authtime dans le ticket pour voir quand est survenue l'authentification originale. Certains services PEUVENT choisir de rejeter les tickets postdatés, ou ils peuvent ne les accepter que dans une certaine période après l'authentification originale. Lorsque le KDC produit un ticket POSTDATED, il sera aussi marqué INVALID, de sorte que le client d'application DOIT présenter le ticket au KDC pour validation avant utilisation.

2.5 Tickets mandatables et mandataires

À certains moments, il peut être nécessaire qu'un principal permette à un service d'effectuer une opération en son nom. Le service doit être capable de prendre l'identité du client, mais seulement pour un objet particulier. Un principal peut permettre à un service de faire cela en lui accordant un mandat *(proxy)*.

Le processus de délivrance d'un mandat en utilisant les fanions mandat et mandatable est utilisé pour fournir des accréditifs à utiliser avec des services spécifiques. Bien que ce soit conceptuellement aussi un mandat, les utilisateurs qui souhaitent déléguer leur identité dans une forme utilisable à tous propos DOIVENT utiliser le mécanisme de transmission de ticket décrit au paragraphe suivant pour transmettre un TGT.

Le fanion PROXIABLE *(mandatable)* dans un ticket n'est normalement interprété que par le service d'allocation de tickets. Il peut être ignoré par les serveurs d'application. Lorsqu'il est établi, ce fanion dit au serveur de délivrance de tickets qu'il est d'accord pour produire un nouveau ticket (mais pas un TGT) avec une adresse réseau différente fondée sur ce ticket. Ce fanion est établi s'il est demandé par le client à l'authentification initiale. Par défaut, le client demandera qu'il soit établi lorsqu'il demande un TGT, et qu'il soit rétabli lorsqu'il demande tout autre ticket.

Ce fanion permet à un client de passer au serveur mandat d'effectuer une demande distante en son nom (par exemple, un client de service d'impression peut donner mandat au serveur d'impression d'accéder aux fichiers du client sur un serveur de fichiers particulier afin de satisfaire à une demande d'impression).

Afin de compliquer l'utilisation des accréditifs volés, les tickets Kerberos sont souvent valides pour les seules adresses réseau spécifiquement incluses dans le ticket, mais il est permis, comme option de politique de permettre des demandes et de produire des tickets sans aucune adresse réseau spécifiée. Lorsqu'il accorde un mandat, le client DOIT spécifier la nouvelle adresse réseau à partir de laquelle le mandat sera utilisé ou indiquer que la mandat est produit pour être utilisé sur toute adresse.

Le fanion PROXY est établi dans un ticket par le TGS lorsqu'il produit un ticket mandat. Les serveurs d'application PEUVENT vérifier ce fanion ; et en option, ils PEUVENT demander une authentification supplémentaire de la part de l'agent qui présente le mandat afin de fournir une trace d'audit.

2.6 Tickets transmissibles

La transmission d'authentification est une instance de mandat dans laquelle le service qui est accordé est l'utilisation complète de l'identité du client. Un exemple d'utilisation possible est celui d'un usager qui s'enregistre sur un système distant et veut l'authentification pour travailler à partir de ce système comme si l'enregistrement était local.

Le fanion FORWARDABLE *(transmissible)* dans un ticket n'est normalement interprété que par le service d'allocation de tickets. Il peut être ignoré par les serveurs d'application. Le fanion FORWARDABLE a une interprétation similaire à celle du fanion PROXIABLE, sauf que les TGT peuvent aussi être produits avec des adresses réseau différentes. Ce fanion est rétabli par défaut, mais les utilisateurs PEUVENT demander qu'il soit établi en mettant l'option FORWARDABLE dans la demande d'AS lorsqu'ils demandent les TGT initial.

Ce fanion permet la transmission d'authentification sans exiger que l'usager entre à nouveau un mot de passe. Si le fanion n'est pas établi, la transmission d'authentification n'est alors pas permise, mais le même résultat peut toujours être obtenu si l'usager s'engage dans l'échange d'AS, spécifie les adresses réseau demandées, et fournit un mot de passe.

Le fanion FORWARDED *(transmis)* est établi par le TGS lorsque un client présente un ticket avec le fanion FORWARDABLE établi et demande un ticket transmis en spécifiant l'option KDC FORWARDED et en fournissant un ensemble d'adresses pour le nouveau ticket. Il est aussi établi dans tous les tickets produits sur la base de tickets

ayant le fanion FORWARDED établit. Les serveurs d'application peuvent choisir de traiter les tickets FORWARDED différemment des tickets non marqués FORWARDED.

Si les tickets sans adresse sont transmis d'un système à un autre, les clients DEVRAIENT continuer d'utiliser cette option pour obtenir un nouveau TGT afin d'avoir des clés de session différentes sur les différents systèmes.

2.7 Vérification des politiques traversées

Dans Kerberos, le serveur d'application est responsable en dernier ressort de l'acceptation ou du rejet de l'authentification, et il DEVRAIT vérifier que seuls des KDC de confiance sont impliqués dans l'authentification d'un principal. Les champs de transit dans le ticket identifient quels domaines (et donc quels KDC) ont été impliqués dans le processus d'authentification, et un serveur d'application devrait normalement vérifier ce champ. Si l'un d'eux n'est pas de confiance pour authentifier le principal de client indiqué (probablement déterminé par une politique fondée sur le domaine), la tentative d'authentification DOIT être rejetée. La présence de KDC de confiance dans cette liste ne donne aucune garantie ; un KDC compromis peut avoir fabriqué la liste.

Bien que le serveur d'extrémité décide en dernier ressort si l'authentification est valide, le KDC pour le domaine du serveur d'extrémité PEUT appliquer une politique spécifique du domaine pour la validation du champ de transit et l'acceptation des accréditifs pour l'authentification de traversée de domaine. Lorsque le KDC applique de telles vérifications et accepte une telle authentification de traversée de domaine, il va établir le fanion TRANSITED-POLICY-CHECKED dans les tickets de service qu'il produit sur la base du TGT de traversée de domaine. Un client PEUT demander que les KDC ne vérifient pas les champs traversés en mettant le fanion DISABLE-TRANSITED-CHECK. Les KDC sont encouragés à respecter ce fanion mais ne sont pas obligés de le faire.

Les serveurs d'application DOIVENT soit faire eux-mêmes les vérifications de domaine traversé soit rejeter les tickets de traversée de domaine sans que TRANSITED-POLICY-CHECKED soit mis.

2.8 OK-as-Delegate

Pour certaines applications, un client peut avoir besoin de déléguer son autorité à un serveur pour agir en son nom en contactant d'autres services. Cela exige que le client transmette les accréditifs à un serveur intermédiaire. La capacité pour un client à obtenir un ticket de service pour un serveur n'apporte pas d'information au client sur la question de savoir si le serveur devrait être considéré comme de confiance pour accepter des accréditifs délégués. Le fanion OK-AS-DELEGATE donne à un KDC un moyen pour communiquer une politique de domaine local à un client sur le sujet de savoir si un serveur intermédiaire est de confiance pour accepter de tels accréditifs.

La copie des fanions de ticket dans la partie chiffrée de la réponse du KDC peut avoir le fanion OK-AS-DELEGATE établi pour indiquer au client que le serveur spécifié dans le ticket a été déterminé par la politique du domaine comme étant un récepteur convenable de délégation. Un client peut utiliser la présence de ce fanion pour l'aider à décider de déléguer des accréditifs (en accordant soit un mandat soit un TGT retransmis) à ce serveur. Il est acceptable d'ignorer la valeur de ce fanion. Lorsqu'il établit ce fanion, un administrateur devrait considérer la sécurité et le placement du serveur sur lequel va fonctionner le service, ainsi que si le service exige l'utilisation d'accréditifs délégués.

2.9 Autres options de KDC

Il y a trois options supplémentaires qui PEUVENT être établies dans une demande de KDC du client.

2.9.1 Renewable-OK

L'option RENEWABLE-OK *(d'accord pour renouveler)* indique que le client acceptera un ticket renouvelable si un ticket avec la durée de vie demandée ne peut pas être autrement fourni. Si un ticket avec la durée de vie demandée ne peut pas être fourni, le KDC PEUT alors produire un ticket renouvelable avec un renew-till *(renouveler jusqu'à)* égal à la fin de durée de vie demandée. La valeur du champ renew-till PEUT encore être ajustée par des limites déterminées par le site ou des limites imposées par le principal ou serveur individuel.

2.9.2 ENC-TKT-IN-SKEY

Dans sa forme de base, le protocole Kerberos prend en charge l'authentification dans un montage client-serveur et ne

convient pas bien pour l'authentification dans un environnement d'homologue à homologue parce que la clé à long terme de l'usager ne reste pas sur la station de travail après la connexion initiale. L'authentification de tels homologues peut être prise en charge par Kerberos dans sa variante d'usager à usager. L'option ENC-TKT-IN-SKEY prend en charge l'authentification d'usager à usager en permettant que le KDC produise un ticket de service chiffré en utilisant la clé de session provenant d'un autre TGT produite pour un autre usager. L'option ENC-TKT-IN-SKEY n'est honorée que par le service d'allocation de tickets. Elle indique que le ticket à produire pour le serveur est à chiffrer dans la clé de session à partir du second TGT supplémentaire fourni avec la demande. Voir les détails spécifiques au paragraphe 3.3.3.

2.9.3 Authentification de matériel sans mot de passe

L'option OPT-HARDWARE-AUTH indique que le client souhaite utiliser une forme d'authentification de matériel à la place de ou en plus du mot de passe du client ou autre clé de chiffrement à longue durée de vie. OPT-HARDWARE-AUTH n'est honoré que par le service d'authentification. Si il est pris en charge et admis par la politique, le KDC va retourner un code d'erreur de KDC_ERR_PREAUTH_REQUIRED et inclure les METHOD-DATA exigées pour effectuer une telle authentification.

3 Échanges de messages

Les paragraphes qui suivent décrivent les interactions entre les clients et serveurs réseau et les messages impliqués dans ces échanges.

3.1 L'échange de service d'authentification

Résumé

Direction du message	Type de message	Paragraphe
1. Client à Kerberos	KRB_AS_REQ	5.4.1
2. Kerberos à client	KRB_AS_REP ou	5.4.2
	KRB_ERROR	5.9.1

L'échange de service d'authentification (AS) entre le client et le serveur d'authentification Kerberos est initialisé par un client qui souhaite obtenir des accréditifs d'authentification pour un serveur donné mais ne détient pas actuellement d'accréditifs. Dans forme de base, la clé secrète du client est utilisée pour le chiffrement et le déchiffrement. Cet échange est normalement utilisé à l'initiation d'une session de connexion pour obtenir des accréditifs pour un serveur d'allocation de tickets (TGS, *Ticket-Granting Server*) qui seront utilisés ultérieurement pour obtenir des accréditifs pour d'autres serveurs (voir au paragraphe 3.3) sans plus avoir besoin d'utiliser la clé secrète du client. Cet échange est aussi utilisé pour demander des accréditifs pour des services qui ne doivent pas passer par le service d'allocation de tickets, mais exigent plutôt une connaissance de la clé secrète d'un principal, comme dans un service à mot de passe changeant (le service à mot de passe changeant refuse les demandes si le demandeur ne peut pas démontrer qu'il a connaissance du vieux mot de passe d'utilisateur ; exiger cette connaissance empêche les changements non autorisés de mot de passe par quelqu'un qui arrive dans une session où il n'est pas invité).

Cet échange ne fournit par lui-même aucune assurance de l'identité de l'usager. Pour authentifier un usager qui s'inscrit sur un système local, les accréditifs obtenus dans l'échange d'AS peuvent d'abord être utilisés dans un échange TGS pour obtenir des accréditifs pour un serveur local ; ces accréditifs doivent ensuite être vérifiés par un serveur local à travers l'achèvement réussi de l'échange client/serveur.

L'échange AS consiste en deux messages : KRB_AS_REQ du client à Kerberos, et KRB_AS_REP ou KRB_ERROR en réponse. Les formats de ces messages sont décrits aux paragraphes 5.4.1, 5.4.2, et 5.9.1.

Dans la demande, le client envoie (en clair) sa propre identité et l'identité du serveur pour lequel il demande les accréditifs, d'autres informations sur les accréditifs qu'il demande, et un nom occasionnel généré de façon aléatoire, qui peut être utilisé pour détecter des répétitions et pour associer les réponses aux demandes correspondantes. Ce nom occasionnel DOIT être généré de façon aléatoire par le client et mémorise pour vérification par rapport au nom occasionnel dans la réponse attendue. La réponse, KRB_AS_REP, contient un ticket que le client présentera au serveur, et une clé de session qui sera partagée par le client et le serveur. La clé de session et les informations supplémentaires sont chiffrées avec la clé secrète du client. La partie chiffrée du message KRB_AS_REP contient aussi le nom

occasionnel qui DOIT correspondre au nom occasionnel provenant du message KRB_AS_REQ.

Sans pré-authentification, le serveur d'authentification ne sait pas si le client est réellement le principal nommé dans la demande. Il envoie simplement une réponse sans savoir s'il sont les mêmes et sans s'en soucier. Ceci est acceptable parce que personne sauf le principal dont l'identité a été donnée dans la demande ne sera capable d'utiliser la réponse. Ses informations critiques sont chiffrées avec la clé du principal. Cependant, un attaquant peut envoyer un message KRB_AS_REQ pour obtenir du texte en clair connu afin d'attaquer la clé du principal. Et particulièrement si la clé se fonde sur un mot de passe, cela peut créer un danger pour la sécurité. Ainsi, la demande initiale prend en charge un champ facultatif qui peut être utilisé pour passer des informations supplémentaires dont il pourrait être besoin pour l'échange initial. Ce champ DEVRAIT être utilisé pour la pré-authentification, comme décrit aux paragraphes 3.1.1 et 5.2.7.

Diverses erreurs peuvent survenir ; elles sont indiquées par une réponse d'erreur (KRB_ERROR) à la place de la réponse KRB_AS_REP. Le message d'erreur n'est pas chiffré. Le message KRB_ERROR contient des informations qui peuvent être utilisées pour l'associer au message auquel il répond. Le contenu du message KRB_ERROR n'est pas protégé en intégrité. Et donc, le client ne peut pas détecter les répétitions, contrefaçons, ou modifications. Une solution à ce problème sera donnée dans une prochaine version du protocole.

3.1.1 Génération du message KRB_AS_REQ

Le client peut spécifier un certain nombre d'options dans la demande initiale. Parmi ces options figure la question de savoir si la pré-authentification est à effectuer ; si le ticket demandé est renouvelable, mandatable, ou transmissible ; si il devrait être postdaté ou permettre le postdatage de tickets déduits ; et si un ticket renouvelable sera accepté à la place d'un ticket non renouvelable si la date d'expiration du ticket demandé ne peut pas être satisfaite par un ticket non renouvelable (du fait de contraintes de configuration).

Le client prépare le message KRB_AS_REQ et l'envoie au KDC.

3.1.2 Réception du message KRB_AS_REQ

Si tout va bien, le traitement du message KRB_AS_REQ résultera en la création d'un ticket que le client présentera au serveur. Le format du ticket est décrit au paragraphe 5.3.

Parce que Kerberos peut fonctionner sur des transports non fiables comme UDP, le KDC DOIT être prêt à retransmettre des réponses dans le cas où elles sont perdues. Si un KDC reçoit une demande identique à l'une de celles qu'il a récemment traitées avec succès, le KDC DOIT répondre par un message KRB_AS_REP plutôt qu'une erreur de répétition. Afin de réduire la quantité de texte chiffré à disposition d'un attaquant potentiel, les KDC PEUVENT envoyer la même réponse que générée lors du premier traitement de la demande. Les KDC DOIVENT obéir à ce comportement de répétition même si le transport réel utilisé est fiable.

3.1.3 Génération du message KRB_AS_REP

Le serveur d'authentification cherche dans sa base de données les principaux de client et de serveur nommés dans le message KRB_AS_REQ, et en extrait les clés respectives. Si le principal client demandé nommé dans la demande est inconnu parce qu'il n'existe pas dans la base de données de principaux du KDC, un message d'erreur est retourné avec un KDC_ERR_C_PRINCIPAL_UNKNOWN.

Si il lui est demandé de faire ainsi, le serveur pré-authentifie la demande, et si la vérification de pré-authentification échoue, un message d'erreur est retourné avec le code KDC_ERR_PREAUTH_FAILED. Si la pré-authentification est exigée, mais n'était pas présente dans la demande, un message d'erreur avec le code KDC_ERR_PREAUTH_REQUIRED est retourné, et un objet METHOD-DATA sera mémorisé dans le champ e-data du message KRB-ERROR pour spécifier quels mécanismes de pré-authentification sont acceptables. Cela inclura habituellement les éléments PA-ETYPE-INFO et/ou PA-ETYPE-INFO2 comme décrits ci-dessous. Si le serveur ne peut pas s'accommoder d'un des types de chiffrement demandé par le client, un message d'erreur avec le code KDC_ERR_ETYPE_NOSUPP est retourné. Autrement, le KDC génère une clé de session "aléatoire", ce qui signifie, entre autres choses, qu'il devrait être impossible de deviner la prochaine clé de session sur la base de la connaissance des clés de session passées. Bien que cela puisse être réalisé avec un générateur de nombres pseudo-aléatoires si il se fonde sur les principes cryptographiques, il est plus souhaitable d'utiliser un générateur de nombres vraiment aléatoires, comme un de ceux fondés sur la mesure de phénomènes physiques aléatoires. Voir la [RFC4086] pour un exposé de

fond sur la notion d'aléatoire.

En réponse à une demande d'AS, si il y a plusieurs clés de chiffrement inscrites pour un client dans la base de données Kerberos, le champ etype de la demande d'AS est utilisé par le KDC pour choisir la méthode de chiffrement à utiliser pour protéger la partie chiffrée du message KRB_AS_REP qui est envoyé au client. Si il y a plus d'un type de chiffrement fort dans la liste etype, le KDC DEVRAIT utiliser le premier etype fort valide pour lequel une clé de chiffrement est disponible.

Lorsque la clé d'usager est générée à partir d'un mot de passe ou d'une phrase de passe, la fonction de chaîne à clé pour le type de clé de chiffrement particulier est utilisé, comme spécifié dans la [RFC3961]. La valeur de sel et les paramètres supplémentaires pour la fonction de chaîne à clé ont des valeurs par défaut (spécifiées respectivement à la Section 4 et par la spécification du mécanisme de chiffrement) qui peuvent être subrogées par les données de pré-authentification (PA-PW-SALT, PA-AFS3-SALT, PA-ETYPE-INFO, PA-ETYPE-INFO2, etc.). Comme le KDC est présumé mémoriser une copie seulement de la clé résultante, ces valeurs ne devraient pas être changées pour des clés fondées sur des mots de passe excepté lorsqu'on change la clé du principal.

Lorsque le serveur d'AS va inclure les données de pré-authentification dans une KRB-ERROR ou dans une AS-REP, il DOIT utiliser PA-ETYPE-INFO2, et non PA-ETYPE- INFO, si le champ etype de la AS-REQ du client comporte au moins un type de chiffrement "plus récent". Autrement (lorsque le champ etype de l'AS-REQ du client ne comporte aucun type de chiffrement "plus récent"), il DOIT envoyer les deux PA-ETYPE-INFO2 et PA-ETYPE-INFO (tous deux avec une entrée pour chaque enctype). Un enctype "plus récent" est tout enctype spécifié officiellement en premier concurremment ou ultérieurement à la publication de la présente RFC. Les enctypes DES, 3DES, ou RC4 et tous ceux définis dans la [RFC1510] ne sont pas des enctypes "plus récents".

Il n'est pas possible de générer une clé d'usager de façon fiable selon une phrase de passe donnée sans contacter le KDC, car on ne pourra pas savoir si des valeurs de sel ou de paramètres de remplacement sont nécessaires.

Le KDC va essayer d'allouer le type de la clé de session aléatoire d'après la liste des méthodes dans le champ etype. Le KDC va sélectionner le type approprié en utilisant la liste des méthodes fournie et les informations tirées de la base de données Kerberos qui indique les méthodes de chiffrement acceptables pour le serveur d'application. Le KDC ne produira pas de tickets avec un type faible de chiffrement de clé de session.

Si l'heure de début demandée est absente, indique une heure passée, ou est dans la fenêtre de biais d'horloge acceptable pour le KDC et si l'option POSTDATE n'a pas été spécifiée, l'heure de début du ticket est réglée à l'heure en cours du serveur d'authentification. Si il indique une heure future au delà du biais d'horloge acceptable, mais si l'option POSTDATED n'a pas été spécifiée, l'erreur KDC_ERR_CANNOT_POSTDATE est alors retournée. Autrement, l'heure de début demandée est vérifiée par rapport à la politique du domaine local (l'administrateur peut décider d'interdire certain types ou gammes de tickets postdatés), et si l'heure de début du ticket est acceptable, il est réglé comme demandé, et le fanion INVALID est mis dans le nouveau ticket. Le ticket postdaté DOIT être validé avant usage en le présentant au KDC après que l'heure de début a été atteinte.

L'heure d'expiration du ticket sera réglée au plus tôt de l'heure de fin demandée et de l'heure déterminée par la politique locale, éventuellement en utilisant des facteurs spécifiques du domaine ou du principal. Par exemple, l'heure d'expiration PEUT être réglée au plus tôt de ce qui suit :

* L'heure d'expiration (endtime) demandée dans le message KRB_AS_REQ.
* L'heure de début du ticket plus la durée de vie maximum admissible associée au principal client d'après la base de données du serveur d'authentification.
* L'heure de début du ticket plus la durée de vie maximum admissible associée au principal serveur.
* L'heure de début du ticket plus la durée de vie maximum réglée par la politique du domaine local.

Si l'heure d'expiration demandée moins l'heure de début (comme déterminé ci-dessus) est inférieure à la durée de vie minimum déterminée par un site, un message d'erreur avec le code KDC_ERR_NEVER_VALID est retourné. Si l'heure d'expiration demandée pour le ticket excède ce qui a été déterminé ci-dessus, et si l'option 'RENEWABLE-OK' était demandée, le fanion 'RENEWABLE' est alors mis dans le nouveau ticket, et la valeur renouveler-jusqu'à est mise comme si l'option 'RENEWABLE' était demandée (les noms de champ et d'option sont décrits au paragraphe 5.4.1).

Si l'option RENEWABLE avait été demandée ou si l'option RENEWABLE-OK était établie et qu'un ticket renouvelable va être produit, le champ renouveler-jusqu'à PEUT être réglé au plus tôt de :

* Sa valeur demandée.
* L'heure de début du ticket plus le minimum des deux durées de vie maximum associées aux entrées de base de

données des principaux.
* L'heure de début du ticket plus la durée de vie renouvelable maximum réglée par la politique du domaine local.

Le champ des fanions du nouveau ticket aura les options suivantes établies si ils ont été demandés et si la politique du domaine local le permet : FORWARDABLE, MAY-POSTDATE, POSTDATED, PROXIABLE, RENEWABLE. Si le nouveau ticket est postdaté (l'heure de début est à venir), son fanion INVALID sera aussi établi.

Si tout ce qui précède réussit, le serveur va chiffrer la partie de texte chiffré du ticket en utilisant la clé de chiffrement extraite de l'enregistrement du principal serveur dans la base de données Kerberos en utilisant le type de chiffrement associé à la clé du principal serveur. (Ce choix N'EST PAS affecté par le champ etype dans la demande.) Il formate alors un message KRB_AS_REP (voir au paragraphe 5.4.2), en copiant les adresses de la demande dans le caddr de la réponse, en plaçant toutes les données de pré-authentification demandées dans le padata de la réponse, et en chiffrant la partie de texte chiffré dans la clé du client en utilisant une méthode de chiffrement acceptable demandée dans le champ etype de la demande, ou dans une clé spécifiée par le mécanisme de pré-authentification utilisé.

3.1.4 Génération du message KRB_ERROR

Plusieurs erreurs peuvent survenir, et le serveur d'authentification répond en envoyant un message d'erreur, KRB_ERROR, au client, avec les champs code d'erreur et e-text réglés aux valeurs appropriées. Le contenu du message d'erreur et ses détails sont décrits au paragraphe 5.9.1.

3.1.5 Réception du message KRB_AS_REP

Si le type du message de réponse est KRB_AS_REP, le client vérifie alors que les champs cname et crealm de la portion de texte en clair de la réponse correspondent à ce qui était demandé. Si un champ padata est présent, il peut être utilisé pour déduire la clé secrète appropriée pour déchiffrer le message. Le client déchiffre la partie chiffrée de la réponse en utilisant sa clé secrète et vérifie que le nom occasionnel dans la partie chiffrée correspond au nom occasionnel qu'il a fourni dans sa demande (pour détecter des répétitions). Il vérifie aussi que sname et srealm de la réponse correspondent à ceux de la demande (ou ont autrement des valeurs attendues), et que le champ d'adresse d'hôte est aussi correct. Il mémorise alors le ticket, la clé de session, les heures de début et d'expiration, et les autres informations pour un usage ultérieur. Le champ last-req (et le champ déconseillé key-expiration) tirés de la partie chiffrée de la réponse PEUVENT être vérifiés pour notifier à l'usager une expiration de clé imminente. Ceci permet au programme client de suggérer un remède, comme un changement de mot de passe.

À la validation du message KRB_AS_REP (par confrontation du nom occasionnel retourné avec celui envoyé dans le message KRB_AS_REQ), le client sait que l'heure en cours au KDC est celle lue dans le champ authtime de la partie chiffrée de la réponse. Le client a la faculté d'utiliser cette valeur pour la synchronisation d'horloge dans les messages suivants en enregistrant avec le ticket la différence (décalage) entre la valeur de authtime et l'horloge locale. Ce décalage peut alors être utilisé par le même usager pour ajuster l'heure lue sur l'horloge système lors de la génération des messages [DGT96].

Cette technique DOIT être utilisée lorsqu'on règle le biais d'horloge au lieu de changer directement l'horloge système, parce que la réponse du KDC n'est authentifiée qu'auprès de l'usager dont la clé secrète a été utilisée, mais pas auprès du système ou station de travail. Si l'horloge était ajustée, un attaquant en collusion avec un usager qui se connecte dans une station de travail pourraient se mettre d'accord sur un mot de passe, et il en résulterait une réponse de KDC qui serait correctement validée même si elle n'a pas pour origine un KDC de confiance pour la station de travail.

Le déchiffrement correct du message KRB_AS_REP n'est pas suffisant à l'hôte pour vérifier l'identité de l'usager ; l'usager et un attaquant pourraient coopérer pour générer un message de format KRB_AS_REP qui se déchiffre correctement mais ne serait pas du KDC approprié. Si l'hôte souhaite vérifier l'identité de l'usager, il DOIT exiger que l'usager présente des accréditifs d'application qui puissent être vérifiés en utilisant une clé secrète que l'hôte aura mémorisé en toute sécurité. Si ces accréditifs peuvent être vérifiés, l'identité de l'usager peut alors être assurée.

3.1.6 Réception du message KRB_ERROR

Si le type de message de réponse est KRB_ERROR, le client l'interprète alors comme une erreur et effectue toute tâche spécifique de l'application qui est nécessaire pour récupérer.

3.2 L'échange d'authentification client/serveur

Résumé

Direction du message	Type de message	Paragraphe
Client à serveur d'application	KRB_AP_REQ	5.5.1
[facultatif] serveur d'application à client	KRB_AP_REP ou	5.5.2
	KRB_ERROR	5.9.1

L'échange d'authentification client/serveur (CS) est utilisé par les applications réseau pour authentifier le client auprès du serveur et vice versa. Le client DOIT déjà avoir acquis des accréditifs pour le serveur en utilisant l'échange AS ou TGS.

3.2.1 Le message KRB_AP_REQ

KRB_AP_REQ contient des informations d'authentification qui DEVRAIENT faire partie du premier message dans une transaction authentifiée. Il contient un ticket, un authentifiant, et quelques informations de comptabilité supplémentaires (voir au paragraphe 5.5.1 le format exact). Le ticket est insuffisant par lui-même pour authentifier un client, car les tickets passent à travers le réseau en clair (les tickets contiennent à la fois une portion chiffrée et une portion non chiffrée, de sorte qu'ici, texte en clair se réfère à l'unité entière, qui peut être copiée à partir d'un message et répétée dans un autre sans aucune habileté cryptographique). L'authentifiant est utilisé pour empêcher la répétition invalide de tickets en prouvant au serveur que le client connaît la clé de session du ticket et est donc fondé à utiliser le ticket. Le message KRB_AP_REQ est par ailleurs appelé un "en-tête d'authentification".

3.2.2 Génération d'un message KRB_AP_REQ

Lorsque un client souhaite initier l'authentification auprès d'un serveur, il obtient (soit avec une mémoire cache d'accréditifs, l'échange d'AS, ou l'échange TGS) un ticket et une clé de session pour le service désiré. Le client PEUT réutiliser tous les tickets qu'il détient jusqu'à leur expiration. Pour utiliser un ticket, le client construit un nouvel authentifiant à partir de l'heure du système et de son nom, et facultativement à partir d'une somme de contrôle spécifique de l'application, un numéro de séquence initial à utiliser dans les messages KRB_SAFE ou KRB_PRIV, et/ou une sous-clé de session à utiliser dans les négociations pour une clé de session unique pour cette session particulière. Les authentifiants NE DOIVENT PAS être réutilisés et DEVRAIENT être rejeté si ils sont répétés sur un serveur. Noter que cela peut rendre les applications fondées sur des transports non fiables difficiles à coder correctement. Si le transport peut délivrer des messages dupliqués, un nouvel authentifiant DOIT être généré pour chaque essai, ou le serveur d'application DOIT faire correspondre demandes et réponses et répéter la première répétition en réponse à un duplicata détecté.

Si un numéro de séquence doit être inclus, il DEVRAIT être choisi de façon aléatoire de sorte que même après l'échange de nombreux messages il n'y ait aucune probabilité de collision avec un autre numéro de séquence utilisé.

Le client PEUT indiquer une exigence d'authentification mutuelle ou l'utilisation d'un ticket fondé sur une clé de session (pour l'authentification d'usager à usager, voir le paragraphe 3.7) en mettant le ou les fanions appropriés dans le champ ap-options du message.

L'authentifiant est chiffré dans la clé de session et combiné avec le ticket pour former le message KRB_AP_REQ, qui est alors envoyé au serveur d'extrémité avec toutes informations supplémentaires spécifiques de l'application.

3.2.3 Réception du message KRB_AP_REQ

L'authentification se fonde sur l'heure en cours du serveur (les horloges DOIVENT être à peu près synchronisée), sur l'authentifiant, et sur le ticket. Plusieurs erreurs sont possibles. Si une erreur survient, on attend du serveur qu'il réponde un message KRB_ERROR au client. Ce message PEUT être encapsulé dans le protocole d'application si sa forme brute n'est pas acceptable pour le protocole. Le format des messages d'erreur est décrit au paragraphe 5.9.1.

L'algorithme de vérification des informations d'authentification est le suivant. Si le type de message n'est pas KRB_AP_REQ, le serveur retourne l'erreur KRB_AP_ERR_MSG_TYPE. Si la version de clé est indiquée par le ticket dans KRB_AP_REQ n'est pas une de celles que le serveur peut utiliser (par exemple, elle indique une vieille clé, et le serveur ne possède plus de copie de la vieille clé), l'erreur KRB_AP_ERR_BADKEYVER est retournée. Si le fanion

USE-SESSION-KEY est mis dans le champ ap-options, il indique au serveur l'utilisation de l'authentification d'usager à usager, et le chiffrement du ticket avec la clé de session provenant du TGT du serveur plutôt qu'avec la clé secrète du serveur. Voir au paragraphe 3.7 une description plus complète de l'effet de l'authentification d'usager à usager sur tous les messages dans le protocole Kerberos.

Le champ srealm dans la portion non chiffrée du ticket dans KRB_AP_REQ est utilisé pour spécifier quelle clé secrète devrait utiliser le serveur pour déchiffrer le ticket, parce qu'il est possible au serveur d'être enregistré dans plusieurs domaines,avec des clés différentes dans chacun. Le code d'erreur KRB_AP_ERR_NOKEY est retourné si le serveur n'a pas la clé appropriée pour déchiffrer le ticket.

Le ticket est déchiffré en utilisant la version de la clé du serveur spécifiée par le ticket. Si la routine de déchiffrement détecte une modification du ticket (chaque système de chiffrement DOIT fournir des sauvegardes pour détecter le texte chiffré modifié), l'erreur KRB_AP_ERR_BAD_INTEGRITY est retournée (il y a de bonnes chances que différentes clés aient été utilisées pour chiffrer et pour déchiffrer).

L'authentifiant est déchiffré en utilisant la clé de session extraite du ticket déchiffré. Si le déchiffrement montre qu'il a été modifié, l'erreur KRB_AP_ERR_BAD_INTEGRITY est retournée. Le nom et le domaine du client tirés du ticket sont comparés aux champs correspondants dans l'authentifiant. Si ils ne correspondent pas, l'erreur KRB_AP_ERR_BADMATCH est retournée ; ceci est normalement causé par une erreur client ou une tentative d'attaque. Les adresses dans le ticket (s'il en est) sont alors examinées à la recherche d'une adresse correspondant à l'adresse mentionnée par le système d'exploitation du client. Si aucune correspondance n'est trouvée, ou si le serveur insiste sur les adresses du ticket mais qu'aucune n'est présente dans le ticket, l'erreur KRB_AP_ERR_BADADDR est retournée. Si l'heure locale (du serveur) et l'heure du client dans l'authentifiant diffèrent de plus que le biais d'horloge admissible (par exemple, 5 minutes), l'erreur KRB_AP_ERR_SKEW est retournée.

À moins que serveur d'application ne fournisse ses propres moyens pour se protéger contre la répétition (par exemple, une séquence de mise en cause-réponse initiée par le serveur après authentification, ou l'utilisation d'une sous-clé de chiffrement générée par le serveur), le serveur DOIT utiliser une mémoire cache de répétition pour se souvenir de tous les authentifiants présentés pendant le biais d'horloge acceptable. Une analyse soigneuse du protocole et de la mise en œuvre de l'application est recommandée avant d'éliminer cette mémoire cache. La mémoire cache de répétition va mémoriser au moins les champ de nom du serveur, de nom du client, d'heure et de microsecondes tirés des authentifiants vus le plus récemment, et si un tuplet correspondant est trouvé, l'erreur KRB_AP_ERR_REPEAT est retournée. Noter qu'ici, le rejet est restreint aux authentifiants provenant du même principal vers le même serveur. Les autres principaux de clients qui communiquent avec le même principal de serveur ne devraient pas voir leurs authentifiants rejetés si les champs heure et microseconde se trouvent correspondre à d'autres authentifiants du client.

Si un serveur perd la trace des authentifiants présentés pendant le biais d'horloge admissible, il DOIT rejeter toutes les demandes jusqu'à la fin de l'intervalle de biais d'horloge, lui donnant l'assurance que tout authentifiant perdu ou répété va tomber en –dehors du biais d'horloge admissible et ne peut plus être répété avec succès. Si cela n'était pas fait, un attaquant pourrait subvertir l'authentification en enregistrant le ticket et l'authentifiant envoyés sur le réseau à un serveur et les rejouer à la suite d'un événement qui a causé la perte de trace d'authentifiants récemment vus par le serveur.

Note de mise en œuvre : Si un client génère plusieurs demandes au KDC avec le même horodatage, y compris le champ microsecondes, toutes les demandes reçues sauf la première seront rejetées comme répétitions. Cela peut arriver, par exemple, si la résolution de l'horloge du client est trop grossière. Les mises en œuvre client DEVRAIENT s'assurer que les horodatages ne sont pas réutilisés, éventuellement en incrémentant le champ microsecondes dans l'horodatage lorsque l'horloge retourne la même heure pour plusieurs demandes.

Si plusieurs serveurs (par exemple, différents services sur une machine, ou un seul service mis en œuvre sur plusieurs machines) partagent un principal de service (pratique qu'on ne recommande pas en général, mais dont on reconnaît qu'elle est utilisée dans certains cas), ils DOIVENT partager cette mémoire cache de répétition, ou bien le protocole d'application DOIT être conçu de façon à en éliminer le besoin. Noter que ceci s'applique à tous les services. Si un protocole d'application n'a pas de protection anti-répétition incorporée, un authentifiant utilisé avec un tel service pourrait être répété ultérieurement dans un service différent service avec le même principal de service mais sans protection contre la répétition, si le premier n'enregistre pas les informations d'authentifiant dans la mémoire cache de répétition ordinaire.

Si un numéro de séquence est fourni dans l'authentifiant, le serveur le sauvegarde pour utilisation ultérieure en traitant les messages KRB_SAFE et/ou KRB_PRIV. Si une sous clé est présente, le serveur la sauvegarde pour utilisation

ultérieure ou l'utilise pour aider à générer sont propre choix d'une sous-clé à retourner dans un message KRB_AP_REP.

Le serveur calcule l'age du ticket : heure locale (du serveur) moins l'heure de début dans le ticket. Si l'heure de début est plus tardive que l'heure en cours de plus que le biais d'horloge admissible, ou si le fanion INVALID est mis dans le ticket, l'erreur KRB_AP_ERR_TKT_NYV est retournée. Autrement, si l'heure en cours est plus tardive que l'heure de fin de plus que le biais d'horloge admissible, l'erreur KRB_AP_ERR_TKT_EXPIRED est retournée.

Si toutes ces vérifications ont réussi sans erreur, le serveur est assura que le client possède les accréditifs du principal nommé dans le ticket, et donc, que le client a été authentifié auprès du serveur.

Réussir ces vérifications ne fournit que l'authentification du principal désigné ; cela n'implique pas l'autorisation d'utiliser le service désigné. Les applications DOIVENT faire une décision d'autorisation distincte sur la base du nom authentifié de l'utilisateur, de l'opération demandée, des informations de commande d'accès locales telles que celles contenues dans un fichier .k5login ou .k5users, et éventuellement d'un service d'autorisation distribué distinct.

3.2.4 Génération d'un message KRB_AP_REP

Normalement, la demande d'un client va inclure à la fois les informations d'authentification et sa demande initiale dans le même message, et le serveur n'a pas besoin de répondre explicitement au KRB_AP_REQ. Cependant, si l'authentification mutuelle (authentifiant non seulement le client auprès du serveur, mais aussi le serveur auprès du client) est effectuée, le message KRB_AP_REQ aura MUTUAL-REQUIRED établi dans son champ ap-options, et un message KRB_AP_REP est exigé en réponse. Comme avec le message d'erreur, ce message PEUT être encapsulé dans le protocole d'application si sa forme "brute" n'est pas acceptable pour le protocole d'application. Les champs horodatage et microseconde utilisés dans la réponse DOIIVENT être les champs horodatage et microseconde du client (tels que fournis par l'authentifiant). Si un numéro de séquence doit être inclus, il DEVRAIT être choisi de façon aléatoire comme décrit ci-dessus pour l'authentifiant. Une sous-clé PEUT être incluse si le serveur désire négocier une sous-clé différente. Le message KRB_AP_REP est chiffré dans la clé de session extraite du ticket.

Noter que dans le protocole Kerberos Version 4, l'horodatage dans la réponse était l'horodatage du client plus un. Ceci n'est pas nécessaire en Version 5 parce que le message de Version 5 est formaté de telle façon qu'il n'est pas possible de créer la réponse par une opération chirurgicale judicieuse du message (même en forme chiffrée) sans connaissance des clés de chiffrement appropriées.

3.2.5 Réception du message KRB_AP_REP

Si un message KRB_AP_REP est retourné, le client utilise la clé de session provenant des accréditifs obtenus pour que le serveur déchiffre le message et vérifie que les champs horodatage et microseconde correspondent à ceux de l'authentifiant qu'il a envoyé au serveur. Si ils correspondent, le client est alors assuré que le serveur est authentique. Le numéro de séquence et la sous-clé (si elle est présente) sont conservés pour utilisation ultérieure. (Noter que pour chiffrer le message KRB_AP_REP, la sous-clé de session n'est pas utilisée, même si elle est présente dans l'authentification.)

3.2.6 Utilisation de la clé de chiffrement

Après que l'échange KRB_AP_REQ/KRB_AP_REP est terminé, le client et le serveur partagent une clé de chiffrement qui peut être utilisée par l'application. Dans certains cas, l'utilisation de cette clé de session sera implicite dans le protocole ; dans d'autres, la méthode d'utilisation doit être choisie entre plusieurs possibilités. L'application PEUT choisir la clé de chiffrement réelle à utiliser pour KRB_PRIV, KRB_SAFE, ou d'autres utilisations spécifiques de l'application fondées sur la clé de session à partir du ticket et des sous-clés dans le message KRB_AP_REP et l'authentifiant. Les mises en œuvre du protocole PEUVENT fournir des routines pour choisir les sous-clés sur la base des clés de session et de nombres aléatoires et générer une clé négociée à retourner dans le message KRB_AP_REP.

Pour atténuer l'effet des défaillances de la génération de nombres aléatoires chez le client, il est vivement recommandé que toute clé déduite par une application pour utilisation ultérieure incluse la pleine entropie de clé déduite de la clé de session générée par le KDC portée dans le ticket. On laisse les négociations de protocole sur la façon d'utiliser la clé (par exemple, pour choisir un type de chiffrement ou de somme de contrôle) au programmeur de l'application. Le protocole Kerberos n'impose pas de contrainte sur les options de mise en œuvre, mais un exemple de la façon dont cela peut être fait figure ci-après.

Une façon qu'une application peut choisir pour négocier une clé à utiliser pour la protection ultérieure d'intégrité et de confidentialité est que le client propose une clé dans le champ sous-clé à l'authentifiant. Le serveur peut alors choisir une clé en utilisant la clé proposée par le client en entrée, retournant la nouvelle sous-clé dans le champ sous-clé de la réponse de l'application. Cette clé peut alors être utilisée pour la suite de la communication.

Avec les échanges d'authentification aussi bien unilatérale que mutuelle, les homologues devraient veiller à ne pas s'envoyer l'un l'autre d'informations sensibles sans garanties appropriées. En particulier, les applications qui exigent la confidentialité ou l'intégrité DEVRAIENT utiliser la réponse KRB_AP_REP du serveur au client pour que le client et le serveur s'assurent tous deux de l'identité de leur homologue. Si un protocole d'application exige la confidentialité de ses messages, il peut utiliser le message KRB_PRIV (paragraphe 3.5). Le message KRB_SAFE (paragraphe 3.4) peut être utilisé pour s'assurer de l'intégrité.

3.3 L'échange de service d'allocation de ticket (TGS)

Résumé

Direction du message	Type de message	Paragraphe
1. Client à Kerberos	KRB_TGS_REQ	5.4.1
2. Kerberos à client	KRB_TGS_REP ou	5.4.2
	KRB_ERROR	5.9.1

L'échange TGS entre un client et le TGS Kerberos est initié par un client lorsque il cherche à obtenir des accréditifs d'authentification pour un serveur donné (qui peut être enregistré dans un domaine distant), lorsque il cherche à renouveler ou valider un ticket existant, ou lorsque il cherche à obtenir un ticket mandataire. Dans le premier cas, le client doit déjà avoir acquis un ticket pour le service d'allocation de tickets (TGS, *Ticket-Granting Service*) en utilisant l'échange d'AS (le TGT est habituellement obtenu lorsque un client s'authentifie initialement auprès du système, comme lorsque un usager se connecte). Le format de message pour l'échange TGS est presque identique à celui de l'échange d'AS. La principale différence est que le chiffrement et le déchiffrement de l'échange TGS n'a pas lieu avec la clé du client. Au lieu de cela, on utilise la clé de session provenant du TGT ou le ticket renouvelable, ou la sous-clé de session provenant d'un authentifiant. Comme dans le cas de tous les serveurs d'application, les tickets arrivés à expiration ne sont pas acceptés par le TGS, aussi une fois qu'un ticket renouvelable ou TGT arrive à expiration, le client doit utiliser un échange distinct pour obtenir des tickets valides.

L'échange TGS consiste en deux messages : une demande (KRB_TGS_REQ) du client au serveur d'allocation de tickets Kerberos, et une réponse (KRB_TGS_REP ou KRB_ERROR). Le message KRB_TGS_REQ comporte des informations qui authentifient le client plus une demande d'accréditifs. Les informations d'authentification comportent l'en-tête d'authentification (KRB_AP_REQ), qui inclut le ticket d'allocation de ticket précédemment obtenu, ou le ticket renouvelable ou le ticket invalide, du client. Dans le cas de TGT et de mandat, la demande PEUT inclure un ou plusieurs des éléments suivants : une liste d'adresses réseau, une collection de données d'autorisation typées à sceller dans le ticket pour que le serveur d'application les utilise à des autorisations, ou des tickets supplémentaires (dont l'utilisation sera décrite plus loin). La réponse TGS (KRB_TGS_REP) contient les accréditifs demandés, chiffrés dans la clé de session provenant du TGT ou le ticket renouvelable, ou, si elle est présente, dans la sous-clé de session provenant de l'authentifiant (partie de l'en-tête d'authentification). Le message KRB_ERROR contient un code d'erreur et du texte expliquant ce qui ne va pas. Le message KRB_ERROR n'est pas chiffré. Le message KRB_TGS_REP contient des informations qui peuvent être utilisées pour détecter les répétitions, et pour l'associer au message auquel il répond. Le message KRB_ERROR contient aussi des informations qui peuvent être utilisées pour l'associer au message auquel il répond. Les mêmes commentaires sur la protection de l'intégrité des messages KRB_ERROR mentionnés au paragraphe 3.1 s'appliquent à l'échange TGS.

3.3.1 Génération du message KRB_TGS_REQ

Avant d'envoyer une demande au service d'allocation de tickets, le client DOIT déterminer dans quel domaine il pense qu'est enregistré le serveur d'application. Cela peut se faire de diverses façons. Il peut être connu à l'avance (car le domaine fait partie de l'identifiant du principal), il peut être mémorisé dans un serveur de noms, ou il peut être obtenu d'un fichier de configuration. Si le domaine à utiliser est obtenu d'un serveur de noms, il y a le danger d'être mystifié si le service de noms qui fournit le nom de domaine n'est pas authentifié. Il peut en résulter l'utilisation d'un domaine qui a été compromis, d'où il pourrait résulter la capacité pour un agresseur de compromettre l'authentification du serveur d'application auprès du client.

Si le client connaît le nom et le domaine du principal de service et s'il ne possède pas encore un TGT pour le domaine approprié, il doit alors en obtenir un. Il s'y essaye d'abord en demandant un TGT pour le domaine de destination à un serveur Kerberos pour lequel le client possède un TGT (en utilisant de façon récurrente le message KRB_TGS_REQ). Le serveur Kerberos PEUT retourner un TGT pour le domaine désiré, auquel cas le traitement peut se poursuivre. Autrement, le serveur Kerberos PEUT retourner un TGT pour un domaine qui est plus 'proche' du domaine désiré (plus loin sur le chemin hiérarchique standard entre le domaine du client et domaine du serveur de domaine demandé). Noter que dans ces cas, la mauvaise configuration du serveur Kerberos peut causer des boucles dans le chemin d'authentification résultant, ce que le client devrait veiller à détecter et éviter.

Si le serveur Kerberos retourne un TGT pour un domaine 'plus proche' que le domaine désiré, le client PEUT utiliser la configuration de la politique locale pour vérifier que le chemin d'authentification utilisé est acceptable. Autrement, un client PEUT choisir son propre chemin d'authentification, plutôt que de s'appuyer sur le serveur Kerberos pour en choisir un. Dans l'un et l'autre cas, toute information de politique ou de configuration utilisée pour choisir ou valider des chemins d'authentification, par le serveur Kerberos ou par le client, DOIT être obtenue d'une source de confiance.

Lorsqu'un client obtient un TGT qui est 'plus proche' du domaine de destination, il PEUT mettre en mémoire cache ce ticket et le réutiliser dans des échanges KRB-TGS futurs avec les services dans le domaine 'plus proche'. Cependant, si le client devait obtenir un TGT pour le domaine 'plus proche' en commençant au KDC initial plutôt qu'au titre de l'obtention d'un autre ticket, un chemin plus court vers le domaine 'plus proche' pourrait être utilisé. Ce chemin plus court peut être désirable parce que moins de KDC intermédiaires connaîtraient la clé de session du ticket impliqué. Pour cette raison, les clients DEVRAIENT évaluer si ils ont confiance dans les domaines de transit pour obtenir le ticket 'plus proche' lorsqu'ils prennent la décision d'utiliser le ticket à l'avenir.

Une fois que le client a obtenu un TGT pour le domaine approprié, il détermine quels serveurs Kerberos servent ce domaine et contacte l'un d'entre eux. Leur liste peut être obtenue par un fichier de configuration ou un service réseau, ou elle PEUT être générée à partir du nom du domaine. Tant que les clés secrètes échangées par domaine sont gardées secrètes, seul le déni de service résulte de l'utilisation d'un faux serveur Kerberos.

Comme dans l'échange d'AS, le client PEUT spécifier un certain nombre d'options dans le message KRB_TGS_REQ. Une de ces options est l'option ENC-TKT-IN-SKEY utilisée pour l'authentification d'usager à usager. On trouvera une vue d'ensemble de l'authentification d'usager à usager au paragraphe 3.7. Lors de la génération du message KRB_TGS_REQ, cette option indique que le client inclut un TGT obtenu du serveur d'application dans le champ de tickets supplémentaires de la demande et que le KDC DEVRAIT chiffrer le ticket pour le serveur d'application en utilisant la clé de session provenant de ce ticket supplémentaire, au lieu d'une clé de serveur provenant de la base de données du principal.

Le client prépare le message KRB_TGS_REQ, qui fournit un en-tête d'authentification comme élément du champ padata, et incluant les mêmes champs qu'utilisés dans le message KRB_AS_REQ avec plusieurs champs facultatifs : le champ enc-authorizatfion-data pour l'usage du serveur d'application et les tickets supplémentaires requis par certaines options.

Pour préparer l'en-tête d'authentification, le client peut choisir une sous-clé de session selon laquelle sera chiffrée la réponse du serveur Kerberos. Si le client choisit une sous-clé de session, il faut veiller à s'assurer du caractère aléatoire de la sous-clé de session choisie.

Si la sous-clé de session n'est pas spécifiée, la clé de session provenant du TGT sera utilisée. Si enc-authorization-data est présent, il DOIT être chiffré dans la sous-clé de session, si elle est présente, à partir de la portion d'authentifiant de l'en-tête d'authentification, ou, si elle n'est pas présente, en utilisant la clé de session provenant du TGT.

Une fois préparé, le message est envoyé au serveur Kerberos pour le domaine de destination.

3.3.2 Réception du message KRB_TGS_REQ

Le message KRB_TGS_REQ est traité de manière similaire au message KRB_AS_REQ, mais il y a de nombreuses vérifications supplémentaires à effectuer. D'abord, le serveur Kerberos DOIT déterminer pour quel serveur est le ticket d'accompagnement, et il DOIT choisir la clé appropriée pour le déchiffrer. Pour un message KRB_TGS_REQ normal, ce sera pour le service d'allocation de tickets, et la clé du TGS sera utilisée. Si le TGT a été produit par un autre domaine, les clés inter domaines appropriées DOIVENT être utilisées. Si (a) le ticket d'accompagnement n'est pas un TGT pour le domaine courant, mais est pour un serveur d'application dans le domaine courant, (b) les options

RENEW, VALIDATE, ou PROXY sont spécifiées dans la demande, et (c) le serveur pour lequel un ticket est demandé est le serveur désigné dans le ticket d'accompagnement, puis le KDC va déchiffrer le ticket dans l'en-tête d'authentification en utilisant la clé du serveur pour lequel elle a été produite. Si aucun ticket ne peut être trouvé dans le champ padata, l'erreur KDC_ERR_PADATA_TYPE_NOSUPP est retournée.

Une fois que le ticket d'accompagnement a été déchiffré, la somme de contrôle fournie par l'usager dans l'authentifiant DOIT être vérifiée par rapport au contenu de la demande, et le message DOIT être rejeté si les sommes de contrôle ne correspondent pas (avec un code d'erreur de KRB_AP_ERR_MODIFIED) ou si la somme de contrôle n'est pas à l'épreuve des collisions (avec un code d'erreur de KRB_AP_ERR_INAPP_CKSUM). Si le type de somme de contrôle n'est pas accepté, l'erreur KDC_ERR_SUMTYPE_NOSUPP est retournée. Si les données d'autorisation sont présentes, elles sont déchiffrées en utilisant la sous-clé de session provenant de l'authentifiant.

Si un des déchiffrements indique un échec de vérification d'intégrité, l'erreur KRB_AP_ERR_BAD_INTEGRITY est retournée.

Comme exposé au paragraphe 3.1.2, le KDC DOIT envoyer un message KRB_TGS_REP valide si il reçoit un message KRB_TGS_REQ identique à celui qu'il a traité récemment. Cependant, si l'authentifiant est une répétition, mais que le reste de la demande n'est pas identique, le KDC DEVRAIT alors retourner KRB_AP_ERR_REPEAT.

3.3.3 Génération du message KRB_TGS_REP

Le message KRB_TGS_REP partage son format avec KRB_AS_REP (KRB_KDC_REP), mais avec son champ de type réglé à KRB_TGS_REP. Sa spécification détaillée est au paragraphe 5.4.2.

La réponse va comporter un ticket pour le serveur demandé ou pour un serveur d'allocation de ticket d'un KDC intermédiaire à contacter pour obtenir le ticket demandé. La base de données Kerberos est interrogée pour restituer l'enregistrement pour le serveur approprié (y compris la clé avec laquelle le ticket sera chiffré). Si la demande est pour un TGT d'un domaine distant, et si il n'y a pas de partage de clé avec le domaine demandé, le serveur Kerberos va alors choisir le domaine 'le plus proche' du domaine demandé avec lequel il partage une clé et utiliser ce domaine à la place. C'est le seul cas où la réponse pour le KDC sera pour un serveur différent de celui demandé par le client.

Par défaut, le champ d'adresse, le nom et le domaine du client, la liste des domaines de transit, l'heure de l'authentification initiale, l'heure d'expiration, et les données d'autorisation du ticket nouvellement produit seront copiées du TGT ou du ticket renouvelable. Si les champs de transit ont besoin d'être mis à jour, mais que le type de transit n'est pas accepté, l'erreur KDC_ERR_TRTYPE_NOSUPP est retournée.

Si la demande spécifie une heure de fin, l'heure de fin du nouveau ticket est réglée au minimum de (a) cette demande, (b) de l'heure de fin provenant du TGT, et (c) de l'heur de début du TGT plus le minimum de la durée de vie maximum pour le serveur d'application et la durée de vie maximum pour le domaine local (la durée de vie maximum pour le principal demandeur a déjà été appliquée lors de la production du ticket). Si le nouveau ticket doit être un renouvellement, l'heure de fin ci-dessus est alors remplacée par le minimum de (a) la valeur du champ renew_till du ticket et (b) l'heure de début pour le nouveau ticket plus la durée de vie (heure de fin – heure de début) du vieux ticket.

Si l'option FORWARDED a été demandée, le ticket résultant contiendra alors les adresses spécifiées par le client. Cette option ne sera honorée que si le fanion FORWARDABLE est établi dans le TGT. L'option PROXY est similaire ; le ticket résultant contiendra les adresses spécifiées par le client. Elle ne sera honorée que si le fanion PROXIABLE est établi dans le TGT. L'option PROXY ne sera pas honorée sur les demandes de TGT supplémentaires.

Si l'heure de début demandée est absente, indique une heure passée, ou est dans la fenêtre de biais d'horloge admissible pour le KDC et si l'option POSTDATE n'a pas été spécifiée, l'heure de début du ticket est réglée à l'heure en cours du serveur d'authentification. Si elle indique une heure dans le futur au delà du biais d'horloge admissible, mais si l'option POSTDATED n'a pas été spécifiée ou si le fanion MAY-POSTDATE n'est pas établi dans le TGT, l'erreur KDC_ERR_CANNOT_POSTDATE est alors retournée. Autrement, si le TGT a le fanion MAY-POSTDATE établi, le ticket résultant sera postdaté, et l'heure de début demandée sera vérifiée par rapport à la politique du domaine local. Si elle est acceptable, l'heure de début du ticket sera réglée comme demandé, et le fanion INVALID sera mis. Le ticket postdaté DOIT être validé avant utilisation en le présentant au KDC après l'heure de début a été atteinte. Cependant, en aucun cas l'heure de début, l'heure de fin, ou l'heure de fin de renouvellement-jusqu'à d'un ticket postdaté nouvellement produit ne peut être étendue au delà de l'heure de renouvellement-jusqu'à du TGT.

Si l'option ENC-TKT-IN-SKEY a été spécifiée et si un ticket supplémentaire a été inclus dans la demande, il indique

que le client utilise l'authentification d'usager à usager pour prouver son identité à un serveur qui n'a pas d'accès à une clé persistante. Le paragraphe 3.7 décrit les effets de cette option sur le protocole Kerberos tout entier. Lors de la génération du message KRB_TGS_REP, cette option dans le message KRB_TGS_REQ dit au KDC de déchiffrer le ticket supplémentaire en utilisant la clé pour le serveur auquel le ticket supplémentaire a été produit et de vérifier qu'il est un TGT. Si le nom du serveur demandé manque dans la demande, le nom du client dans le ticket supplémentaire sera utilisé. Autrement, le nom du serveur demandé sera comparé au nom du client dans le ticket supplémentaire. Si c'est différent, la demande sera rejetée. Si la demande réussit, la clé de session provenant du ticket supplémentaire sera utilisée pour chiffrer le nouveau ticket produit au lieu d'utiliser la clé du serveur pour lequel le nouveau ticket sera utilisé.

Si (a) le nom du serveur dans le ticket qui est présenté au KDC au titre de l'en-tête d'authentification n'est pas celui du TGS lui-même, (b) le serveur est enregistré dans le domaine du KDC, et (c) l'option RENEW est demandée, le KDC va alors vérifier que le fanion RENEWABLE est établi dans le ticket, que le fanion INVALID n'est pas mis dans le ticket, et que l'heure renouveler_jusqu'à est toujours à venir. Si l'option VALIDATE est demandée, le KDC vérifiera que l'heure de début est dépassée et que le fanion INVALID est mis. Si l'option PROXY est demandée, le KDC vérifiera alors que le fanion PROXIABLE est mis dans le ticket. Si les essais sont réussis et que le ticket satisfait aux vérifications de liste rouge décrites dans le paragraphe qui suit, le KDC va produire le nouveau ticket approprié.

La partie chiffrée de la réponse dans le message KRB_TGS_REP est chiffrée dans la sous-clé de session provenant de l'authentifiant, s'il est présent, ou dans la clé de session provenant du TGT. Elle n'est pas chiffrée en utilisant la clé secrète du client. De plus, les champs de date d'expiration de la clé du client et de numéro de version sont laissés de côté car ces valeurs sont mémorisées avec les enregistrements de la base de données du client, et il n'est pas besoin de cet enregistrement pour satisfaire une demande sur la base d'un TGT.

3.3.3.1 Recherche de tickets révoqués

Chaque fois qu'une demande est faite au serveur d'allocation de tickets, le ou les tickets présentés sont confrontés à une liste rouge de tickets annulés. Cette liste rouge peut être mise en œuvre en mémorisant une gamme d'horodatages de production de 'tickets suspects' ; si un ticket présenté a un authtime dans cette gamme, il sera rejeté. De cette façon, un TGT volé ou un ticket renouvelable ne peut pas être utilisé pour obtenir des tickets supplémentaires (renouvellements ou autres) une fois que le vol a été rapporté au KDC pour le domaine dans lequel réside le serveur. Tout ticket normal obtenu avant qu'il ait été rapporté volé sera toujours valide (parce que les tickets n'exigent pas d'interaction avec le KDC), mais seulement jusqu'à son heure d'expiration normale. Si les TGT ont été produites pour une authentification inter-domaine, l'utilisation du TGT inter domaine ne sera pas affectée sauf si la liste rouge est transmise aux KDC pour les domaines pour lesquels de tels tickets inter-domaine avaient été produits.

3.3.3.2 Codage du champ de transit

Si l'identité du serveur dans le TGT qui est présenté au KDC au titre de l'en-tête d'authentification est celle du service d'allocation de tickets, mais si le TGT a été produit à partir d'un autre domaine, le KDC va chercher la clé inter domaines partagée avec ce domaine et utiliser cette clé pour déchiffrer le ticket. Si le ticket est valide, le KDC va alors honorer la demande, sous réserve des contraintes soulignées ci-dessus au paragraphe qui décrit l'échange d'AS. La partie domaine de l'identité du client sera tirée du TGT. Le nom de domaine qu'a produit le TGT, si ce n'est pas le domaine du principal client, sera ajouté au champ de transit du ticket à produire. Ceci est accomplis en lisant le champ de transite d'après le TGT (qui est traité comme un ensemble non ordonné de noms de domaines), en ajoutant le nouveau domaine à l'ensemble, et en construisant et écrivant sa forme (abrégée) codée (ce qui peut impliquer un réarrangement du codage existant).

Noter que le service d'allocation de tickets n'ajoute pas le nom de son propre domaine. Au lieu de cela, sa responsabilité est d'ajouter le nom du domaine précédent. Cela empêche un serveur Kerberos malveillant d'inscrire intentionnellement son propre nom (il pourrait, cependant, omettre les noms d'autres domaines).

Ni les noms du domaine local ni ceux du domaine du principal ne sont à inclure dans le champ de transit. Ils apparaissent ailleurs dans le ticket et tous deux sont connus pour prendre part à l'authentification du principal. Parce que les points d'extrémité ne sont pas inclus, aussi bien l'authentification inter domaine local que celle à un seul bond résulte en un champ de transit qui est vide.

Comme à ce champ est ajouté le nom de chaque domaine de transit, il peut éventuellement être très long. Pour diminuer la longueur de ce champ, son contenu est codé. Les codages initialement acceptés sont optimisés sur le cas normal de communication inter domaine : un arrangement hiérarchique de domaines utilisant les noms de domaine de style domaine ou X.500. Ce codage (appelé DOMAIN-X500-COMPRESS) va être décrit ci-dessous.

Les noms de domaine dans le champ de transit sont séparés par une ",". Les caractères ",", "\", les "." en queue, et les espaces (" ") en tête sont des caractères spéciaux, et si ils font partie d'un nom de domaine, ils DOIVENT être entre guillemets dans le champ de transit en les faisant précéder d'une "\".

Un nom de domaine se terminant par un "." est interprété comme étant ajouté au domaine précédent. Par exemple, on peut coder la traversée de EDU, MIT.EDU, ATHENA.MIT.EDU, WASHINGTON.EDU, et CS.WASHINGTON.EDU par :

"EDU,MIT.,ATHENA.,WASHINGTON.EDU,CS.".

Noter que si ATHENA.MIT.EDU, ou CS.WASHINGTON.EDU étaient des points d'extrémité, ils ne seraient pas inclus dans ce champ, et on aurait :

"EDU,MIT.,WASHINGTON.EDU"

Un nom de domaine commençant par une "/" est interprété comme ayant été ajouté au domaine précédent. Pour les besoins de l'ajout, le domaine précédant le premier domaine de la liste est considéré comme le domaine nul (""). Si un nom de domaine commençant par une "/" doit être autonome, il DEVRAIT alors être précédé d'un espace (" "). Par exemple, on peut coder la traversée de /COM/HP/APOLLO, /COM/HP, /COM, et /COM/DEC par :

"/COM,/HP,/APOLLO, /COM/DEC".

Comme dans l'exemple ci-dessus, si /COM/HP/APOLLO et /COM/DEC étaient des points d'extrémité, ils ne seraient pas inclus dans ce champ, et on aurait :

"/COM,/HP"

Un sous-champ nul précédant ou suivant une "," indique que tous les domaines entre le domaine précédent et le prochain domaine ont été traversés. Pour les besoins de l'interprétation des sous-champs nuls, le domaine du client est considéré comme précédant ceux du champ de transit, et le domaine du serveur est considéré les suivre. Et donc, "," signifie que tous les domaines le long du chemin entre le client et le serveur ont été traversés. ",EDU, /COM," signifie que tous les domaines depuis le domaine du client jusqu'à EDU (dans une hiérarchie de style domaine) ont été traversés, et que tous depuis /COM jusqu'au domaine du serveur dans un style X.500 ont aussi été traversés. Cela pourrait survenir si le domaine EDU dans une hiérarchie partage une clé inter domaine directement avec le domaine /COM dans une autre hiérarchie.

3.3.4 Réception du message KRB_TGS_REP

Lorsque le message KRB_TGS_REP est reçu par le client, il est traité de la même manière que le traitement KRB_AS_REP décrit ci-dessus. La principale différence est que la partie chiffrée de la réponse doit être déchiffrée en utilisant la sous-clé de session provenant de l'authentifiant, si il a été spécifié dans la demande, ou la clé de session provenant du TGT, plutôt que la clé secrète du client. Le nom de serveur retourné dans la réponse est le vrai nom de principal du service.

3.4 L'échange KRB_SAFE

Le message KRB_SAFE PEUT être utilisé par des clients qui exigent la capacité à détecter les modifications des messages qu'ils échangent. Ceci est réalisé en incluant la frappe d'une somme de contrôle à l'épreuve des collisions des données d'usager et de quelques informations de contrôle. La somme de contrôle est frappée avec une clé de chiffrement (normalement la dernière clé négociée via des sous-clés, ou la clé de session si aucune négociation n'est survenue.

3.4.1 Génération d'un message KRB_SAFE

Lorsqu'une application souhaite envoyer un message KRB_SAFE, elle collecte ses données et les informations de contrôle appropriées et calcule une somme de contrôle sur l'ensemble. L'algorithme de somme de contrôle devrait être la somme de contrôle frappée mandatée pour être mise en œuvre ainsi que le système de cryptage utilisé pour la clé de session ou de sous-session. La somme de contrôle est générée en utilisant la sous-clé de session, si elle est présente, ou la clé de session. Certaines mises en œuvre utilisent un algorithme de somme de contrôle différent pour les messages

KRB_SAFE, mais il n'est pas toujours possible de faire ainsi de façon interopérable.

Les informations de contrôle pour le message KRB_SAFE comportent à la fois un horodatage et un numéro de séquence. Le concepteur d'une application utilisant le message KRB_SAFE DOIT choisir au moins un des deux mécanismes. Ce choix DEVRAIT se fonder sur les besoins du protocole d'application.

Les numéros de séquence sont utiles lorsque tous les messages envoyés seront reçus par l'homologue. L'état de connexion est actuellement exigé pour l'entretien de la clé de session, aussi la maintenance du prochain numéro de séquence ne devrait pas présenter de problème supplémentaire.

Si le protocole d'application est prévu pour tolérer des pertes de messages sans qu'ils soient renvoyés, l'utilisation de l'horodatage est le mécanisme de détection de répétition approprié. L'utilisation des horodatages est aussi le mécanisme approprié pour les protocoles de diffusion groupée dans lesquels tous les homologues partagent une sous-clé de session commune, mais certains messages seront envoyés à un sous-ensemble d'un homologue.

Après le calcul de la somme de contrôle, le client transmet les informations et la somme de contrôle au receveur dans le format de message spécifié au paragraphe 5.6.1.

3.4.2 Réception du message KRB_SAFE

Lorsque une application reçoit un message KRB_SAFE, elle le vérifie comme suit. Si une erreur survient, un code d'erreur est rapporté pour être utilisé par l'application.

On vérifie d'abord que les champs version et type du protocole du message correspondent respectivement à la version en cours et à KRB_SAFE. Une discordance génère une erreur KRB_AP_ERR_BADVERSION ou KRB_AP_ERR_MSG_TYPE. L'application vérifie que la somme de contrôle utilisée est une somme de contrôle frappée à l'épreuve des collisions qui utilise des clés compatibles avec la clé de session ou clé de sous-session selon le cas approprié (ou la clé d'application déduite des clés de session ou de sous-session). Si ce n'est pas le cas, une erreur KRB_AP_ERR_INAPP_CKSUM est générée. L'adresse de l'envoyeur DOIT être incluse dans les informations de contrôle ; le receveur vérifie que le rapport du système d'exploitation de l'adresse de l'envoyeur correspond à l'adresse de l'envoyeur dans le message, et (si une adresse de réception est spécifiée ou si le receveur exige une adresse) qu'une des adresses du receveur apparaît comme adresse de réception dans le message. Pour travailler avec la traduction d'adresse réseau, les envoyeurs PEUVENT utiliser le type d'adresse directionnel spécifié au paragraphe 8.1 pour l'adresse d'envoyeur et ne pas inclure d'adresse de receveur. Un échec de correspondance pour l'un de ces cas génère une erreur KRB_AP_ERR_BADADDR. Puis les champs horodatage et usec et/ou de numéro de séquence sont vérifiés. Si horodatage et usec sont attendus et absents, ou si ils sont présents mais ne sont pas ceux en cours, l'erreur KRB_AP_ERR_SKEW est générée. Il n'est pas exigé que les horodatages soient strictement ordonnées ; il est seulement exigé qu'ils soient dans la fenêtre de biais d'horloge. Si les champs nom du serveur, ainsi que nom du client, heure, et microsecondes de l'authentifiant correspondent à de tels tuples récemment vus (envoyés ou reçus), l'erreur KRB_AP_ERR_REPEAT est générée. Si un numéro de séquence incorrect est inclus, ou si un numéro de séquence est attendu mais n'est pas présent, l'erreur KRB_AP_ERR_BADORDER est générée. Si ni un horodatage ni usec ni un numéro de séquence ne sont présents, une erreur KRB_AP_ERR_MODIFIED est générée. Finalement, la somme de contrôle est calculée sur les informations de données et de contrôle, et si elle ne correspond pas à la somme de contrôle reçue, une erreur KRB_AP_ERR_MODIFIED est générée.

Si toutes les vérifications réussissent, l'application est sure que le message a été généré par son homologue et n'a pas été modifié dans le transit.

Les mises en œuvre DEVRAIENT accepter tout algorithme de somme de contrôle qui a à la fois la sécurité adéquate et des clés compatibles avec la clé de session ou de sous-session. Les sommes de contrôle non frappées ou qui ne sont pas à l'épreuve des collisions ne conviennent pas pour cette utilisation.

3.5 L'échange KRB_PRIV

Le message KRB_PRIV PEUT être utilisé par les clients qui exigent la confidentialité et la capacité à détecter les modifications des messages échangés. Il réalise cela en chiffrant les messages et en ajoutant des informations de contrôle.

3.5.1 Génération d'un message KRB_PRIV

Lorsque une application souhaite envoyer un message KRB_PRIV, elle collecte ses données et les informations de contrôle appropriées (spécifiée ai paragraphe 5.7.1) et les chiffre avec une clé de chiffrement (habituellement la dernière clé négociée via des sous-clés, ou la clé de session si aucune négociation n'est survenue). Au titre des informations de contrôle, le client DOIT choisir d'utiliser soit un horodatage soit un numéro de séquence (ou les deux) ; voir la discussion au paragraphe 3.4.1 pour des lignes directrice sur ce qu'il faut utiliser. Après le chiffrement des données d'utilisateur et des informations de contrôle, le client transmet le texte chiffré et certaines des informations 'd'enveloppe' au receveur.

3.5.2 Réception du message KRB_PRIV

Lorsque une application reçoit un message KRB_PRIV, elle le vérifie comme suit. Si une erreur survient, un code d'erreur est rapporté pour être utilisé par l'application.

Il est d'abord vérifié que les champs de version et type du protocole du message correspondent respectivement à la version en cours et au KRB_PRIV. Une discordance génère une erreur KRB_AP_ERR_BADVERSION ou KRB_AP_ERR_MSG_TYPE. L'application déchiffre ensuite le texte chiffré et traite le texte clair qui en résulte. Si le déchiffrement montre que les données ont été modifiées, une erreur KRB_AP_ERR_BAD_INTEGRITY est générée.

L'adresse de l'envoyeur DOIT être incluse dans les informations de contrôle ; le receveur vérifie que le rapport du système d'exploitation de l'adresse de l'envoyeur correspond à l'adresse de l'envoyeur dans le message. Si une adresse de receveur est spécifiée ou si le receveur exige une adresse, une des adresses du receveur DOIT aussi apparaître comme adresse de receveur dans le message. Lorsqu'une adresse d'envoyeur ou de receveur pourrait ne pas correspondre par ailleurs à l'adresse du message à cause d'une traduction d'adresse réseau, une application PEUT être écrite pour utiliser les adresses du type d'adresse directionnelle à la place de l'adresse réseau réelle.

L'échec de correspondance d'un des cas génère une erreur KRB_AP_ERR_BADADDR. Pour travailler avec la traduction d'adresse du réseau, les mises en œuvre PEUVENT utiliser le type d'adresse directionnelle défini au paragraphe 7.1 pour l'adresse de l'envoyeur et ne pas inclure d'adresse de receveur.

Après cela intervient la vérification des champs horodatage et usec et/ou de numéro de séquence. Si l'horodatage et usec sont attendus et ne sont pas présents, ou si ils sont présents mais ne sont pas ceux en cours, l'erreur KRB_AP_ERR_SKEW est générée. Si les champs de nom du serveur, ainsi que nom du client, heure, et microseconde provenant de l'authentifiant correspondent à tout tuple de ces valeurs vu récemment, l'erreur KRB_AP_ERR_REPEAT est générée. Si un numéro de séquence incorrect est inclus, ou si un numéro de séquence est attendu mais n'est pas présent, l'erreur KRB_AP_ERR_BADORDER est générée. Si ni un horodatage, ni usec ni un numéro de séquence n'est présent, une erreur KRB_AP_ERR_MODIFIED est générée.

Si toutes les vérifications réussissent, l'application peut supposer que le message a été généré par son homologue et a été transmis en toute sécurité (sans que des intrus aient vu le contenu non chiffré).

3.6 L'échange KRB_CRED

Le message KRB_CRED PEUT être utilisé par les clients qui exigent la capacité à envoyer des accréditifs Kerberos d'un hôte à l'autre. Ceci est réalisé par l'envoi des tickets avec les données chiffrées qui contiennent les clés de session et les autres informations associées à ces tickets.

3.6.1 Génération d'un message KRB_CRED

Lorsque une application souhaite envoyer un message KRB_CRED, elle obtient d'abord (en utilisant l'échange KRB_TGS) des accréditifs à envoyer à l'hôte distant. Elle construit ensuite un message KRB_CRED en utilisant le ou les tickets ainsi obtenus, plaçant la clé de session qu'il est nécessaire d'utiliser pour chaque ticket dans le champ clé de la séquence KrbCredInfo correspondante dans la partie chiffrée du message KRB_CRED.

Les autres informations associées à chaque ticket et obtenues durant l'échange KRB_TGS sont aussi placées dans la séquence KrbCredInfo correspondante dans la partie chiffrée du message KRB_CRED. L'heure en cours et, si ils sont spécifiquement exigés par l'application, les champs nonce, s-address, et r-address sont placés dans la partie chiffrée du message KRB_CRED, qui est alors chiffré avec la clé de chiffrement précédemment échangée dans le KRB_AP (habituellement la dernière clé négociée via des sous-clés, ou la clé de session si aucune négociation n'est intervenue).

Note de mise en œuvre : Lors de la construction d'un message KRB_CRED pour inclusion dans un jeton de contexte initial GSSAPI, la mise en œuvre MIT de Kerberos ne chiffrera pas le message KRB_CRED si la clé de session est DES ou triple DES. Pour l'interopérabilité avec MIT, la mise en œuvre Microsoft ne chiffrera pas le KRB_CRED dans un jeton GSSAPI si elle utilise une clé de session DES. Débutant à la version 1.2.5, MIT Kerberos peut recevoir et décoder des jetons KRB_CRED chiffrés ou non chiffrés dans l'échange GSSAPI. La mise en œuvre Heimdal de Kerberos peut aussi accepter des messages KRB_CRED chiffrés ou non chiffrés. Comme le message KRB_CRED dans un jeton GSSAPI est chiffré dans l'authentifiant, le comportement du MIT ne présente pas de problème de sécurité, bien qu'il soit en violation de la spécification Kerberos.

3.6.2 Réception d'un message KRB_CRED

Lorsque une application reçoit un message KRB_CRED, elle le vérifie. Si une erreur survient, un code d'erreur est rapporté pour être utilisé par l'application. Le message est vérifié en s'assurant que les champs de version et type de protocole correspondent respectivement à la version en cours et à KRB_CRED. Une discordance génère une erreur KRB_AP_ERR_BADVERSION ou KRB_AP_ERR_MSG_TYPE. L'application déchiffre alors le texte chiffré et traite le texte clair résultant. Si le déchiffrement montre que les données ont été modifiées, une erreur KRB_AP_ERR_BAD_INTEGRITY est générée.

Si elle est présente ou exigé, le receveur PEUT vérifier que le rapport du système d'exploitation sur l'adresse de l'envoyeur correspond à l'adresse d'envoyeur du message, et qu'une des adresses de réception apparaît comme adresse de réception dans le message. La vérification d'adresse ne fournit aucune sécurité supplémentaire, car l'adresse, si elle est présente, a déjà été vérifiée dans le message KRB_AP_REQ et il n'y a aucun intérêt pour un agresseur à retourner un message KRB_CRED à celui qui l'a généré. Et donc, le receveur PEUT ignorer l'adresse même si elle est présente afin de mieux travailler dans des environnements de traducteur d'adresse réseau (NAT, *Network Address Translation*). L'échec de correspondance pour l'un de ces cas génère une erreur KRB_AP_ERR_BADADDR. Les receveurs PEUVENT sauter la vérification d'adresse, car le message KRB_CRED ne peut généralement pas être reflété à celui qui l'a généré. Les champs horodatage et usec (et le champ nonce, s'il est exigé) sont vérifiés ensuite. Si horodatage et usec ne sont pas présents, ou si ils sont présents mais ne sont pas celui en cours, l'erreur KRB_AP_ERR_SKEW est générée.

Si toutes les vérifications ont réussi, l'application mémorise chacun des nouveaux tickets dans sa mémoire cache d'accréditifs avec la clé de session et les autres informations dans la séquence KrbCredInfo correspondante de la partie chiffrée du message KRB_CRED.

3.7 Échanges d'authentification d'usager à usager

L'authentification d'usager à usager fournit une méthode pour effectuer l'authentification lorsque le vérificateur n'a pas accès à un service de clé à long terme. Ce peut être le cas lorsqu'on utilise un serveur (par exemple, un serveur fenêtre) comme un utilisateur sur une station de travail. Dans un tel cas, le serveur peut avoir accès au TGT obtenu lorsque l'utilisateur se connecte avec la station de travail, mais comme le serveur fonctionne comme un utilisateur non privilégié, il pourrait n'avoir pas accès aux clés de système. Des situations similaires peuvent survenir en faisant fonctionner des applications d'homologue à homologue.

Résumé

Direction du message	**Type de message**	**Paragraphe**
0. Message du serveur d'application	Non spécifié	
1. Client à Kerberos	KRB_TGS_REQ	3.3 & 5.4.1
2. Kerberos à client	KRB_TGS_REP ou KRB_ERROR	3.3 & 5.4.2 5.9.1
3. Client à serveur d'application	KRB_AP_REQ	3.2 & 5.5.1

Pour traiter ce problème, le protocole Kerberos permet au client de demander que le ticket produit par le KDC soit chiffré en utilisant une clé de session provenant d'un TGT produit par la partie qui va vérifier l'authentification. Ce TGT doit être obtenu du vérificateur au moyen d'un échange extérieur au protocole Kerberos, habituellement au titre du protocole d'application. Ce message est montré dans le résumé ci-dessus comme message 0. Noter que parce que le TGT est chiffré avec la clé secrète du KDC, il ne peut pas être utilisé pour l'authentification sans la possession de la clé

secrète correspondante. De plus, parce que le vérificateur ne révèle pas la clé secrète correspondante, fournir le TGT du vérificateur ne permet pas de se faire passer pour le vérificateur.

Le message 0 du tableau ci-dessus représente une négociation spécifique de l'application entre le client et le serveur, à la fin de laquelle tous deux ont déterminé qu'ils vont utiliser l'authentification d'usager à usager, et le client a obtenu le TGT du serveur.

Ensuite, le client inclut le TGT du serveur comme ticket supplémentaire dans sa demande KRB_TGS_REQ au KDC (message 1 dans le tableau ci-dessus) et spécifie l'option ENC-TKT-IN-SKEY dans sa demande.

S'il est validé conformément aux instructions du paragraphe 3.3.3, le ticket d'application retourné au client (message 2 dans le tableau ci-dessus) va être chiffré en utilisant la clé de session provenant du ticket supplémentaire et le client va le noter lorsqu'il utilise ou mémorise le ticket d'application.

Lorsqu'il contacte le serveur en utilisant un ticket obtenu pour l'authentification d'usager à usager (message 3 dans le tableau ci-dessus), le client DOIT spécifier le fanion USE-SESSION-KEY dans le champ ap-options. Cela dit au serveur d'application d'utiliser la clé de session associée à son TGT pour déchiffrer le ticket de serveur fourni dans la demande d'applications.

4 Spécifications de chiffrement et de somme de contrôle

Les protocoles Kerberos décrits dans le présent document sont conçus pour le chiffrement de messages de tailles arbitraires, en utilisant le chiffrement de flux ou de blocs. Le chiffrement est utilisé pour prouver les identités des entités du réseau qui participent à l'échange de messages. Le centre de distribution de clés (KDC, *Key Distribution Center*) pour chaque domaine est de confiance pour tous les principaux enregistrés dans ce domaine pour mémoriser une clé secrète. La preuve de la connaissance de cette clé secrète est utilisée pour vérifier l'authenticité d'un principal.

Le KDC utilise la clé secrète du principal (dans l'échange d'AS) ou une clé de session partagée (dans l'échange de TGS) pour chiffrer les réponses aux demandes de ticket ; la capacité à obtenir la clé secrète ou la clé de session implique la connaissance des clés appropriées et l'identité du KDC. La capacité d'un principal à déchiffrer la réponse du KDC et à présenter un ticket et un authentifiant correctement formé (généré avec la clé de session provenant de la réponse du KDC) à un service vérifie l'identité du principal ; de même, la capacité du service à extraire la clé de session du ticket et à prouver ainsi sa connaissance du secret dans une réponse vérifie l'identité du service.

La [RFC3961] définit un cadre de travail pour définir les mécanismes de chiffrement et de somme de contrôle à utiliser avec Kerberos. Elle définit aussi plusieurs mécanismes, auxquels il pourra être ajouté dans des mises à jour futures du présent document.

L'opération chaîne à clé fournie par la [RFC3961] est utilisée pour produire une clé à long terme pour un principal (généralement pour un utilisateur). La chaîne salée par défaut, s'il en est fourni une via les données de pré-authentification, est l'enchaînement des composants de domaine et de nom du principal, dans l'ordre, et sans séparateur. Sauf indication contraire, on utilise l'ensemble de paramètres opaque de chaîne à clé par défaut comme il est défini dans la [RFC3961].

Les données chiffrées, et les sommes de contrôle sont transmises en utilisant les objets de données EncryptedData, EncryptionKey, et Checksum définis au paragraphe 5.2.9. Les opérations de chiffrement, déchiffrement, et de somme de contrôle décrites dans le présent document utilisent les opérations correspondantes de chiffrement, déchiffrement, et get_mic décrites dans la [RFC3961], avec la génération implicite de "clé spécifique" utilisant les valeurs de "usage de clé" spécifiées dans la description de chaque objet EncryptedData ou Checksum pour faire varier la clé pour chaque opération. Noter que dans certains cas, la valeur à utiliser dépend de la méthode de choix de la clé ou du contexte du message.

Les usages de clé sont des entiers non signés de 32 bits ; zéro n'est pas permis. Les valeurs d'usage de clé pour le chiffrement ou la somme de contrôle des messages Kerberos sont indiquées à la Section 5 avec les définitions de message. Les valeurs d'usage de clé 512 à 1023 sont réservées pour des utilisations internes à une mise en œuvre de Kerberos. (Par exemple, alimenter un générateur de nombres pseudo aléatoires avec une valeur produite en chiffrant quelque chose avec une clé de session et une valeur d'usage de clé non utilisée pour un autre objet.) Les valeurs d'usage de clés entre 1024 et 2047 (inclus) sont réservées à l'usage des applications ; les applications DEVRAIENT utiliser des valeurs paires pour le chiffrement et des valeurs impaires pour les sommes de contrôle dans cette gamme.

Les valeurs d'usage de clés sont aussi résumées dans un tableau au paragraphe 7.5.1.

Il peut exister d'autres documents qui définissent les protocoles dans les termes des types de chiffrement ou des types de sommes de contrôle de la RFC 1510. Ces documents ne connaissent rien sur les usages de clés. Afin que ces spécifications continuent d'avoir un sens en attendant leur mise à jour, si aucune valeur d'usage de clé n'est spécifiée, les usages de clés 1024 et 1025 doivent être utilisés pour déduire les clés pour le chiffrement et les sommes de contrôle, respectivement. (Ceci ne s'applique pas aux protocoles qui ont leur propre chiffrement indépendamment du présent cadre, en utilisant directement la clé qui résulte de l'échange d'authentification de Kerberos.) De nouveaux protocoles définis en termes de types de chiffrement et de somme de contrôle de Kerberos DEVRAIENT utiliser leurs propres valeurs d'usage de clés.

Sauf indication contraire, aucun chaînage d'état de chiffrement n'est fait d'une opération de chiffrement à une autre.

Note de mise en œuvre : Bien que ce ne soit pas recommandé, certains protocoles d'application vont continuer d'utiliser directement les données de clé, même si c'est seulement dans les spécifications de protocole qui existent actuellement. Une mise en œuvre destinée à prendre en charge les applications Kerberos générales peuvent donc avoir besoin de rendre disponibles les données de clé, ainsi que les attributs et opérations décrits dans la [RFC3961]. Une des raisons les plus courantes pour effectuer directement le chiffrement est le contrôle direct sur la négociation et le choix d'un algorithme de chiffrement "suffisamment fort" (dans le contexte d'une application donnée). Bien que Kerberos ne fournisse pas directement de facilité de négociation des types de chiffrement entre le client et le serveur d'application, il y a des approches qui utilisent Kerberos pour faciliter cette négociation. Par exemple, un client peut ne demander que de types de clé de session "suffisamment forts" au KDC et espérer que tout type retourné par le KDC sera compris et pris en charge par le serveur d'application.

5 Spécifications de message

L'ASN.1 formulé ici devrait être identique au contenu de l'Appendice A. En cas de conflit, le contenu de l'Appendice A devra avoir la priorité.

Le protocole Kerberos est défini ici en termes de notation de syntaxe abstraite n° 1 (ASN.1, *Abstract Syntax Notation One*) [X680], qui fournit une syntaxe de spécification à la fois de la disposition abstraite des messages du protocole et de leurs codages. Les mises en œuvres qui n'utilisent pas un compilateur ASN.1 existant ou une bibliothèque de soutien devraient avoir une bonne compréhension de la spécification ASN.1 réelle afin de s'assurer d'un comportement correct de la mise en œuvre. La notation est plus complexe qu'il ne paraît et certains manuels et guides pour ASN.1 sont trompeurs ou erronés.

Noter qu'en plusieurs endroits, des changements des types abstraits ont été faits par rapport à la RFC 1510. Ceci en partie pour répondre à des hypothèses largement répandues faites par diverses mises en œuvre, qui résultent dans certains cas de violations non intentionnelles de la norme ASN.1. Elles sont clairement mentionnées lorsqu'elles surviennent. Les différences entre les types abstraits de la RFC 1510 et les types abstraits dans le présent document peuvent causer l'émission de codages incompatibles quand on utilise certaines règles de codage, par exemple, les règles de codage compact (PER, *Packed Encoding Rules*). Cette incompatibilité théorique ne devrait pas être pertinente pour Kerberos, car Kerberos spécifie explicitement l'utilisation des règles de codage en métalangage distinctif (DER, *Distinguished Encoding Rules*). Ce peut être un problème pour les protocoles qui cherchent à utiliser les types de Kerberos avec d'autres règles de codage. (Cette pratique n'est pas recommandée.) À très peu d'exceptions près (en particulier l'usage de BIT STRING), les codages résultant de l'utilisation de DER restent identiques entre les types définis dans la RFC 1510 et les types définis dans le présent document.

Les définitions de type de la présente section supposent une définition de module ASN.1 de la forme suivante :

```
KerberosV5Spec2 {
        iso(1) identified-organisation(3) dod(6) internet(1)
        security(5) kerberosV5(2) modules(4) krb5spec2(2)
  } DEFINITIONS EXPLICIT TAGS ::= BEGIN
        -- reste de la définition ici
END
```

Ceci spécifie que le contexte d'étiquetage du module devra être explicite et non automatique.

Noter que dans certaines autres publications (comme la [RFC1510] et la [RFC1964]), la portion "dod" de l'identifiant d'objet est spécifiée par erreur comme ayant la valeur de "5". Dans le cas de la RFC 1964, l'utilisation de la valeur d'OID "correcte" résulterait en un changement du protocole du câble ; elle este donc inchangée pour l'instant.

Noter que ailleurs dans le présent document, la nomenclature de divers types de message est incohérente, mais elle suit largement les conventions du langage C, y compris l'utilisation du caractère souligné (_) et de l'écriture en majuscules de noms destinés à être des constantes numériques. Aussi, à certains endroits, les identifiants (particulièrement ceux qui se réfèrent à des constantes) sont écrits en majuscules afin de les distinguer du texte explicatif environnant.

La notation ASN.1 ne permet pas de soulignés dans les identifiants, aussi dans les définitions ASN.1 réelles, les soulignés sont remplacés par des traits d'union (-). De plus, les noms de membre de la structure et les valeurs définies en ASN.1 DOIVENT commencer par une lettre minuscule, alors que les noms de types DOIVENT commencer par une majuscule.

5.1 Notes spécifiques pour la compatibilité en ASN.1

Pour les besoins de la compatibilité, les mises en œuvre devraient tenir compte des notes spécifiques suivantes concernant l'utilisation de l'ASN.1 dans Kerberos. Ces notes ne décrivent des variantes de l'utilisation standard de l'ASN.1. L'objet de ces notes est plutôt de décrire quelques bizarreries historiques et la non conformité de diverses mises en œuvre, ainsi que des ambiguïtés historiques, qui, bien qu'elles soient de l'ASN.1 valide, peuvent amener une certaine confusion dans la mise en œuvre.

5.1.1 Règles de codage en métalangage distinctif pour l'ASN.1

Le codage des messages du protocole Kerberos doit obéir aux règles de codage en métalangage distinctif (DER, *Distinguished Encoding Rules*) de l'ASN.1 telles que décrites dans [X690]. Certaines mises en œuvre (qui sont principalement celles dérivées de DCE 1.1 et antérieures) sont connues pour utiliser les règles de codage de base (BER, *Basic Encoding Rules*) les plus générales ; en particulier, ces mises en œuvre envoient des codages de longueur indéfinie. Les mises en œuvre PEUVENT accepter de tels codages dans l'intérêt de la rétro compatibilité, bien qu'on doive prévenir les mises en œuvre que le décodage du BER le plus général est plein de périls.

5.1.2 Champs d'entiers facultatifs

Certaines mises en œuvre ne distinguent pas en interne entre une valeur d'entier facultative omise et une valeur transmise de zéro. Les endroits du protocole où ceci est pertinent sont divers champs de microsecondes, de noms occasionnels, et de numéro de séquence. Les mises en œuvre DEVRAIENT traiter les valeurs d'entier facultatives omises comme ayant été transmises avec une valeur de zéro, si c'est ce qu'attend l'application.

5.1.3 Types de SEQUENCE OF vides

Il y a des endroits du protocole où un message contient un type de SEQUENCE OF qui est un membre facultatif. Il peut en résulter un codage contenant une SEQUENCE OF vide. Le protocole Kerberos ne fait pas de distinction sémantique entre un type de SEQUENCE OF facultatif absent et un type de SEQUENCE OF facultatif présent mais vide. Les mises en œuvre NE DEVRAIENT PAS envoyer de codages de SEQUENCE OF vides marqués FACULTATIF, mais DEVRAIENT les accepter comme équivalents à un type FACULTATIF vide. Dans la syntaxe ASN.1 qui décrit les messages Kerberos, les instances de ces types de SEQUENCE OF facultatifs problématiques sont indiquées avec un commentaire.

5.1.4 Numéros d'étiquette non reconnus

De futures révisions du présent protocole pourraient comporter de nouveaux types de message avec des numéros d'étiquette de classe APPLICATION différents. De telles révisions devraient protéger les mises en œuvre plus anciennes en envoyant seulement les types de message que les parties sont connues comprendre ; par exemple, au moyen d'un bit fanion mis par le receveur dans une demande précédente. Dans l'intérêt de la robustesse du traitement d'erreurs, les mises en œuvre DEVRAIENT accepter de traiter de toutes façons un message reçu avec une étiquette non reconnue, et retourner le cas échéant un message d'erreur.

En particulier, les KDC DEVRAIENT retourner KRB_AP_ERR_MSG_TYPE si l'étiquette incorrecte est envoyée sur

un transport TCP. Les KDC NE DEVRAIENT PAS répondre aux messages reçus avec une étiquette inconnue sur un transport UDP afin d'éviter les attaques de déni de service. Pour les applications non KDC, la mise en œuvre de Kerberos indique normalement une erreur à l'application qui prend les mesures appropriées sur la base du protocole d'application.

5.1.5 Numéros d'étiquette supérieurs à 30

Une mise en œuvre toute simple de décodeur ASN.1 DER peut rencontrer des problèmes avec les numéros d'étiquette ASN.1 supérieures à 30, du fait que de tels numéros d'étiquette sont codés en utilisant plus d'un octet. Les révisions futures du présent protocole pourraient utiliser des numéros d'étiquette supérieurs à 30, et les mises en œuvre DEVRAIENT être préparées à retourner une erreur, le cas échéant, lorsque elles ne reconnaissent pas l'étiquette.

5.2 Types Kerberos de base

Ce paragraphe définit un certain nombre de types de base qui sont éventuellement utilisés dans plusieurs messages du protocole Kerberos.

5.2.1 KerberosString

La spécification d'origine du protocole Kerberos dans la RFC 1510 utilise GeneralString à de nombreux endroits pour les données de chaîne lisible par l'homme. Les mises en œuvre historiques de Kerberos ne peuvent pas utiliser toute la puissance de GeneralString. Ce type ASN.1 exige l'utilisation de séquences d'échappement de désignation et d'invocation comme spécifié dans la norme ISO-2022/ECMA-35 [ISO-2022/ECMA-35] pour commuter les ensembles de caractères, et l'ensemble de caractères par défaut qui est désigné sous le nom de G0 est la version de référence internationale (IRV, *International Reference Version*) (a.k.a. U.S. ASCII) de ISO-646/ECMA-6 [ISO-646/ECMA-6] , qui fonctionne.

La norme ISO-2022/ECMA-35 définit quatre éléments de code d'ensembles de caractères (G0 à G3) et deux éléments de code de fonction de contrôle (C0 à C1). DER interdit la désignation des ensembles de caractères autres que les ensembles G0 et C0. Malheureusement, cela semble avoir l'effet collatéral d'interdire l'utilisation de l'ensemble de caractères (ISO Latin) [ISO-8859] ou de tout autre ensemble de caractères qui utilise un jeu de 96 caractères, comme ISO-2022/ECMA-35 interdit de les désigner comme l'élément de code G0. La communauté des normes ASN.1 fait des études sur ces effets collatéraux.

En pratique, de nombreuses mises en œuvre traitent GeneralStrings comme si c'était des chaînes de 8 bits auxquelles les mises en œuvre reviennent par défaut quel que soit le jeu de caractères, sans considération de l'utilisation correcte des séquences d'échappement de désignation de jeu de caractères. Le jeu de caractères par défaut est souvent déterminé par la localisation du système d'exploitation de l'utilisateur habituel. Au moins une mise en œuvre majeure place des caractères Unicode codés en UTF-8 non esquivés dans GeneralString. Cette incapacité à s'en tenir aux spécifications de GeneralString a pour conséquence des problèmes d'interopérabilité lorsque des codages de caractères incompatibles sont utilisés par des clients, services et KDC Kerberos.

Cette situation malheureuse est le résultat d'une documentation défectueuse des restrictions du type ASN.1 de GeneralString dans les spécifications Kerberos précédentes.

Le nouveau (post-RFC 1510) type KerberosString, défini ci-dessous est une GeneralString qui a pour contrainte de ne contenir que des caractères en IA5String.

```
KerberosString          := GeneralString (IA5String)
```

En général, les caractères de contrôle US-ASCII ne devraient pas être utilisés dans KerberosString. Les caractères de contrôle NE DEVRAIENT PAS être utilisés dans des noms de principal ou des noms de domaine.

Pour la compatibilité, les mises en œuvre PEUVENT choisir d'accepter les valeurs de GeneralString qui contiennent des caractères autres que ceux permis par IA5String, mais elles devraient être conscientes que les codes de désignation de jeu de caractères seront vraisemblablement absents, et que le codage devrait probablement presque de toutes façons être traité comme spécifique de la localisation. Les mises en œuvre PEUVENT aussi choisir d'émettre des valeurs de GeneralString qui sont au-delà ce celles permises par IA5String, mais elles devraient être conscientes que faire cela est extraordinairement risqué du point de vue de l'interopérabilité.

Certaines mises en œuvre existantes utilisent GeneralString pour coder des caractères spécifiques de la localisation sans échappement. C'est une violation de la norme ASN.1. La plupart de ces mises en œuvre codent l'US-ASCII dans la moitié gauche, aussi tant que la mise en œuvre ne transmet que de l'US-ASCII, la norme ASN.1 n'est pas violée à cet égard. Lorsque ces mises en œuvre codent les caractères sans échappement spécifiques de la localisation avec le bit de plus fort poids établi, cela viole la norme ASN.1.

D'autres mises en œuvre sont connues pour utiliser GeneralString avec un codage UTF-8. Cela aussi est une violation de la norme ASN.1, car UTF-8 est un codage différent, et pas un jeu "G" à 94 ou 96 caractères tel que défini par la norme ISO 2022. On pense que ces mises en œuvre n'utilisent même pas la séquence d'échappement de ISO 2022 pour changer le codage des caractères. Même si les mises en œuvre devaient annoncer le changement de codage en utilisant la séquence d'échappement, la norme ASN.1 interdit l'utilisation de toute séquence d'échappement autre que celles utilisées pour désigner/invoquer les ensembles "G" ou "C" permis par GeneralString.

De futures révisions de ce protocole permettront presque certainement une représentation plus interopérable des noms de principaux, probablement en y incluant UTF8String.

Noter qu'appliquer une nouvelle contrainte à un type qui n'en avait pas auparavant constitue la création d'un nouveau type ASN.1. Dans ce cas particulier, le changement n'aboutit pas à un changement de codage en DER.

5.2.2 Domaine et PrincipalName

```
Realm           ::= KerberosString
PrincipalName   ::= SEQUENCE {
        name-type       [0] Int32,
        name-string     [1] SEQUENCE OF KerberosString
        }
```

Les noms de domaine Kerberos sont codés comme KerberosStrings. Les domaines ne doivent pas contenir de caractère avec le code 0 (US-ASCII NUL). La plupart des domaines vont normalement comporter plusieurs composants séparés par des points (.), dans le style des noms de domaine Internet, ou séparés par des barres obliques (/), dans le style des noms X.500. Les formes acceptables pour les noms de domaine sont spécifiées au paragraphe 6.1. Un PrincipalName est une séquence typée de composants comportant les sous champs suivants :

name-type *(type de nom)*
Ce champ spécifie le type de nom qui suit. Des valeurs prédéfinies pour ce champ sont spécifiées au paragraphe 6.2. Le name-type DEVRAIT être traité comme un conseil. En ignorant le type de nom, deux noms peuvent être les mêmes (c'est-à-dire, au moins un des composants, ou le domaine, doit être différent).

name-string *(chaîne de nom)*
Ce champ code une séquence de composants qui forment un nom, chaque composant étant codé comme une KerberosString. Pris en semble, un PrincipalName et un domaine forment un identifiant de principal. La plupart des PrincipalNames auront seulement quelques composants (normalement un ou deux).

5.2.3 KerberosTime

```
KerberosTime            ::= GeneralizedTime – sans fraction de seconde
```

Les horodatages utilisés dans Kerberos sont codés comme GeneralizedTimes. Une valeur de KerberosTime ne doit pas inclure de portions fractionnées de la seconde. Comme exigé par les DER, elle ne doit pas non plus inclure de séparateur, et elle doit spécifier la zone de temps UTC (Z). Exemple : Le seul format valide pour l'heure UTC 6 minutes, 27 secondes après 21 h le 6 novembre 1985 est 19851106210627Z.

5.2.4 Contraintes sur des types d'entiers

Certains membres entiers de types DEVRAIENT être contraints à des valeurs représentables en 32 bits, pour la compatibilité avec des limites raisonnables de mise en œuvre.

```
Int32           ::= INTEGER (-2147483648..2147483647)
```

```
                -- valeurs signées représentables en 32 bits

UInt32          ::= INTEGER (0..4294967295)
                -- valeurs de 32 bits non signées

Microseconds    ::= INTEGER (0..999999)
                -- microsecondes
```

Bien qu'il en résulte des changements des types abstraits de la version de la RFC 1510, le codage en DER ne devrait pas être altéré. Les mises en œuvre anciennes étaient normalement limitées de toute façon à des valeurs d'entier de 32 bits, et les numéros alloués DEVRAIENT de toute façon entrer dans l'espace des valeurs entières représentables en 32 bits afin de promouvoir l'interopérabilité.

Plusieurs champs d'entier dans les messages sont contraints à des valeurs fixes.

pvno
: Aussi TKT-VNO ou AUTHENTICATOR-VNO, ce champ récurant est toujours l'entier constant 5. Il n'y a pas de moyen facile pour faire entrer ce champ dans un numéro de version de protocole utile, aussi sa valeur est fixe.

msg-type
: Ce champ d'entier est habituellement identique au numéro d'étiquette d'application du type du message qui le contient.

5.2.5 HostAddress et HostAddresses

```
HostAddress             ::= SEQUENCE  {
        addr-type       [0] Int32,
        address         [1] OCTET STRING
    }

        -- NOTE : HostAddresses est toujours utilisé comme champ OPTIONAL et ne devrait pas être vide.

HostAddresses   -- NOTE : légèrement différent de celui de la rfc1510, mais a une transposition de valeur
                -- et un même codage
                        ::= SEQUENCE OF HostAddress
```

Le codage d'adresse d'hôte comporte deux champs :

addr-type
Ce champ spécifie le type d'adresse suivant. Les valeurs prédéfinies pour ce champ sont spécifiées au paragraphe 7.5.3.

address
Ce champ code une seule adresse du type addr-type.

5.2.6 AuthorizationData

```
        -- NOTE : AuthorizationData est toujours utilisé comme champ OPTIONAL et ne devrait pas être vide.
        AuthorizationData   ::= SEQUENCE OF SEQUENCE {
        ad-type             [0] Int32,
        ad-data             [1] OCTET STRING
        }
```

ad-data
: Ce champ contient des données d'autorisation à interpréter conformément à la valeur du champ ad-type correspondant.

ad-type
: Ce champ spécifie le format du sous-champ ad-data. Toutes les valeurs négatives sont réservées pour usage local. Les valeurs non négatives sont réservées pour un usage enregistré.

Chaque séquence de type et données est perçue comme un élément d'autorisation. Les éléments PEUVENT être spécifiques de l'application ; cependant, il y a un ensemble commun d'éléments récursifs qui devrait être compris par toutes les mises en œuvre. Ces éléments contiennent d'autres éléments qui leur sont incorporés, et l'interprétation de l'élément encapsulant détermine quels éléments incorporés doivent être interprétés, et ceux qui peuvent être ignorés.

Ces éléments communs de données d'autorisation sont définis de façon récursive, signifiant que le ad-data pour ces types contiendra lui-même une séquence de données d'autorisation dont l'interprétation est affectée par l'élément encapsulant. Selon la signification de l'élément encapsulant, les éléments encapsulés peuvent être ignorés, pourraient être interprétés comme produits directement par le KDC, ou pourraient être mémorisés dans une partie séparée de texte en clair du ticket. Les types des éléments encapsulants sont spécifiés au titre de la spécification Kerberos parce que le comportement fondé sur ces valeurs devrait être compris de toutes les mises en œuvre, tandis que d'autres éléments n'ont besoin d'être compris que par les applications qu'ils affectent.

Les éléments de données d'autorisation sont considérés comme critiques si ils sont présents dans un ticket ou un authentifiant. Si un type d'élément de données d'autorisation inconnu est reçu par un serveur dans une AP-REQ ou dans un ticket contenu dans une AP-REQ, alors, sauf si il est encapsulé dans un élément de données d'autorisation connu qui amende la criticité des éléments qu'il contient, l'authentification DOIT échouer. Les données d'autorisation sont destinées à restreindre l'utilisation d'un ticket. Si le service ne peut pas déterminer si la restriction s'applique à ce service, il peut en résulter une faiblesse de la sécurité de l'utilisation de ce ticket pour ce service. Les éléments d'autorisation qui sont facultatifs peuvent être inclus dans un élément AD-IF-RELEVANT.

Dans les définitions qui suivent, la valeur du ad-type pour l'élément sera spécifiée dans la partie dernière phrase du sous-paragraphe, et la valeur du ad-data sera comme indiqué dans la structure ASN.1 qui suit le titre du sous-paragraphe.

Contenu des ad-data	**ad-type**
Codage DER de AD-IF-RELEVANT	1
Codage DER de AD-KDCIssued	4
Codage DER de AD-AND-OR	5
Codage DER de AD-MANDATORY-FOR-KDC	8

5.2.6.1 IF-RELEVANT

AD-IF-RELEVANT ::= AuthorizationData

Les éléments AD encapsulés au sein de l'élément if-relevant sont destinés seulement à être interprétés par les serveurs d'application qui comprennent le ad-type particulier de l'élément incorporé. Les serveurs d'application qui ne comprennent pas le type d'un élément incorporé au sein de l'élément if-relevant PEUVENT ignorer l'élément non interprétable. Cet élément aide à l'interopérabilité des mises en œuvre qui peuvent avoir des extensions locales pour l'autorisation.

Le ad-type pour AD-IF-RELEVANT est (1).

5.2.6.2 KDCIssued

```
AD-KDCIssued        ::= SEQUENCE {
        ad-checksum     [0] Checksum,
        i-realm         [1] Realm OPTIONAL,
        i-sname         [2] PrincipalName OPTIONAL,
        elements        [3] AuthorizationData
        }
```

ad-checksum

Somme de contrôle cryptographique calculée sur le codage DER de AuthorizationData dans le champ "elements", tapée avec la clé de session. Son checksumtype est le type de somme de contrôle obligatoire pour le type de chiffrement de la clé de session, et sa valeur d'usage de clé est 19.

i-realm, i-sname

Le nom du principal qui envoie s'il est différent de celui du KDC lui-même. Ce champ devrait être utilisé lorsque le KDC peut vérifier l'authenticité des éléments signés par le principal qui envoie, et il permet à ce KDC de notifier au serveur d'application la validité de ces éléments.

elements

Séquence d'éléments de données d'autorisation produite par le KDC.

Le champ ad-data produit par le KDC est destiné à fournir un moyen pour que les accréditifs de principal Kerberos incorporent en leur sein les attributs de privilège et autres mécanismes d'autorisation positive, amplifiant les privilèges du principal au-delà de ce qui peut être fait en utilisant des accréditifs sans un tel élément a-data.

Les moyens ci-dessus ne peuvent être fournis sans cet élément à cause de la définition du champ authorization-data qui permet d'ajouter des éléments à volonté par le porteur d'un TGT au moment où il demande les tickets de service, et des éléments peuvent aussi être ajoutés à un ticket délégué par inclusion dans l'authentifiant.

Pour les éléments produits par le KDC, ceci est empêché parce que les éléments sont signés par le KDC en incluant une somme de contrôle chiffrée en utilisant la clé du serveur (la même que celle utilisée pour chiffrer le ticket ou une clé dérivée de cette clé). Les éléments encapsulés avec l'élément produit par le KDC DOIVENT être ignorés par le serveur d'application si cette "signature" n'est pas présente. De plus, les éléments encapsulés au sein de cet élément à partir d'un TGT PEUVENT être interprétés par le KDC, et utilisés comme base, conformément à la politique, pour inclure des éléments nouvellement signés au sein de tickets dérivés, mais ils ne seront pas copiés directement sur un ticket dérivé. Si ils sont copiés directement sur un ticket dérivé par un KDC qui n'est pas au courant de cet élément, la signature ne sera pas correcte pour les éléments de ticket d'application, et le champ sera ignoré par le serveur d'application.

Cet élément et les éléments qu'il encapsule PEUVENT être ignorés en toute sécurité par les applications, serveur d'applications, et KDC qui ne mette pas en œuvre cet élément.

Le ad-type pour AD-KDC-ISSUED est (4).

5.2.6.3 AND-OR

```
AD-AND-OR               ::= SEQUENCE {
        condition-count  [0] Int32,
        elements         [1] AuthorizationData
        }
```

Lorsque des éléments AD restrictifs sont encapsulés au sein de l'élément and-or l'élément and-or est considéré comme satisfait si et seulement si au moins le nombre d'éléments encapsulés spécifié dans condition-count est satisfait. Donc, cet élément PEUT être utilisés pour mettre en œuvre une opération "ou" en réglant le champ condition-count à 1, et il PEUT spécifier une opération "et" en réglant le compte de condition au nombre d'éléments incorporés. Les serveurs d'application qui ne mettent pas en œuvre cet élément DOIVENT rejeter les tickets qui contiennent des éléments de données d'autorisation de ce type.
Le ad-type pour AD-AND-OR est (5).

5.2.6.4 MANDATORY-FOR-KDC

```
AD-MANDATORY-FOR-KDC    ::= AuthorizationData
```

Les éléments AD encapsulés au sein de l'élément mandatory-for-kdc sont à interpréter par le KDC. Les KDC qui ne comprennent pas le type d'un élément incorporé au sein de l'élément mandatory-for-kdc DOIVENT rejeter la demande.
Le ad-type pour AD-MANDATORY-FOR-KDC est (8).

5.2.7 PA-DATA

Historiquement, les PA-DATA étaient connues comme "données de pré-authentification", ce qui signifie qu'elles étaient utilisées pour augmenter l'authentification initiale auprès du KDC. Depuis cette époque, elles ont aussi été utilisées comme trou typé avec lequel on étend les échanges de protocole avec le KDC.

```
PA-DATA                 ::= SEQUENCE {
        -- NOTE : la première étiquette est [1], et non [0]
        padata-type     [1] Int32,
        padata-value    [2] OCTET STRING -- peut être codée AP-REQ
        }
```

padata-type
Indique la façon d'interpréter l'élément padata-value. Les valeurs négatives de padata-type sont réservées pour des utilisations non enregistrées ; les valeurs non négatives sont utilisées pour une interprétation répertoriée du type d'élément.

padata-value
Contient habituellement le codage en DER d'un autre type ; le champ padata-type identifie quel type est codé ici.

type de padata	Nom	Contenu de la valeur padata
1	pa-tgs-req	Codage en DER de AP-REQ
2	pa-enc-horodatage	Codage en DER de PA-ENC-TIMESTAMP
3	pa-pw-salt	sel (non codé en ASN.1)
11	pa-etype-info	Codage en DER de ETYPE-INFO
19	pa-etype-info2	Codage en DER de ETYPE-INFO2

Ce champ PEUT aussi contenir des informations nécessaires à certaines extensions au protocole Kerberos. Par exemple, il peut être utilisé pour vérifier l'identité d'un client avant qu'aucune réponse ne soit retournée.

Le champ padata peut aussi contenir les informations nécessaires pour aider le KDC ou le client à choisir la clé nécessaire pour générer ou déchiffrer la réponse. Cette forme de padata est utile pour la prise en charge de l'utilisation de certaines cartes de jetons avec Kerberos. Les détails de telles extensions sont spécifiés dans des documents séparés. Voir [Pat92] pour d'autres utilisations de ce champ.

5.2.7.1 PA-TGS-REQ

Dans le cas de demandes de tickets supplémentaires (KRB_TGS_REQ), padata-value contiendra une AP-REQ codée. La somme de contrôle dans l'authentifiant (qui DOIT être à l'épreuve des collisions) est à calculer sur le codage de KDC-REQ-BODY.

5.2.7.2 Pré-authentification d'horadatage chiffré

Il y a des types de pré-authentification qui peuvent être utilisés pour pré-authentifier un client au moyen d'un horodatage chiffré.

```
PA-ENC-TIMESTAMP     ::= EncryptedData -- PA-ENC-TS-ENC

PA-ENC-TS-ENC        ::= SEQUENCE {
        patimestamp  [0] KerberosTime -- client's time --,
        pausec       [1] Microseconds OPTIONAL
}
```

Patimestamp contient l'heure du client, et pausec contient les microsecondes, qui PEUVENT être omises si un client ne va pas générer plus d'une demande par seconde. Le texte chiffré (padata-value) comporte le codage de PA-ENC-TS-ENC, chiffré en utilisant la clé secrète du client et une valeur d'usage de clé de 1.

Ce type de pré-authentification n'était pas présent dans la RFC 1510, mais de nombreuses mises en œuvre le prennent en charge.

5.2.7.3 PA-PW-SALT

La padata-value pour ce type de pré-authentification contient le sel pour la chaîne à clé à utiliser par le client pour obtenir la clé pour décrypter la partie chiffrée d'un message AS-REP. Malheureusement, pour des raisons historiques, l'ensemble de caractères à utiliser n'est pas spécifié et probablement spécifique de la localisation.

Ce type de pré-authentification n'était pas présent dans la RFC 1510, mais de nombreuses mises en œuvre le prennent en charge. Il est nécessaire dans tous les cas où le sel pour l'algorithme de chaîne à clé n'est pas celui par défaut.

Dans l'exemple trivial, une chaîne de sel de longueur zéro est très courante pour les domaines qui ont converti leurs bases de données de principal à partir de Kerberos Version 4.

Un KDC NE DEVRAIT PAS envoyer PA-PW-SALT lorsqu'il produit un message KRB-ERROR qui demande une pré-authentification supplémentaire. Note de mise en œuvre : Certaine mises en œuvre de KDC produisent un PA-PW-SALT erroné lorsqu'elle envoient un message KRB-ERROR qui demande une pré-authentification supplémentaire. Donc, les clients DEVRAIENT ignorer un PA-PW-SALT accompagnant un message KRB-ERROR qui demande une pré-authentification supplémentaire. Comme noté au paragraphe 3.1.3, un KDC NE DOIT PAS envoyer PA-PW-SALT lorsque la AS-REQ du client comporte au moins un etype "newer".

5.2.7.4 PA-ETYPE-INFO

Le type de pré-authentification ETYPE-INFO est envoyé par le KDC dans une KRB-ERROR indiquant l'exigence d'une pré-authentification supplémentaire. Il est habituellement utilisé pour notifier à un client quelle clé utiliser pour

le chiffrement d'un horodatage chiffré pour les besoins de l'envoi d'une valeur PA-ENC-TIMESTAMP de pré-authentification. Il PEUT aussi être envoyé dans une AS-REP pour fournir des informations au client sur le sel de clé à utiliser pour la chaîne à clé que le client doit utiliser pour obtenir la clé de déchiffrement de la partie chiffrée de la AS-REP.

```
ETYPE-INFO-ENTRY      ::= SEQUENCE {
        etype         [0] Int32,
        salt          [1] OCTET STRING OPTIONAL
}

ETYPE-INFO            ::= SEQUENCE OF ETYPE-INFO-ENTRY
```

Le sel, comme celui de PA-PW-SALT, est aussi complètement non spécifié par rapport au jeu de caractères et est probablement spécifique de la localisation.

Si ETYPE-INFO est envoyé dans une AS-REP, il doit être exactement une ETYPE-INFO-ENTRY, et son etype doit correspondre à celui de la enc-part dans la AS-REP.

Ce type de pré-authentification n'était pas présent dans la RFC 1510, mais de nombreuses mises en œuvre qui acceptent les horodatages chiffrés pour la pré-authentification ont aussi besoin de prendre en charge ETYPE-INFO. Comme noté au paragraphe 3.1.3, un KDC NE DOIT PAS envoyer PA-ETYPE-INFO lorsque la AS-REQ du client comporte au moins un etype "newer".

5.2.7.5 PA-ETYPE-INFO2

Le type de pré-authentification ETYPE-INFO2 est envoyé par le KDC dans une KRB-ERROR indiquant l'exigence d'une pré-authentification supplémentaire. Il est habituellement utilisé pour notifier à un client quelle clé utiliser pour le chiffrement d'un horodatage chiffré pour les besoins de l'envoi d'une valeur PA-ENC-TIMESTAMP de pré-authentification. Il PEUT aussi être envoyé dans une AS-REP pour fournir des informations au client sur le sel de clé à utiliser pour la chaîne à clé que le client doit utiliser pour obtenir la clé de déchiffrement de la partie chiffrée de la AS-REP.

```
ETYPE-INFO2-ENTRY     ::= SEQUENCE {
        etype         [0] Int32,
        salt          [1] KerberosString OPTIONAL,
        s2kparams     [2] OCTET STRING OPTIONAL
}

ETYPE-INFO2           ::= SEQUENCE SIZE (1..MAX) OF ETYPE-INFO2-ENTRY
```

Le type du sel est KerberosString, mais les installations existantes peuvent avoir des caractères spécifiques de la localisation mémorisés dans les chaînes salées, et les développeurs PEUVENT choisir de les traiter.

L'interprétation de s2kparams est spécifiée dans la description du crypto système associé au etype. Chaque crypto système a une interprétation par défaut de s2kparams qui restera si cet élément est omis dans le codage de ETYPE-INFO2-ENTRY.

Si ETYPE-INFO2 est envoyé dans une AS-REP, il doit être exactement une ETYPE-INFO2-ENTRY, et son etype doit correspondre à celui de la enc-part dans la AS-REP.

L'ordre préféré des données de pré-authentification "conseillé" qui affecte le choix de clé du client est : ETYPE-INFO2, suivi par ETYPE-INFO, suivi par PW-SALT. Comme noté au paragraphe 3.1.3, un KDC NE DOIT PAS envoyer de ETYPE-INFO ou PW-SALT lorsque la AS-REQ du client comporte au moins un etype "newer".

Le type de pré-authentification ETYPE-INFO2 n'était pas présent dans la RFC 1510.

5.2.8 KerberosFlags

Pour plusieurs types de messages, un type spécifique de chaîne binaire obligée, KerberosFlags, est utilisé.

```
KerberosFlags         ::= BIT STRING (SIZE (32..MAX))
```

-- nombre minimum de bits qui doivent être envoyés, mais pas moins de 32

Note de compatibilité : Les paragraphes suivants décrivent un changement par rapport à la description de la RFC 1510 des chaînes binaires qui résulteraient en une incompatibilité dans le cas d'une mise en œuvre strictement conforme au DER de l'ASN.1 et à la RFC 1510.

Les chaînes binaires ASN.1 ont plusieurs usages. L'usage le plus simple d'une chaîne binaire est de contenir un vecteur de bits, sans signification particulière attachée aux bits individuels. Ce vecteur de bits n'est pas nécessairement un multiple de huit bits de longueur. L'utilisation par Kerberos d'une chaîne binaire comme vecteur booléen compact dans lequel chaque élément a une signification distincte pose quelques problèmes. La notation naturelle pour un vecteur booléen compact est la notation ASN.1 "NamedBit", et les DER exigent que les codages d'une chaîne binaire qui utilise la notation "NamedBit" excluent tous bits à zéro en queue. Il est facile de négliger cette troncature, tout particulièrement dans les mises en œuvre de langage C qui choisissent naturellement de mémoriser les vecteurs booléens comme des entiers de 32 bits.

Par exemple, si la notation de KDCOptions devait inclure la notation "NamedBit", comme dans la RFC 1510, et si la valeur KDCOptions à coder avait seulement le bit "transmissible" (bit numéro un) établi, le codage DER DOIT n'inclure que deux bits : le premier bit réservé ("réservé", bit numéro zéro, valeur zéro) et le bit de valeur un (bit numéro un) pour "transmissible".

La plupart des mises en œuvre existantes de Kerberos envoient inconditionnellement 32 bits sur le réseau lors du codage de chaînes binaires utilisées comme vecteurs booléens. Ce comportement viole la syntaxe ASN.1 utilisée pour les valeurs de fanion dans la RFC 1510, mais cela survient si fréquemment que la description du protocole en est modifiée pour s'en accommoder.

Par conséquent, le présent document retire la notation "NamedBit" pour les bits individuels, les reléguant en commentaires. La contrainte de taille sur le type KerberosFlags exige qu'au moins 32 bits soient codés à tout moment, bien qu'une mise en œuvre laxiste PUISSE choisir d'accepter moins de 32 bits et de traiter les bits manquants comme mis à zéro.

Actuellement, aucune utilisation de KerberosFlags ne spécifie plus de 32 bits de fanion, bien que de futures révisions du présent document puissent le faire. Lorsque plus de 32 bits sont à transmettre dans une valeur de KerberosFlags, les futures révisions du présent document spécifieront vraisemblablement que le plus petit nombre de bits nécessaires pour coder le bit de valeur un de plus haut rang devrait être envoyé. Ceci est assez similaire au codage en DER d'une chaîne binaire qui est déclarée avec la notation "NamedBit".

5.2.9 Cryptosystem-Related Types

De nombreux messages de protocole Kerberos contiennent un EncryptedData comme conteneur pour des données chiffrées de façon arbitraire, qui sont souvent le codage chiffré d'un autre type de données. Les champs au sein de EncryptedData assistent le receveur dans le choix d'une une clé avec laquelle déchiffrer les données incluses.

```
EncryptedData           ::= SEQUENCE {
        etype           [0] Int32 -- EncryptionType --,
        kvno            [1] UInt32 OPTIONAL,
        cipher          [2] OCTET STRING – texte chiffré
}
```

etype
Ce champ identifie quel algorithme de chiffrement a été utilisé pour chiffrer le texte chiffré.

kvno
Ce champ contient le numéro de version de la clé selon laquelle les données sont chiffrées. Il n'est présent que dans les messages chiffrés avec des clés à longue durée, comme les clés secrètes de principal.

cipher
Ce champ contient le texte chiffré, codé comme une OCTET STRING *(chaîne d'octets)*. (Noter que le mécanisme de chiffrement défini dans la RFC 3961 DOIT incorporer aussi la protection d'intégrité, aussi aucune somme de contrôle supplémentaire n'est exigée.)

Le type EncryptionKey est le moyen par lequel les clés cryptographiques utilisées pour le chiffrement sont transférées.

```
EncryptionKey           ::= SEQUENCE {
        keytype         [0] Int32 -- actually encryption type --,
        keyvalue        [1] OCTET STRING
}
```

keytype
Ce champ spécifie le type de chiffrement de la clé de chiffrement qui suit dans le champ keyvalue. Bien que son nm soit "keytype", il spécifie en réalité le type de chiffrement. Précédemment, plusieurs systèmes de chiffrement qui effectuaient le chiffrement différemment, mais étaient capables d'utiliser des clés avec les mêmes caractéristiques étaient permis en partageant un numéro alloué pour désigner le type de clé ; cet usage est maintenant déconseillé.

keyvalue
Ce champ contient la clé elle-même, codée comme une chaîne d'octets.

Les messages qui contiennent des données de texte en clair à authentifier le feront habituellement en utilisant un membre du type Checksum. La plupart des instances de Checksum utilisent un hachage chiffré, bien que des exceptions existent.

```
Checksum                ::= SEQUENCE {
        cksumtype       [0] Int32,
        checksum        [1] OCTET STRING
  }
```

cksumtype
Ce champ indique l'algorithme utilisé pour générer la somme de contrôle qui l'accompagne.

checksum
Ce champ contient la somme de contrôle elle-même, codée comme une chaîne d'octets.

Voir à la Section 4 une brève description de l'utilisation du chiffrement et des sommes de contrôle dans Kerberos.

5.3 Tickets

Ce paragraphe décrit les paramètres de format et de chiffrement pour les tickets et les authentifiants. Lorsque un ticket ou un authentifiant est inclus dans un message de protocole, il est traité comme un objet opaque. Un ticket est un enregistrement qui aide un client à s'authentifier auprès d'un service. Un ticket contient les informations suivantes :

```
Ticket                  ::= [APPLICATION 1] SEQUENCE {
        tkt-vno         [0] INTEGER (5),
        realm           [1] Realm,
        sname           [2] PrincipalName,
        enc-part        [3] EncryptedData -- EncTicketPart
  }

-- Partie chiffrée du ticket
EncTicketPart           ::= [APPLICATION 3] SEQUENCE {
        flags           [0] TicketFlags,
        key             [1] EncryptionKey,
        crealm          [2] Realm,
        cname           [3] PrincipalName,
        transited       [4] TransitedEncoding,
        authtime        [5] KerberosTime,
        starttime       [6] KerberosTime OPTIONAL,
        endtime         [7] KerberosTime,
        renew-till      [8] KerberosTime OPTIONAL,
        caddr           [9] HostAddresses OPTIONAL,
        authorization-data [10] AuthorizationData OPTIONAL
```

```
  }

-- champs traversés codés
TransitedEncoding           ::= SEQUENCE {
        tr-type             [0] Int32 -- doit être enregistré --,
        contents            [1] OCTET STRING
  }

TicketFlags     ::= KerberosFlags
                -- reserved(0),
                -- forwardable(1),
                -- forwarded(2),
                -- proxiable(3),
                -- proxy(4),
                -- may-postdate(5),
                -- postdated(6),
                -- invalid(7),
                -- renewable(8),
                -- initial(9),
                -- pre-authent(10),
                -- hw-authent(11),
-- les fanions suivants sont nouveaux depuis la RFC 1510
                -- transited-policy-checked(12),
                -- ok-as-delegate(13)
```

tkt-vno
Ce champ spécifie le numéro de version pour le format de ticket. Ce document décrit la version numéro 5.

realm
Ce champ spécifie le domaine qui a produit un ticket. Il sert aussi à identifier la partie de domaine de l'identifiant de principal du serveur. Comme un serveur Kerberos ne peut produire de tickets que pour des serveurs au sein de son domaine, les deux seront toujours identiques.

sname
Ce champ spécifie tous les composants de la partie nom de l'identité du serveur, incluant les parties qui identifient une instance spécifique d'un service.

enc-part
Ce champ contient le codage chiffré de la séquence EncTicketPart. Il est chiffré avec la clé partagée par Kerberos et le serveur d'extrémité (la clé secrète du serveur), en utilisant une clé de valeur d'usage de 2.

flags
Ce champ indique lesquelles des diverses options ont été utilisées ou demandées lorsque le ticket a été produit. La signification des fanions est la suivante :

Bits	**Nom**	**Description**
0	réservé	Réservé pour une expansion future de ce champ.
1	transmissible	Le fanion FORWARDABLE n'est normalement interprété que par le TGS, et peut être ignoré par les serveurs d'extrémité. Lorsqu'il est mis, ce fanion dit au serveur d'allocation de tickets que c'est d'accord pour produire un nouveau TGT avec une adresse réseau différente fondée sur le ticket présenté.
2	transmis	Lorsqu'il est mis, ce fanion indique que le ticket a été transmis ou a été produit sur la base d'une authentification impliquant un TGT transmis.
3	mandatable	Le fanion PROXIABLE n'est normalement interprété que par le TGS, et peut être ignoré par les serveurs d'extrémité. Le fanion PROXIABLE a une interprétation identique à celle du fanion FORWARDABLE, excepté que le fanion PROXIABLE dit au serveur d'allocation de ticket que seuls des non TGT peuvent être produits avec des adresses réseau différentes.
4	mandaté	Lorsque il est mis, ce fanion indique qu'un ticket est un mandataire.
5	peut-postdater	Le fanion PEUT-POSTDATER n'est normalement interprété que par le TGS, et peut

		être ignoré par les serveurs d'extrémité. Ce fanion dit au serveur d'allocation de ticket qu'un ticket post-daté PEUT être produit sur la base de ce TGT.
6	postdaté	Ce fanion indique que ce ticket a été postdaté. Le service d'extrémité peut vérifier le champ authtime pour voir quand l'authentification originelle est survenue.
7	invalide	Ce fanion indique qu'un ticket est invalide, et qu'il doit être validé par le KDC avant utilisation. Les serveurs d'application doivent rejeter les tickets qui ont ce fanion.
8	renouvelable	Le fanion RENOUVELABLE n'est normalement interprété que par le TGS, et peut habituellement être ignoré par les serveurs d'extrémité (certains serveurs particulièrement soupçonneux PEUVENT interdire les ticket renouvelables). Un ticket renouvelable peut être utilisé pour obtenir un ticket de remplacement qui expire à une date ultérieure.
9	initial	Ce fanion indique que ce ticket a été produit en utilisant le protocole d'AS, et non pas produit sur la base d'un TGT.
10	pré-authentification	Ce fanion indique que durant l'authentification initiale, le client a été authentifié par le KDC avant la production d'un ticket. La force de la méthode de pré-authentification n'est pas indiquée, mais est acceptable pour le KDC.
11	authentification-du-matériel	Ce fanion indique que le protocole employé pour l'authentification initiale exige l'utilisation d'un matériel dont on espère qu'il n'est possédé que par le client désigné. La méthode d'authentification de matériel est choisie par le KDC et la force de la méthode n'est pas indiquée.
12	politique-de-transit-vérifiée	Ce fanion indique que le KDC pour le domaine a vérifié le champ de transit par rapport à une politique définie par le domaine en matière de certificateurs de confiance. Si ce fanion est ôté (0), le serveur d'application doit alors vérifier le champ de transit lui-même, et si il est incapable de le faire, il doit rejeter l'authentification. Si le fanion est mis (1), le serveur d'application PEUT alors sauter sa propre validation du champ de transit, en s'appuyant sur la validation effectuée par le KDC. À son choix, le serveur d'application PEUT encore appliquer sa propre validation fondée sur une politique d'acceptation distincte. Ce fanion est nouveau par rapport à la RFC 1510.
13	ok-comme-délégué	Ce fanion indique que le serveur (et non le client) spécifié dans le ticket a été déterminé par la politique du domaine comme étant un receveur de délégation convenable. Un client peut utiliser la présence de ce fanion pour l'aider à décider de déléguer des accréditifs (accorder un mandat ou transmettre un TGT) à ce serveur. Le client est libre d'ignorer la valeur de ce fanion. Lorsqu'il met ce fanion, un administrateur devrait considérer la sécurité et le placement du serveur sur lequel le service va fonctionner, ainsi que si le service requiert l'utilisation d'accréditifs délégués. Ce fanion est nouveau par rapport à la RFC 1510.
14-31	réservé	Réservé à une utilisation future.

key
Ce champ existe dans le ticket et la réponse du KDC et il est utilisé pour passer la clé de session de Kerberos au serveur d'application et au client.

crealm
Ce champ contient le nom du domaine dans lequel le client est enregistré et dans lequel l'authentification initiale a eu lieu.

cname
Ce champ contient la partie nom de l'identifiant de principal du client.

transited
Ce champ donne la liste des noms des domaines Kerberos qui ont pris part à l'authentification de l'usager à qui ce ticket a été produit. Il ne spécifie pas l'ordre dans lequel les domaines ont été traversés. Voir au paragraphe 3.3.3.2 les détails de la façon dont ce champ code les domaines traversés. Quand les noms des CA sont à incorporer dans le champ de transit (comme spécifié pour certaines extensions au protocole), les noms X.500 des CA DEVRAIENT être transposés en éléments dans le champ de transit en utilisant la transposition définie par la RFC 2253.

authtime
Ce champ indique l'heure de l'authentification initiale du principal désigné. C'est l'heure de production du ticket original sur lequel est fondé ce ticket. Il est inclus dans le ticket pour fournir des informations supplémentaires au service d'extrémité, et pour fournir les informations nécessaires pour la mise en œuvre d'un service de "liste rouge" au

KDC. Un service d'extrémité qui serait particulièrement paranoïaque pourrait refuser d'accepter des tickets pour lesquels l'authentification initiale serait survenue "trop loin" dans le passé. Ce champ est aussi retourné au titre de la réponse du KDC. Quand il est retourné au titre de la réponse à l'authentification initiale (KRB_AS_REP), c'est l'heure en cours au serveur Kerberos. Il N'EST PAS recommandé que cette valeur horaire soit utilisée pour régler l'horloge de la station de travail, car la station de travail ne peut pas déterminer de façon fiable qu'une telle KRB_AS_REP vienne réellement du KDC approprié au bon moment.

starttime
Ce champ dans le ticket spécifie l'heure après laquelle le ticket sera valide. Avec le champ endtime, ce champ spécifie la durée de vie du ticket. Si le champ starttime est absent du ticket, le champ authtime DEVRAIT alors être utilisé à sa place pour déterminer la durée de vie du ticket.

endtime
Ce champ contient l'heure après laquelle le ticket ne sera plus honoré (son heure d'expiration). Noter que des services individuels PEUVENT placer leurs propres limites à la vie d'un ticket et PEUVENT rejeter des tickets qui ne sont pas encore arrivés à expiration. Comme telle, c'est réellement une limite supérieure de l'heure d'expiration du ticket.

renew-till
Ce champ n'est présent que dans les tickets qui ont le fanion RENEWABLE mis dans le champ des fanions. Il indique l'heure de fin maximum qui peut être incluse dans un renouvellement. Il peut aussi être vu comme l'heure d'expiration absolue pour le ticket, y compris tous les renouvellements.

caddr
Ce champ dans un ticket contient zéro (si il est omis) ou plusieurs (s'il est présent) adresses d'hôte. Ce sont les adresses à partir desquelles le ticket peut être utilisé. Si il n'y a pas d'adresses, le ticket peut être utilisé à partir de toute localisation. La décision du KDC de produire, ou par le serveur d'extrémité d'accepter des tickets sans adresse est une décision de politique qui est laissée à Kerberos et aux administrateurs de service d'extrémité ; ils PEUVENT refuser de produire ou d'accepter de tels tickets. À cause du large développement de la traduction des adresses réseau, il est recommandé que les politiques permettent la production et l'acceptation de tels tickets.

Les adresses réseau sont incluses dans le ticket pour rendre plus difficile à un agresseur d'utiliser des accréditifs volés. Comme la clé de session n'est pas envoyée sur le réseau en clair, les accréditifs ne peuvent pas être volés simplement en écoutant le réseau ; un agresseur doit gagner l'accès à une clé de session (peut-être à travers des failles de la sécurité du système d'exploitation ou une session non surveillée d'un utilisateur négligeant) pour utiliser des tickets volés.

Noter que l'adresse réseau à partir de laquelle une connexion est reçue ne peut pas être déterminée de façon fiable. Même si elle pouvait l'être, un agresseur qui a compromis la station de travail du client pourrait utiliser les accréditifs à partir de là. Inclure les adresses réseau ne fait que rendre plus difficile, mais pas impossible, à un agresseur de sortir avec des accréditifs volés et de les utiliser ensuite à partir d'une localisation "sûre".

authorization-data
Le champ authorization-data est utilisé pour passer les données d'autorisation du principal au nom duquel un ticket a été produit au service d'application. Si aucune donnée d'autorisation n'est incluse, ce champ sera laissé de côté. L'expérience nous montre que le nom de ce champ prête à confusion, et un meilleur nom serait "restrictions". Malheureusement, il n'est pas possible de changer le nom pour le moment.

Ce champ contient des restrictions à toute autorité obtenue sur la base d'une authentification utilisant le ticket. Il est possible à tout principal en possession d'accréditifs d'ajouter des entrées au champ de données d'autorisation car ces entrées prolongent les restrictions qui peuvent être faites avec le ticket. De telles additions peuvent être faites en spécifiant les entrées supplémentaires lorsque un nouveau ticket est obtenu durant l'échange TGS, ou elles PEUVENT être ajoutées durant une délégation chaînée utilisant le champ de données d'autorisation de l'authentifiant.

Comme les entrées peuvent être ajoutées à ce champ par le détenteur des accréditifs, sauf lorsque une entrée est authentifiée séparément par encapsulation dans l'élément produit par le KDC, il n'est pas admissible pour la présence d'une entrée dans le champ de données d'autorisation d'un ticket d'amplifier les privilèges qu'on obtiendrait de l'utilisation d'un ticket.

Les données dans ce champ peuvent être spécifiques du service d'extrémité ; le champ contiendra les noms des objets spécifiques du service, et les droits de ces objets. Le format pour ce champ est décrit au paragraphe 5.2.6. Bien que Kerberos ne soit pas concerné par le format du contenu des sous-champs, il en porte les informations de type (ad-type).

En utilisant le champ authorization_data, un principal est capable de produire un mandat valide pour un objet spécifique. Par exemple, un client souhaitant imprimer un fichier peut obtenir qu'un mandat de serveur de fichiers soit passé au serveur d'impression. En spécifiant le nom du fichier dans le champ authorization_data, le serveur de fichiers sait que le serveur d'impression peut seulement utiliser les droits du client lorsque il accède au fichier particulier à imprimer.

On peut construire un service séparé qui fournit l'autorisation ou la certification d'appartenance à un groupe en utilisant le champ authorization-data. Dans ce cas, l'entité qui accorde l'autorisation (et non l'entité autorisée) peut obtenir un ticket en son nom propre (par exemple, le ticket est produit au nom d'un serveur privilégié), et cette entité ajoute des restrictions à sa propre autorité et délègue au client l'autorité restreinte à travers un mandataire. Le client présenterait alors cet accréditif d'autorisation au serveur d'application séparément de l'échange d'authentification. Autrement, des tels accréditifs d'autorisation PEUVENT être incorporés dans le ticket qui authentifie l'entité autorisée, lorsque l'autorisation est authentifiée séparément en utilisant l'élément de données d'autorisation produit par le KDC (voir au paragraphe 5.2.6.2).

De même, si on spécifie le champ authorization-data d'un mandataire et qu'on laisse en blanc les adresses d'hôte, le ticket et la clé de session qui résultent peuvent être traités comme une capacité. Voir dans [Neu93] quelques utilisations suggérées pour ce champ.

Le champ authorization-data est facultatif et n'a pas à être inclus dans un ticket.

5.4 Spécifications pour les échanges AS et TGS

Ce paragraphe spécifie le format des messages utilisés dans l'échange entre le client et le serveur Kerberos. Le format des messages d'erreur possibles figure au paragraphe 5.9.1.

5.4.1 Définition de KRB_KDC_REQ

Le message KRB_KDC_REQ n'a pas de numéro d'étiquette d'application par lui-même. Il est incorporé dans KRB_AS_REQ ou KRB_TGS_REQ, qui ont chacune une étiquette d'application, selon que la demande est celle d'un ticket initial ou d'un ticket supplémentaire. Dans les deux cas, le message est envoyé du client au KDC pour demander des accréditifs pour un service.

Les champs de message sont les suivants :

```
AS-REQ                  ::= [APPLICATION 10] KDC-REQ
TGS-REQ                 := [APPLICATION 12] KDC-REQ

KDC-REQ                 ::= SEQUENCE {                  -- NOTE: la première étiquette est [1], pas [0]
        pvno            [1] INTEGER (5) ,
        msg-type        [2] INTEGER (10 -- AS -- | 12 -- TGS --),
        padata          [3] SEQUENCE OF PA-DATA OPTIONAL    -- NOTE : non vide --,
        req-body        [4] KDC-REQ-BODY
}

KDC-REQ-BODY            ::= SEQUENCE {
      kdc-options       [0] KDCOptions,
      cname             [1] PrincipalName OPTIONAL      -- Ne sert que dans AS-REQ --,
      realm             [2] Realm                 -- Domaine du serveur et aussi du client dans AS-REQ --,
      sname             [3] PrincipalName OPTIONAL,           from [4] KerberosTime OPTIONAL,
      till              [5] KerberosTime,
      rtime             [6] KerberosTime OPTIONAL,
      nonce             [7] UInt32,
      etype             [8] SEQUENCE OF Int32       -- Type de chiffrement dans l'ordre de préférence --,
      addresses         [9] HostAddresses OPTIONAL,
      enc-authorization-data [10] EncryptedData OPTIONAL          -- AuthorizationData --,
      additional-tickets [11] SEQUENCE OF Ticket OPTIONAL   -- NOTE : non vide
```

```
}

KDCOptions          ::= KerberosFlags
                        -- reserved(0),
                        -- forwardable(1),
                        -- forwarded(2),
                        -- proxiable(3),
                        -- proxy(4),
                        -- allow-postdate(5),
                        -- postdated(6),
                        -- unused7(7),
                        -- renewable(8),
                        -- unused9(9),
                        -- unused10(10),
                        -- opt-hardware-auth(11),
                        -- unused12(12),
                        -- unused13(13),
                        -- 15 est réservé pour la canonisation
                        -- unused15(15),
                        -- 26 n'était pas utilisé dans la rfc 1510
                        -- disable-transited-check(26),--
                        -- renewable-ok(27),
                        -- enc-tkt-in-skey(28),
                        -- renew(30),
                        -- validate(31)
```

Les champs dans ce message sont les suivants :

pvno
Ce champ est inclus dans chaque message, et spécifie le numéro de version du protocole. Ce document spécifie la version 5 du protocole.

msg-type
Ce champ indique le type d'un message de protocole. Il sera presque toujours le même que l'identifiant d'application associé à un message. Il est inclus pour rendre l'identifiant plus facilement accessible à l'application. Pour le message KDC-REQ, ce type sera KRB_AS_REQ ou KRB_TGS_REQ.

padata
Contient les données de pré-authentification. Les demandes de tickets supplémentaires (KRB_TGS_REQ) DOIVENT contenir un padata de PA-TGS-REQ.

Le champ padata (données de pré-authentification) contient une séquence d'informations d'authentification qui peuvent être nécessaires avant que les accréditifs puissent être produits ou décryptés.

req-body
Ce champ est un paramètre fictif qui délimite l'extension des champs restants. Si une somme de contrôle est à calculer sur la demande, elle est calculée sur un codage de la séquence KDC-REQ-BODY qui est enclose dans le champ req-body.

kdc-options
Ce champ apparaît dans les demandes KRB_AS_REQ et KRB_TGS_REQ au KDC et indique les fanions que le client veut mettre sur les tickets ainsi que les autres informations qui vont modifier le comportement du KDC. Lorsque c'est approprié, le nom d'une option peut être le même que celui du fanion qui est mis par cette option. Bien que dans la plupart des cas le bit dans le champ options soit le même que celui du champ flags, cela n'est pas garanti, aussi il n'est pas acceptable de simplement copier le champ options dans le champ flags. Diverses vérifications doivent être faites avant qu'une option ne soit honorée.

Le champ kdc_options est un champ binaire, où les options choisies sont indiquées par le bit mis à (1), et les options non choisies et les champs réservés ne sont pas établis (mis à 0). Le codage des bits est spécifié au paragraphe 5.2. Les options sont décrites plus en détail à la Section 2 ci-dessus. La signification des options est la suivante :

Bits	Nom	Description
0	RÉSERVÉ	Réservé pour l'expansion future de ce champ.
1	TRANSMISSIBLE	L'option TRANSMISSIBLE indique que le ticket à produire aura son fanion transmissible établi. Il peut n'être établi que dans la demande initiale, ou dans une demande ultérieure si le TGT sur lequel il est fondé est aussi transmissible.
2	TRANSMIS	L'option TRANSMIS n'est spécifiée que dans une demande au serveur d'allocation de tickets et ne sera honorée que si le TGT dans la demande à son bit TRANSMISSIBLE mis. Cette option indique qu'il s'agit d'une demande de transmission. La ou les adresses de l'hôte de qui le ticket résultant doit être valide sont incluses dans le champ d'adresses de la demande.
3	MANDATABLE	L'option MANDATABLE indique que le ticket à produire doit avoir son fanion mandatable mis. Il peut n'être établi que dans la demande initiale, ou dans une demande ultérieure si le TGT sur lequel il se fonde est aussi mandatable.
4	MANDATAIRE	L'option MANDATAIRE indique qu'il s'agit d'une demande de mandataire. Cette option ne sera honorée que si le TGT dans la demande a son bit MANDATABLE mis. La ou les adresses de l'hôte de qui le ticket résultant doit être valide sont incluses dans le champ d'adresses de la demande.
5	PEUT-POSTDATER	L'option PEUT-POSTDATER indique que le ticket à produire aura son fanion PEUT-POSTDATER mis. Il peut n'être établi que sur la demande initiale, ou dans une demande suivante si le TGT sur lequel il se fonde a aussi son fanion PEUT-POSTDATER mis.
6	POSTDATÉ	L'option POSTDATÉ indique qu'il s'agit d'une demande de ticket postdaté. Cette option ne sera honorée que si le TGT sur lequel elle se fonde a son fanion PEUT-POSTDATER mis. Le ticket résultant aura aussi son fanion INVALIDE mis, et ce fanion peut être remis à zéro par une demande ultérieure au KDC après que l'heure de début du ticket ait été atteinte.
7	RÉSERVÉ	Cette option est présentement inutilisée.
8	RENOUVELABLE	L'option RENOUVELABLE indique que le ticket à produire aura son fanion RENOUVELABLE mis. Il peut n'être établi que sur la demande initiale, ou lorsque le TGT sur lequel se fonde la demande est aussi renouvelable. Si cette option est demandée, le champ rtime dans la demande contient alors l'heure d'expiration absolue désirée pour le ticket.
9	RÉSERVÉ	Réservé pour PK-Cross.
10	RÉSERVÉ	Réservé pour utilisation future.
11	RÉSERVÉ	Réservé pour opt-hardware-auth.
12-25	RÉSERVÉ	Réservé pour utilisation future.
26	VÉRIFICATION-DE-TRANSIT-DÉSACTIVÉE	Par défaut, le KDC vérifiera le champ de transit d'un TGT par rapport à la politique du domaine local avant de délivrer les tickets déduits sur la base du TGT. Si ce fanion est mis dans la demande, la vérification du champ de transit est désactivée. Les tickets produits sans effectuer cette vérification seront notés par la valeur (0) du fanion POLITIQUE-DE-TRANSIT-VÉRIFIÉE, indiquant au serveur d'application que le champ de transit doit être vérifié localement. Les KDC sont invités à honorer l'option VÉRIFICATION-DE-TRANSIT-DÉSACTIVÉE, mais elle n'est pas exigée. Ce fanion est nouveau par rapport à la RFC 1510.
27	RENOUVELABLE-OK	Le fanion RENEVELABLE-OK indique qu'un ticket renouvelable sera acceptable si un ticket avec la durée de vie requise ne peut autrement être fourni, auquel cas un ticket renouvelable peut être produit avec un renouveler-jusqu'à égal à l'heure de fin demandée. La valeur du champ renouveler-jusqu'à peut encore être limitée par des limites locales, ou des limites choisies par le principal ou serveur individuel.
28	ENC-TKT-IN-SKEY	Cette option n'est utilisée que par le service d'allocation de tickets. L'option ENC-TKT-IN-SKEY indique que le ticket pour le serveur d'extrémité est à chiffrer avec la clé de session provenant du TGT supplémentaire fourni.
29	RÉSERVÉ	Réservé pour une utilisation future.
30	RENOUVELER	Cette option n'est utilisée que par le service d'allocation de tickets. L'option RENOUVELER indique que la présente demande est d'un renouvellement. Le ticket fourni est chiffré avec la clé secrète sur le serveur sur lequel elle est valide. Cette option ne sera honorée que si le ticket à renouveler a son fanion

		RENOUVELABLE mis et si l'heure de son champ renouveler-jusqu'à n'est pas passée. Le ticket à renouveler est passé dans le champ padata au titre de l'en-tête d'authentification.
31	VALIDER	Cette option n'est utilisée que par le service d'allocation de tickets. L'option VALIDER indique que la demande est de valider un ticket postdaté. Elle ne sera honorée que si le ticket présenté est postdaté, a présentement son fanion INVALIDE mis, et serait par ailleurs utilisable à ce moment. Un ticket ne peut pas être validé avant son heure de début. Le ticket présenté pour validation est chiffré avec la clé du serveur pour lequel elles est valide et est passé dans le champ padata au titre de l'en-tête d'authentification.

cname et sname
Ces champ sont les mêmes que ceux décrits pour le ticket au paragraphe 5.3. Le sname ne peut être absent que lorsque l'option ENC-TKT-IN-SKEY est spécifiée. Si le sname est absent, le nom du serveur est tiré du nom du client dans le ticket passé comme ticket supplémentaire.

enc-authorization-data
Le champ enc-authorization-data, s'il est présent (et il ne peut être présent que sous la forme de TGS_REQ), est un codage des authorization-data désirées chiffré avec la sous-clé de session si elle est présente dans l'authentifiant, ou autrement avec la clé de session dans le TGT (l'authentifiant et le TGT viennent tous deux du champ padata dans la KRB_TGS_REQ). La valeur d'usage de clé utilisée pour le chiffrement est 5 si une sous-clé de session est utilisée, ou 4 si la clé de session est utilisée.

realm
Ce champ spécifie la partie domaine de l'identifiant de principal du serveur. Dans l'échange d'AS, c'est aussi la partie domaine de l'identifiant de principal du client.

from
Ce champ est inclus dans les demandes de ticket KRB_AS_REQ et KRB_TGS_REQ lorsque le ticket demandé est à postdater. Il spécifie l'heure de début désirée pour le ticket demandé. Si ce champ est omis, le KDC DEVRAIT alors utiliser à la place l'heure en cours.

till
Ce champ contient la date d'expiration demandée par le client dans une demande de ticket. Il n'est pas facultatif, mais si l'heure de fin demandée est "19700101000000Z", le ticket demandé aura l'heure de fin maximum permise selon la politique du KDC. Note de mise en œuvre : Cet horodatage spécial correspond à une valeur UNIX time_t de zéro sur la plupart des systèmes.

rtime
Ce champ est l'heure de renouvellement-jusqu'à demandée envoyée d'un client au KDC dans une demande de ticket. Il est facultatif.

nonce
Ce champ fait partie de la demande et de la réponse du KDC. Il est destiné à contenir un nombre aléatoire généré par le client. Si le même nombre est inclus dans la réponse chiffrée provenant du KDC, cela montre à l'évidence que la réponse est fraîche et n'a pas été répétée par un agresseur. Les noms occasionnels NE DOIVENT JAMAIS être réutilisés.

etype
Ce champ spécifie l'algorithme de chiffrement désiré à utiliser dans la réponse.

addresses
Ce champ est inclus dans la demande initiale de tickets, et il est facultativement inclus dans les demandes de tickets supplémentaires provenant du serveur d'allocation de tickets. Il spécifie les adresses à partir desquelles le ticket demandé doit être valide. Normalement, il inclut les adresses de l'hôte du client. Si un mandataire est demandé, ce champ contiendra d'autres adresses. Le contenu de ce champ est habituellement copié par le KDC dans le champ caddr du ticket résultant.

additional-tickets
Des tickets supplémentaires PEUVENT être facultativement inclus dans une demande au serveur d'allocation de

tickets. Si l'option ENC-TKT-IN-SKEY a été spécifiée, la clé de session provenant du ticket supplémentaire sera alors utilisée à la place de la clé du serveur pour chiffrer le nouveau ticket. Lorsque l'option ENC-TKT-IN-SKEY est utilisée pour l'authentification d'usager à usager, ce ticket supplémentaire PEUT être un TGT produit par le domaine local ou un TGT inter-domaine produit pour le domaine du KDC en cours par un KDC distant. Si plus d'une option exigeant des tickets supplémentaires a été spécifiée, les tickets supplémentaires sont alors utilisés dans l'ordre spécifié par l'ordre des bits des options bits (voir ci-dessus à kdc-options).

Le nombre d'étiquettes d'application sera dix (10) ou douze (12) selon que la demande est pour un ticket initial (AS-REQ) ou pour un ticket supplémentaire (TGS-REQ).

Les champs facultatifs (addresses, authorization-data, et additional-tickets) ne sont inclus que si nécessaires pour effectuer l'opération spécifiée dans le champ kdc-options.

Noter que dans KRB_TGS_REQ, le numéro de version du protocole apparaît deux fois et deux types de messages différents apparaissent : le message KRB_TGS_REQ contient ces champs tout comme le fait l'en-tête d'authentification (KRB_AP_REQ) qui est passé dans le champ padata.

5.4.2 Définition de KRB_KDC_REP

Le format de message KRB_KDC_REP est utilisé pour la réponse du KDC à une demande initiale (AS) ou à une demande ultérieure (TGS). Il n'y a pas de type de message pour KRB_KDC_REP. Le type sera KRB_AS_REP ou KRB_TGS_REP. La clé utilisée pour chiffrer la partie de texte chiffrée de la réponse dépend du type de message. Pour KRB_AS_REP, le texte crypté est chiffré avec la clé secrète du client, et le numéro de version de clé du client est inclus dans le numéro de version de clé pour les données chiffrées. Pour KRB_TGS_REP, le ciphertext est chiffré avec la sous-clé de session provenant de l'authentifiant ; si il est absent, le ciphertext est chiffré avec la clé de session provenant due TGT utilisé dans la demande. Dans ce cas, aucun numéro de version ne sera présent dans la séquence EncryptedData.

Le message KRB_KDC_REP contient les champs suivants :

```
AS-REP          ::= [APPLICATION 11] KDC-REP

TGS-REP         ::= [APPLICATION 13] KDC-REP

KDC-REP         ::= SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (11 -- AS -- | 13 -- TGS --),
        padata          [2] SEQUENCE OF PA-DATA OPTIONAL        -- NOTE : non vide --,
        crealm          [3] Realm,
        cname           [4] PrincipalName,
        ticket          [5] Ticket,
        enc-part        [6] EncryptedData               -- EncASRepPart ou EncTGSRepPart, selon le cas
  }

EncASRepPart    ::= [APPLICATION 25] EncKDCRepPart

EncTGSRepPar     ::= [APPLICATION 26] EncKDCRepPart

EncKDCRepPart   ::= SEQUENCE {
        key             [0] EncryptionKey,
        last-req        [1] LastReq,
        nonce           [2] UInt32,
        key-expiration  [3] KerberosTime OPTIONAL,
        flags           [4] TicketFlags,
        authtime        [5] KerberosTime,
        starttime       [6] KerberosTime OPTIONAL,
        endtime         [7] KerberosTime,
        renew-till      [8] KerberosTime OPTIONAL,
        srealm          [9] Realm,
        sname           [10] PrincipalName,
```

```
        caddr           [11] HostAddresses OPTIONAL
    }

LastReq                 ::=     SEQUENCE OF SEQUENCE {
        lr-type         [0] Int32,
        lr-value        [1] KerberosTime
    }
```

pvno et msg-type
Ces champs sont décrits ci-dessus au paragraphe 5.4.1. msg-type est KRB_AS_REP ou KRB_TGS_REP.

padata
Ce champ est décrit en détails au paragraphe 5.4.1. Une utilisation possible est de coder une chaîne "salée" de à utiliser avec un algorithme de chaîne à clé. Cette capacité est utile pour faciliter les transitions si un nom de domaine doit changer (par exemple, lors du rachat d'une entreprise) ; dans un tel cas, toutes les entrées déduites de mots de passe existantes dans la base de données du KDC auraient en fanion marquant qu'elles ont besoin d'une chaîne salée spéciale jusqu'au prochain changement de mot de passe.

crealm, cname, srealm, et sname
Ces champs sont les mêmes que ceux décrits pour le ticket au paragraphe 5.3.

ticket
C'st le ticket nouvellement produit du paragraphe 5.3.

enc-part
Ce champ est un fourre-tout pour le texte chiffré et les informations qui s'y rapportent, qui forme la partie chiffrée d'un message. La description de la partie chiffrée du message suit chaque apparition de ce champ.

La valeur d'usage de clé pour le chiffrement de ce champ est 3 dans un message AS-REP, en utilisant la clé à long terme du client ou une autre clé choisie via des mécanismes de pré-authentification. Dans un message TGS-REP, la valeur d'usage de clé est 8 si la clé de session TGS est utilisée, ou 9 si une sous-clé d'authentifiant TGS est utilisée.

Note pour la compatibilité : Certaines mises en œuvre envoient inconditionnellement une EncTGSRepPart chiffrée (numéro d'étiquette d'application 26) dans ce champ sans considérer si la réponse est une AS-REP ou une TGS-REP. Dans l'intérêt de la compatibilité, les développeurs PEUVENT assouplir la vérification du numéro d'étiquette de la ENC-PART déchiffrée.

key
Ce champ est le même que celui décrit pour le ticket au paragraphe 5.3.

last-req
Ce champ est retourné par le KDC et spécifie l'heure de la dernière demande d'un principal. Selon les informations disponibles, ce peut être la dernière fois qu'une demande de TGT a été faite, ou la dernière fois qu'une demande fondée sur un TGT a réussi. Il peut aussi couvrir tous les serveurs d'un domaine, ou juste un serveur particulier. Certaines mises en œuvre PEUVENT afficher ces informations à l'usager pour aider à découvrir des utilisations non autorisées d'une identité. C'est d'un esprit similaire à l'affichage de la dernière connexion affichée à la connexion dans les systèmes en temps partagé.

lr-type
Ce champ indique comment interpréter le champ lr-value suivant. Les valeurs négatives indiquent que ces informations n'appartiennent qu'au serveur qui répond. Les valeurs non négatives appartiennent à tous les serveurs pour le domaine.

Si le champ lr-type est zéro (0), aucune information n'est alors portée par le sous-champ lr-value. Si la valeur absolue du champ lr-type est un (1), le sous-champ lr-value est alors l'heure de la dernière demande initiale pour un TGT. Si elle est deux (2), le sous-champ lr-value est alors l'heure de la dernière demande initiale. Si c'est trois (3), le sous-champ lr-value est alors l'heure de la production du plus nouveau TGT utilisé. Si c'est quatre (4), le sous-champ lr-value est alors l'heure du dernier renouvellement. Si c'est cinq (5), le sous-champ lr-value est alors l'heure de la dernière demande (de tout type). Si c'est six (6), le sous-champ lr-value est alors l'heure de l'arrivée à expiration du mot de passe. Si c'est sept (7), le sous-champ lr-value est alors l'heure de l'arrivée à expiration du compte.

lr-value
Ce champ contient l'heure de la dernière demande. L'heure DOIT être interprétée conformément au contenu du sous-champ lr-type qui l'accompagne.

nonce
Ce champ est décrit ci-dessus au paragraphe 5.4.1.

key-expiration
Le champ key-expiration fait partie de la réponse du KDC et spécifie l'heure à laquelle la clé secrète du client va arriver à expiration. L'expiration peut être le résultat du vieillissement d'un mot de passe ou de l'expiration d'un compte. Si il est présent, il DEVRAIT être réglé au plus tôt de l'expiration de la clé de l'utilisateur et de l'expiration du compte. L'utilisation de ce champ est déconseillée, et le champ last-req DEVRAIT être utilisé à la place pour porter ces informations. Ce champ sera normalement laissé en dehors de la réponse TGS car la réponse à une demande de TGS est chiffrée avec une clé de session et aucune information client n'a à être restituée de la base de données du KDC. Il appartient au client d'application (habituellement le programme de connexion) de prendre les mesures appropriées (comme la notification à l'usager) si l'heure d'expiration est imminente.

flags, authtime, starttime, endtime, renew-till et caddr
Ces champs sont des duplications de ceux qui figurent dans la portion chiffré du ticket joint (voir au paragraphe 5.3), et ils sont fournis pour que le client PUISSE vérifier qu'ils correspondent à la demande prévue et afin d'aider à une mise en mémoire cache appropriée du ticket. Si le message est du type KRB_TGS_REP, le champ caddr ne sera rempli que si la demande était pour un mandataire ou un ticket transmis, ou si l'usager substitue un sous-ensemble des adresses venant du TGT. Si les adresses demandées par le client ne sont pas présentes ou pas utilisées, les adresses contenues dans le ticket devront alors être les mêmes que celles incluses dans le TGT.

5.5 Spécifications de message client/serveur (CS)

Ce paragraphe spécifie le format des messages utilisés pour l'authentification du client auprès du serveur d'application.

5.5.1 Définition de KRB_AP_REQ

Le message KRB_AP_REQ contient le numéro de version du protocole Kerberos, le type de message KRB_AP_REQ, un champ d'options pour indiquer toutes les options utilisées, et le ticket et l'authentifiant eux-mêmes. Le message KRB_AP_REQ est souvent désigné comme "en-tête d'authentification".

```
AP-REQ          ::= [APPLICATION 14] SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (14),
        ap-options      [2] APOptions,
        ticket          [3] Ticket,
        authenticator   [4] EncryptedData -- Authentifiant
  }

APOptions       ::= KerberosFlags
                    -- reserved(0),
                    -- use-session-key(1),
                    -- mutual-required(2)
```

pvno et msg-type
Ces champs sont décrits ci-dessus au paragraphe 5.4.1. Le type de message msg-type est KRB_AP_REQ.

ap-options
Ce champ apparaît dans la demande d'application (KRB_AP_REQ) et affecte la façon dont la demande est traitée. C'est un champ binaire, où les options choisies sont indiquées par le bit mis à un (1), et les options non choisies et les champs réservés sont mis à zéro (0). Le codage des bits est spécifié au paragraphe 5.2. La signification des options est la suivante :

Bit(s)	Nom	Description
0	réservé	Réservé pour l'expansion future de ce champ.
1	use-session-key	L'option USE-SESSION-KEY indique que le ticket que présente le client à un serveur est chiffré avec la clé de session venant du TGT du serveur. Quand cette option n'est pas spécifiée, le ticket est chiffré avec la clé secrète du serveur.
2	mutual-required	L'option MUTUAL-REQUIRED dit au serveur que le client exige l'authentification mutuelle, et qu'il doit répondre par un message KRB_AP_REP.
3-31	réservé	Réservé pour utilisation future.

ticket
Ce champ est un ticket authentifiant le client auprès du serveur.

authenticator
Ceci contient l'authentifiant chiffré, qui inclut le choix d'une sous-clé par le client.

L'authentifiant chiffré est inclus dans la AP-REQ ; il certifie à un serveur que l'envoyeur a une connaissance récente de la clé de chiffrement du ticket d'accompagnement, pour aider le serveur à détecter les répétitions. Il aide aussi au choix d'une "vraie clé de session" à utiliser dans cette session. Le codage DER de ce qui suit est chiffré avec la clé de session du ticket, avec une valeur d'usage de clé de 11 dans les échanges d'application normaux, ou 7 lorsque utilisée comme champ PA-TGS-REQ PA-DATA d'un échange TGS-REQ (voir au paragraphe 5.4.1):

```
-- Authentifiant non chiffré
Authenticator               ::= [APPLICATION 2] SEQUENCE  {
        authenticator-vno   [0] INTEGER (5),
        crealm              [1] Realm,
        cname               [2] PrincipalName,
        cksum               [3] Checksum OPTIONAL,
        cusec               [4] Microseconds,
        ctime               [5] KerberosTime,
        subkey              [6] EncryptionKey OPTIONAL,
        seq-number          [7] UInt32 OPTIONAL,
        authorization-data  [8] AuthorizationData OPTIONAL
  }
```

authenticator-vno
Ce champ spécifie le numéro de version du format de l'authentifiant. Le présent document spécifie la version 5.

crealm et cname
Ces champs sont les mêmes que ceux décrits pour le ticket au paragraphe 5.3.

cksum
Ce champ contient une somme de contrôle des données d'application qui accompagnent la KRB_AP_REQ, calculée en utilisant une valeur d'usage de clé de 10 dans les échanges d'application normaux, ou 6 lorsque utilisée dans le champ TGS-REQ PA-TGS-REQ AP-DATA.

cusec
Ce champ contient la partie microseconde de l'horodatage du client. Sa valeur (avant chiffrement) va de 0 à 999999. Il apparaît souvent avec ctime. Les deux champs sont utilisés ensemble pour spécifier un horodatage raisonnablement précis.

ctime
Ce champ contient l'heure en cours sur l'hôte du client.

subkey
Ce champ contient le choix du client d'une clé de chiffrement à utiliser pour protéger cette session d'application spécifique. Sauf si une application spécifie autre chose, si ce champ est laissé de côté, la clé de session venant du ticket sera utilisée.

seq-number
Ce champ facultatif comporte le numéro de séquence initiale à utiliser par les messages KRB_PRIV ou KRB_SAFE

lorsque les numéros de séquence sont utilisés pour détecter les répétitions. (Il peut aussi être utilisé par des messages spécifiques de l'application.) Lorsqu'il est inclus dans l'authentifiant, ce champ spécifie le numéro de séquence initiale pour les messages du client au serveur. Lorsqu'il est inclus dans le message AP-REP, le numéro de séquence initiale est celui des messages du serveur au client. Lorsqu'il est utilisé dans les messages KRB_PRIV ou KRB_SAFE, il est incrémenté de un après l'envoi de chaque message. Les numéros de séquence sont dans la gamme de 0 à 2^32 - 1 et reviennent à zéro après la valeur 2^32 - 1.

Pour que les numéros de séquence prennent adéquatement en charge la détection des répétitions, ils DEVRAIENT être non répétitifs, même à travers les frontières de connexion. Le numéro de séquence initiale DEVRAIT être aléatoire et uniformément distribué à travers l'espace complet des numéros de séquence possibles, de sorte qu'il ne puisse pas être deviné par un agresseur et de sorte que lui et les numéros de séquence successifs ne répètent pas d'autres séquences. Au cas où plus de 2^32 messages devraient être générés dans une série de messages KRB_PRIV ou KRB_SAFE, un changement de clé DEVRAIT être effectué avant que les numéros de séquence soient réutilisés avec la même clé de chiffrement.

Note de mise en œuvre : Historiquement, certaines mises en œuvre transmettaient des nombres algébriques de compléments deux comme numéros de séquence. Dans l'intérêt de la compatibilité, les mises en œuvre PEUVENT accepter le nombre négatif équivalent où elles attendent un nombre positif supérieur à 2^31 - 1.

Note de mise en œuvre : Comme noté auparavant, certaines mises en œuvre omettent le numéro de séquence facultatif lorsque sa valeur serait zéro. Les mises en œuvre PEUVENT accepter un numéro de séquence omis lorsqu'elles attendent une valeur de zéro, et NE DEVRAIENT PAS transmettre un authentifiant avec un numéro de séquence initiale de zéro.

authorization-data
Ce champ est le même que celui décrit pour le ticket au paragraphe 5.3. Il est facultatif et n'apparaît que lorsque des restrictions supplémentaires sont à mettre à l'utilisation d'un ticket, au-delà de celles portées par le ticket lui-même.

5.5.2 Définition de KRB_AP_REP

Le message KRB_AP_REP contient le numéro de version du protocole Kerberos, le type de message, et un horodatage chiffré. Le message est envoyé en réponse à une demande d'application (KRB_AP_REQ) pour laquelle l'option d'authentification mutuelle a été choisie dans le champ ap-options.

```
AP-REP          ::= [APPLICATION 15] SEQUENCE {
        vno         [0] INTEGER (5),
        sg-type     [1] INTEGER (15),
        nc-part      [2] EncryptedData -- EncAPRepPart
  }

EncAPRepPart    ::= [APPLICATION 27] SEQUENCE {
        time        [0] KerberosTime,
        usec        [1] Microseconds,
        ubkey       [2] EncryptionKey OPTIONAL,
        eq-number   [3] UInt32 OPTIONAL
  }
```

La partie codée EncAPRepPart est chiffrée avec la clé de session partagée du ticket. Le champ d sous-clé facultatif peut être utilisé dans une négociation arrangée par l'application pour choisir une clé de session par association.

pvno et msg-type
Ces champs sont décrits ci-dessus au paragraphe 5.4.1. Le type de message msg-type est KRB_AP_REP.

enc-part
Ce champ est décrit ci-dessus au paragraphe 5.4.2. Il est calculé avec une valeur d'usage de clé de 12.

ctime
Ce champ contient l'heure en cours sur l'hôte du client.

cusec

Ce champ contient la partie microseconde de l'horodatage du client.

subkey
Ce champ contient une clé de chiffrement qui est à utiliser pour protéger cette session d'application spécifique. Voir au paragraphe 3.2.6 les spécificités de la façon d'utiliser ce champ pour négocier une clé. Sauf spécification contraire par une application, si ce champ est laissé de côté, la sous-clé de session venant de l'authentifiant ou si ce dernier est aussi laissé de côté, la clé de session venant du ticket sera utilisée.

seq-number
Ce champ est décrit ci-dessus au paragraphe 5.3.2.

5.5.3 Réponse à message d'erreur

Si une erreur survient pendant le traitement de la demande d'application, le message KRB_ERROR sera envoyé en réponse. Voir au paragraphe 5.9.1 le format du message d'erreur. Les champs cname et crealm PEUVENT être laissés de côté si le serveur ne peut pas déterminer leurs valeurs appropriées d'après le message KRB_AP_REQ correspondant. Si l'authentifiant était déchiffrable, les champs ctime et cusec contiendront ses valeurs.

5.6 Spécification du message KRB_SAFE

Ce paragraphe spécifie le format d'un message qui peut être utilisé par l'un ou l'autre côté (client ou serveur) d'une application pour envoyer un message inaltérable à son homologue. Il suppose qu'une clé de session ait été échangée précédemment (par exemple, en utilisant les messages KRB_AP_REQ/KRB_AP_REP).

5.6.1 Définition de KRB_SAFE

Le message KRB_SAFE contient des données d'usager avec une somme de contrôle à l'épreuve des collisions chiffrée avec la dernière clé de chiffrement négociée via des sous-clés, ou avec la clé de session si aucune négociation n'est intervenue. Les champs de message sont les suivants :

```
KRB-SAFE          ::= [APPLICATION 20] SEQUENCE {
        pvno          [0] INTEGER (5),
        msg-type      [1] INTEGER (20),
        safe-body     [2] KRB-SAFE-BODY,
        cksum         [3] Checksum
  }

KRB-SAFE-BODY     ::= SEQUENCE {
        user-data     [0] OCTET STRING,
        horodatage    [1] KerberosTime OPTIONAL,
        usec          [2] Microseconds OPTIONAL,
        seq-number    [3] UInt32 OPTIONAL,
        s-address     [4] HostAddress,
        r-address     [5] HostAddress OPTIONAL
  }
```

pvno et msg-type
Ces champs sont décrits ci-dessus au paragraphe v5.4.1. le type de message msg-type est KRB_SAFE.

safe-body
Ce champ est un fourre tout pour le corps du message KRB-SAFE.

cksum
Ce champ contient la somme de contrôle des données d'application, calculée avec une valeur d'usage de clé de 15.
La somme de contrôle est calculée sur le codage de la séquence KRB-SAFE. D'abord, le cksum est mis à un type zéro, valeur de longueur zéro, et la somme de contrôle est calculée sur le codage de la séquence KRB-SAFE. Puis la somme de contrôle est réglée au résultat de ce calcul. Finalement, la séquence KRB-SAFE est codée à nouveau. Cette méthode, bien que différente de celle spécifiée dans la RFC 1510, correspond à la pratique existante.

user-data
Ce champ fait partie des messages KRB_SAFE et KRB_PRIV, et contient les données spécifiques de l'application qui sont passées de l'envoyeur au receveur.

horodatage
Ce champ fait partie des messages KRB_SAFE et KRB_PRIV. Son contenu est l'heure en cours telle que la connaît l'envoyeur du message. En vérifiant l'horodatage, le receveur du message est capable de s'assurer qu'il a été généré récemment, et n'est pas une répétition.

usec
Ce champ fait partie des en-têtes KRB_SAFE et KRB_PRIV. Il contient la partie microsecondes de l'horodatage.

seq-number
Ce champ est décrit ci-dessus au paragraphe 5.3.2.

s-address (adresse de l'envoyeur)
Ce champ spécifie l'adresse utilisée par l'envoyeur du message.

r-address
Ce champ spécifie l'adresse utilisée par le receveur du message. Il PEUT être omis pour certaines utilisations (comme les protocoles de diffusion), mais le receveur PEUT arbitrairement rejeter de tels messages. Ce champ, avec s-address, peut être utilisé pour aider à détecter des messages qui ont été incorrectement ou malicieusement délivrés à un mauvais receveur.

5.7 Spécification du message KRB_PRIV

Ce paragraphe spécifie le format d'un message qui peut être utilisé par l'un et l'autre côté (client ou serveur) d'une application pour envoyer en toute sécurité et confidentialité un message à son homologue. Il suppose qu'une clé de session ait précédemment été échangée (par exemple, en utilisant les messages KRB_AP_REQ/KRB_AP_REP).

5.7.1 Définition de KRB_PRIV

Le message KRB_PRIV contient des données d'utilisateur chiffrées avec la clé de session. Les champs de message sont les suivants :

```
KRB-PRIV        ::= [APPLICATION 21] SEQUENCE {
        pvno        [0] INTEGER (5),
        msg-type    [1] INTEGER (21),           -- NOTE : Il n'y a pas d'étiquette [2]
        enc-part    [3] EncryptedData           -- EncKrbPrivPart
  }

EncKrbPrivPart  ::= [APPLICATION 28] SEQUENCE {
        user-data   [0] OCTET STRING,
        horodatage  [1] KerberosTime OPTIONAL,
        usec        [2] Microseconds OPTIONAL,
        seq-number  [3] UInt32 OPTIONAL,
        s-address   [4] HostAddress             -- adresse de l'envoyeur
        r-address   [5] HostAddress OPTIONAL    -- adresse du receveur
  }
```

pvno et msg-type
Ces champs sont décrits ci-dessus au paragraphe 5.4.1. le type de message msg-type est KRB_PRIV.

enc-part
Ce champ détient un codage de la séquence EncKrbPrivPart chiffrée avec la clé de session, avec une valeur d'usage de clé de 13. Ce codage chiffré est utilisé pour le champ enc-part du message KRB-PRIV.

user-data, horodatage, usec, s-address, et r-address
Ces champs sont décrits ci-dessus au paragraphe 5.6.1.

seq-number
Ce champ est décrit ci-dessus au paragraphe 5.3.2.

5.8 Spécification du message KRB_CRED

Ce paragraphe spécifie le format d'un message qui peut être utilisé pour envoyer des accréditifs Kerberos d'un principal à un autre. Il est présenté ici pour encourager l'utilisation d'un mécanisme commun par les applications lors de la transmission des tickets ou la fourniture de mandataires aux serveurs subordonnés. Il suppose qu'une clé de session a déjà été échangée, peut-être en utilisant les messages KRB_AP_REQ/KRB_AP_REP.

5.8.1 Définition de KRB_CRED

Le message KRB_CRED contient une séquence de tickets à envoyer et les informations nécessaires pour utiliser les tickets, comportant la clé de session de chacun. Les informations nécessaires pour utiliser les tickets sont chiffrées avec une clé de chiffrement échangée précédemment ou transférée à l'aide du message KRB_CRED. Les champs de message sont les suivants :

```
KRB-CRED        ::= [APPLICATION 22] SEQUENCE {
        pvno        [0] INTEGER (5),
        msg-type    [1] INTEGER (22),
        tickets     [2] SEQUENCE OF Ticket,
        enc-part    [3] EncryptedData -- EncKrbCredPart
  }

EncKrbCredPart  ::= [APPLICATION 29] SEQUENCE {
        ticket-info [0] SEQUENCE OF KrbCredInfo,
        nonce       [1] UInt32 OPTIONAL,
        horodatage  [2] KerberosTime OPTIONAL,
        usec        [3] Microseconds OPTIONAL,
        s-address   [4] HostAddress OPTIONAL,
        r-address   [5] HostAddress OPTIONAL
  }

KrbCredInfo      ::= SEQUENCE {
        key         [0] EncryptionKey,
        prealm      [1] Realm OPTIONAL,
        pname       [2] PrincipalName OPTIONAL,
        flags       [3] TicketFlags OPTIONAL,
        authtime    [4] KerberosTime OPTIONAL,
        starttime   [5] KerberosTime OPTIONAL,
        endtime     [6] KerberosTime OPTIONAL,
        renew-till  [7] KerberosTime OPTIONAL,
        srealm      [8] Realm OPTIONAL,
        sname       [9] PrincipalName OPTIONAL,
        caddr       [10] HostAddresses OPTIONAL
  }
```

pvno et msg-type
Ces champs sont décrits ci-dessus au paragraphe 5.4.1. Le type de message msg-type est KRB_CRED.

tickets
Ce sont les tickets obtenus du KDC pour utilisation spécifique par le receveur prévu. Les tickets successifs sont appariés avec la séquence KrbCredInfo correspondante de la enc-part du message KRB- CRED.

enc-part
Ce champ détient un codage de la séquence EncKrbCredPart chiffrée avec la clé de session partagée par l'envoyeur et

le receveur prévu, avec une valeur d'usage de clé de 14. Ce codage chiffré est utilisé pour le champ enc-part du message KRB-CRED.

Note de mise en œuvre : Les mises en œuvre de certaines applications, et plus particulièrement de certaines mises en œuvre du mécanisme GSS-API de Kerberos, ne chiffrent pas séparément le contenu de la partie EncKrbCredPart du message KRB-CRED lorsqu'elle l'envoient. Dans le cas de ces mécanismes GSS-API, ce n'est pas une faiblesse pour la sécurité, car tout le message KRB-CRED est lui-même incorporé dans un message chiffré.

nonce
Si elle en a la capacité, une application PEUNT exiger l'inclusion d'un nom occasionnel généré par le receveur du message. Si la même valeur est incluse comme nom occasionnel dans le message, cela donne la preuve que le message est frais et n'a pas été répété par un agresseur. Un nom occasionnel NE DOIT PAS être réutilisé.

horodatage et usec
Ces champs spécifient l'heure à laquelle le message KRB-CRED a été généré. L'heure est utilisée pour fournir l'assurance que le message est frais.

s-address et r-address
Ces champs sont décrits ci-dessus au paragraphe 5.6.1. Ils sont utilisés facultativement pour fournir une assurance supplémentaire de l'intégrité du message KRB-CRED.

key
Ce champ existe dans le ticket correspondant passé par le message KRB-CRED et est utilisé pour passer la clé de session de l'envoyeur au destinataire prévu. Le codage du champ est décrit au paragraphe 5.2.9.

Les champs suivants sont facultatifs. S'ils sont présents, ils peuvent être associés aux accréditifs dans le fichier de ticket distant. S'ils sont laissés de côté, il est alors supposé que le receveur des accréditifs connaît déjà leurs valeurs.

prealm et pname
Nom et domaine de l'entité de principal délégué.

flags, authtime, starttime, endtime, renew-till, srealm, sname, et caddr
Ces champs contiennent les valeurs des champs correspondants dans le ticket trouvé dans le champ ticket. Les descriptions des champs sont identiques aux descriptions dans le message KDC-REP.

5.9 Spécification de message d'erreur

Ce paragraphe spécifié le format du message KRB_ERROR. Les champs inclus dans le message sont destinés à retourner autant d'informations que possible sur l'erreur. On ne s'attend pas à ce que toutes les informations exigées par les champs soient disponibles pour tous les types d'erreur. Si les informations appropriées ne sont pas disponibles lors de la composition du message, le champ correspondant sera laissé de côté pour ce message.

Noter que comme le message KRB_ERROR n'est pas protégé en intégrité, il est assez possible à un agresseur de le synthétiser ou modifier. En particulier, cela signifie que le client NE DEVRAIT PAS utiliser ces champs dans ce message pour des besoins critiques pour la sécurité, comme le réglage d'une horloge système ou la génération d'un authentifiant frais. Le message peut cependant être utile pour informer un usager des raisons d'un échec.

5.9.1 Définition de KRB_ERROR

Le message KRB_ERROR comporte les champs suivants :

```
KRB-ERROR       ::= [APPLICATION 30] SEQUENCE {
        pvno        [0] INTEGER (5),
        msg-type    [1] INTEGER (30),
        ctime       [2] KerberosTime OPTIONAL,
        cusec       [3] Microseconds OPTIONAL,
        stime       [4] KerberosTime,
        susec       [5] Microseconds,
```

```
        error-code      [6] Int32,
        crealm          [7] Realm OPTIONAL,
        cname           [8] PrincipalName OPTIONAL,
        realm           [9] Realm                       -- domaine du service --,
        sname           [10] PrincipalName              -- domaine du service --,
        e-text          [11] KerberosString OPTIONAL,
        e-data          [12] OCTET STRING OPTIONAL
    }
```

pvno et msg-type
Ces champs sont décrits ci-dessus au paragraphe 5.4.1. Le type de message msg-type est KRB_ERROR.

ctime et cusec
Ces champs sont décrits ci-dessus au paragraphe 5.5.2. Si les valeurs de ces champs sont connues de l'entité qui génère l'erreur (comme se sera le cas si KRB-ERROR est généré en réponse à, par exemple, un échec de demande de service d'authentification), ils devraient être remplis dans la KRB-ERROR. Si les valeurs ne sont pas disponibles, ces champs peuvent être omis.

stime
Ce champ contient l'heure en cours au serveur. Il est du type KerberosTime.

susec
Ce champ contient la partie microseconde de l'horodatage du serveur. Sa gamme de valeurs va de 0 à 999999. Il apparaît avec stime. Les deux champs sont utilisés en conjonction pour spécifier un horodatage raisonnablement précis.

error-code
Ce champ contient le code d'erreur retourné par Kerberos ou le serveur lors de l'échec d'une demande. Pour interpréter la valeur de ce champ, voir la liste des codes d'erreur au paragraphe 7.5.9. Les mises en œuvre sont invitées à fournir un support en langage national pour l'affichage des messages d'erreur.

crealm, et cname
Ces champs sont décrits ci-dessus au paragraphe 5.3. Lorsque l'entité qui génère l'erreur connaît ces valeurs, elles devraient être remplies dans le KRB-ERROR. Si les valeurs ne sont pas connues, les champs crealm et cname DEVRAIENT être omis.

realm et sname
Ces champs sont décrits ci-dessus au paragraphe 5.3.

e-text
Ce champ contient du texte supplémentaire pour aider à expliquer le code d'erreur associé à l'échec de demande (par exemple, il peut inclure un nom de principal qui est inconnu).

e-data
Ce champ contient des données supplémentaires sur l'erreur, à utiliser par l'application, pour l'aider à récupérer de l'erreur ou à la traiter. Si le code d'erreur est KDC_ERR_PREAUTH_REQUIRED, le champ e-data contiendra alors un codage de la séquence des champs padata, chacun correspondant à une méthode acceptable de pré-authentification et contenant facultativement des données pour la méthode :

```
    METHOD-DATA         ::= SEQUENCE OF PA-DATA
```

Pour les codes d'erreur définis dans le présent document autres que KDC_ERR_PREAUTH_REQUIRED, le format et le contenu du champ e-data sont définis par la mise en œuvre. De même, pour les codes d'erreur futurs, le format et le contenu du champ e-data sera défini par la mise en œuvre sauf spécification contraire. Qu'ils soient définis par la mise en œuvre ou dans un document futur, le champ e-data PEUT prendre la forme de TYPED-DATA :

```
TYPED-DATA          ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
        data-type       [0] Int32,
        data-value      [1] OCTET STRING OPTIONAL
    }
```

5.10 Numéros d'étiquette d'application

Le tableau suivant fait la liste des numéros d'étiquette de classe d'application utilisés par divers types de données définies dans ce paragraphe.

Numéros d'étiquette	Nom de type	Commentaires
0		non utilisé
1	Ticket	PDU
2	Authenticator	non PDU
3	EncTicketPart	non PDU
4-9		non utilisé
10	AS-REQ	PDU
11	AS-REP	PDU
12	TGS-REQ	PDU
13	TGS-REP	PDU
14	AP-REQ	PDU
15	AP-REP	PDU
16	RESERVED16	TGT-REQ (d'utilisateur à utilisateur)
17	RESERVED17	TGT-REQ (d'utilisateur à utilisateur)
18-19		non utilisé
20	KRB-SAFE	PDU
21	KRB-PRIV	PDU
22	KRB-CRED	PDU
23-24		non utilisé
25	EncASRepPart	non PDU
26	EncTGSRepPart	non PDU
27	EncApRepPart	non PDU
28	EncKrbPrivPart	non PDU
29	EncKrbCredPart	non PDU
30	KRB-ERROR	PDU

Les types ASN.1 marqués ci-dessus comme "PDU" (unités de données de protocole) sont les seuls types ASN.1 perçus comme types de niveau supérieur dans le protocole Kerberos, et sont les seuls types qui peuvent être utilisés comme éléments dans un autre protocole qui utilise Kerberos.

6 Contraintes de dénomination

6.1 Noms de domaine

Bien que les noms de domaine soient codés comme des GeneralStrings et que techniquement un domaine puisse choisir le nom qu'il veut, l'interopérabilité à travers les frontières de domaine exige un accord sur la façon dont les noms de domaine sont alloués et les informations qu'ils comportent.

Pour mettre en application ces conventions, chaque domaine DOIT se conformer aux conventions elles-mêmes, et il DOIT exiger que tout domaine avec lequel il partage des clés inter domaines se conforme aussi aux conventions et qu'il exige la même chose de ses voisins.

Les noms de domaine Kerberos sont sensibles à la casse. Les noms de domaine qui ne diffèrent que par la casse des caractères ne sont pas équivalents. Il y a actuellement trois styles de noms de domaine : domaine, X500, et les autres. Ci-après figurent des exemples de chaque style :

domaine : ATHENA.MIT.EDU
X500 : C=US/O=OSF
autre : NAMETYPE:rest/of.name=sans-restrictions

Les noms de domaine du style domaine DOIVENT ressembler à des noms de domaine : ils comportent des composants séparés par des points (.) et ils ne contiennent ni deux point (:) ni barres obliques (/). Bien que les noms de domaine eux-mêmes ne soient pas sensibles à la casse, afin que les domaines correspondent, la casse doit aussi correspondre. Lors de l'établissement d'un nouveau nom de domaine fondé sur un nom de domaine internet, il est recommandé par

convention que les caractères soient convertis en majuscules.

Les noms X.500 contiennent le signe égal (=) et ne peuvent pas contenir deux points (:) avant le signe égal. Les noms de domaine pour les noms X.500 doivent être des représentations de chaîne des noms avec des composants séparés par des barres obliques. Les barres obliques en tête et en queue ne seront pas incluses. Noter que la barre oblique de séparation est cohérente avec les mises en œuvre de Kerberos fondées sur la RFC 1510, mais elle est différente du séparateur recommandé dans la RFC 2253.

Les noms qui entrent dans la catégorie autre DOIVENT commencer par un préfixe ne contenant pas de signe égal (=) ou point (.), et le préfixe DOIT être suivi par deux points (:) et le reste du nom. Tous les préfixes attendent ceux qui commencent à être utilisés. Actuellement, aucun n'est alloué.

La catégorie réservée comporte des chaînes qui n'entrent pas dans les trois premières catégories. Tous les noms de cette catégorie sont réservés. Il est peut vraisemblable que des noms soient alloués dans cette catégorie sauf s'il y a de très forts arguments pour ne pas utiliser la catégorie 'autre'.

Ces règles garantissent qu'il n'y aura pas de conflit entre les divers styles de nom. Les contraintes supplémentaires suivantes s'appliquent à l'allocation de noms de domaine dans les catégories domaine et X.500 : le nom d'un domaine pour les formats domaine ou X.500 doit être utilisé par l'organisation propriétaire (à qui il a été alloué) d'un nom de domaine Internet ou d'un nom X.500, ou, dans le cas où un tel nom d'est pas enregistré, l'autorité pour utiliser un nom de domaine PEUT être dérivée de l'autorité du domaine parent. Par exemple, si il n'y a pas de nom de domaine pour E40.MIT.EDU, l'administrateur du domaine MIT.EDU peut alors autoriser la création d'un domaine de ce nom.

Ceci est acceptable parce que l'organisation à laquelle le parent est alloué est vraisemblablement aussi l'organisation autorisée à allouer des noms à ses enfants dans les systèmes de nom X.500 et domaine. Si le parent alloue un nom de domaine sans l'enregistrer aussi dans la hiérarchie de nom de domaine ou X.500, il est de la responsabilité du parent de s'assurer qu'à l'avenir il n'existera pas un nom identique au nom de domaine de l'enfant sauf si il est alloué à la même entité comme nom de domaine.

6.2 Noms de principal

Comme c'est le cas pour les noms de domaine, des conventions sont nécessaires pour s'assurer que tous sont d'accord sur les informations impliquées par un nom de principal. Le champ name-type qui fait partie du nom de principal indique le type d'informations impliquées par le nom. Le name-type DEVRAIT n'être traité que comme un conseil pour interpréter la signification d'un nom. Il n'y a pas de sens à lui rechercher une équivalence. Les noms de principal qui ne diffèrent que par le name-type identifient le même principal. Le type de nom ne crée pas une partition de l'espace de nom. En ignorant le type de nom, deux noms ne peuvent être les mêmes (c'est-à-dire, au moins un des composants, ou le domaine, DOIVENT être différents). Les types de nom suivants sont définis :

Type de nom	Valeur	Signification
NT-UNKNOWN	0	Type de nom inconnu
NT-PRINCIPAL	1	Seulement le nom du principal comme dans DCE, ou pour les utilisateurs
NT-SRV-INST	2	Service et autre instance unique (krbtgt)
NT-SRV-HST	3	Service avec nom d'hôte comme instance (telnet, rcommands)
NT-SRV-XHST	4	Service avec hôte comme composants restants
NT-UID	5	ID unique
NT-X500-PRINCIPAL	6	Nom distinctif codé en X.509 [RFC2253]
NT-SMTP-NAME	7	Nom en forme de nom de messagerie électronique SMTP (par exemple, user@example.com)
NT-ENTERPRISE	10	Nom d'entreprise - peut être transposé en nom de principal

Lorsque un nom n'implique pas d'informations autres que son unicité à un moment particulier, le type de nom PRINCIPAL DEVRAIT être utilisé. Le type de nom principal DEVRAIT être utilisé pour les utilisateurs, et il peut aussi être utilisé pour un serveur unique. Si le nom est un ID unique généré par la machine qu'il est garanti ne n'être jamais réalloué, le type de nom d'UID DEVRAIT alors être utilisé. (Noter que c'est généralement une mauvaise idée de réallouer des noms de n'importe quel type car des entrées périmées peuvent rester dans des listes de contrôle d'accès.)

Si le premier composant d'un nom identifie un service et si des composants restants identifient une instance du service

d'une manière spécifiée par le serveur, le type de nom de SRV-INST DEVRAIT alors être utilisé. Un exemple de ce type de nom est le service d'allocation de tickets Kerberos dont le nom a un premier composant de krbtgt et un second composant identifiant le domaine pour lequel le ticket est valide.
Si le premier composant d'un nom identifie un service et qu'il y a un seul composant suivant le nom de service qui identifie l'instance comme l'hôte sur lequel fonctionne le serveur, le type de nom SRV-HST DEVRAIT alors être utilisé. Ce type est normalement utilisé pour les services Internet tels que telnet et les commandes R de Berkeley. Si les composants séparés du nom de l'hôte apparaissent comme des composants successifs qui suivent le nom du service, le type de nom SRV-XHST DEVRAIT alors être utilisé. Ce type peut être utilisé pour identifier les serveurs sur des hôtes qui ont des noms X.500, et où la barre oblique (/) pourrait autrement être ambiguë.

Un type de nom de NT-X500-PRINCIPAL DEVRAIT être utilisé lorsque un nom provenant d'un certificat X.509 est traduit en un nom Kerberos. Le codage du nom X.509 comme un principal de Kerberos doit être conforme aux règles de codage spécifiées dans la RFC 2253.

Un type de nom de SMTP permet à un nom d'être d'une forme qui ressemble à un nom de messagerie électronique SMTP. Ce nom, comportant un "@" et un nom de domaine, est utilisé comme premier composant du nom de principal.

Un type de nom de UNKNOWN DEVRAIT être utilisé lorsque la forme du nom n'est pas connue. Lors d'une comparaison des noms, un type de nom UNKNOWN correspondra aux principaux authentifiés avec des noms de tout type. Un principal authentifié avec un type de nom de UNKNOWN ne correspondra cependant qu'à d'autres noms de type UNKNOWN.

Les noms de tout type avec un composant initial de 'krbtgt' sont réservés pour le service d'allocation de tickets Kerberos. Voir au paragraphe 7.3 la forme de tels noms.

6.2.1 Nom des principaux de serveur

L'identifiant de principal pour un serveur sur un hôte sera généralement composé de deux parties : (1) le domaine du KDC avec lequel le serveur est enregistré, et (2) un nom à deux composants de type NT-SRV-HST, si le nom d'hôte est un nom de domaine Internet, ou un nom multi composants de type NT-SRV-XHST, si le nom de l'hôte est d'une forme (telle que X.500) qui permet la barre oblique (/) comme séparateurs. Le premier composant du nom à deux ou plusieurs composants va identifier le service, et les derniers composants vont identifier l'hôte. Lorsque le nom de l'hôte n'est pas sensible à la casse (par exemple, avec les noms de domaine Internet) le nom de l'hôte DOIT être en minuscules. Si c'est spécifié par le protocole d'application pour des services tels que telnet et les commandes R de Berkeley qui fonctionnent avec des privilèges de système, le premier composant PEUNT être ka chaîne 'host' au lieu d'un identifiant spécifique du service.

7 Constantes et autres valeurs définies

7.1 Types d'adresse d'hôte

Toutes les valeurs négatives pour le type d'adresse d'hôte sont réservées à une utilisation locale. Toutes les valeurs non négatives sont réservées aux champs et interprétations de type officiellement alloués.

Adresses Internet (IPv4)
Les adresses Internet (IPv4) sont des quantités de 32 bits (4 octets), codées dans l'ordre MSB (octet de plus fort poids d'abord). Une adresse IPv4 de bouclage NE DEVRAIT PAS apparaître dans une PDU Kerberos. Le type des adresses IPv4 est deux (2).
Adresses Internet (IPv6)
Les adresses IPv6 [RFC3513] sont des quantités de 128 bits (16 octets), codées dans l'ordre MSB (octet de plus fort poids d'abord). Le type des adresses IPv6 est vingt quatre (24). Les adresses suivantes NE DOIVENT PAS apparaître dans une PDU Kerberos :

* L'adresse non spécifiée
* L'adresse de bouclage
* Les adresses de liaison locale

Ces restrictions s'appliquent à l'inclusion dans les champs d'adresse des PDU de Kerberos, mais pas aux champs d'adresse des paquets qui pourraient porter de telles PDU. La restriction est nécessaire parce que l'utilisation d'une

adresse d'une portée non mondiale pourrait permettre l'acceptation d'un message envoyé à partir d'un noeud qui pourrait avoir la même adresse, mais qui ne serait pas l'hôte prévu par l'entité qui a ajouté la restriction. Si le type d'adresse liaison locale doit être utilisé pour une communication, la restriction d'adresse dans les tickets ne doit pas être utilisée (c'est-à-dire que des tickets sans adresse doivent être utilisés).

Les adresses IPv6 transposées en IPv4 DOIVENT être représentées comme adresses de type 2.

Adresses DECnet de Phase IV
Les adresses DECnet de Phase IV sont des adresses de 16 bits, codées dans l'ordre LSB. Le type des adresses DECnet de Phase IV est douze (12).
Adresses Netbios
Les adresses Netbios sont des adresses de 16 octets normalement composées de caractères alphanumériques de 1 à 15 et bourrées avec le caractère US-ASCII SPC (code 32). Le 16ème octet DOIT être le caractère US-ASCII NUL (code 0). Le type des adresses Netbios est vingt (20).

Adresses directionnelles
Inclure l'adresse de l'envoyeur dans les messages KRB_SAFE et KRB_PRIV n'est pas souhaitable dans de nombreux environnements parce que les adresses peuvent être changées dans le transport par des traducteurs d'adresse réseau. Cependant, si ces adresses sont retirées, les messages peuvent être soumis à une attaque par réflexion dans laquelle un message est répliqué en retour à son origine. Le type d'adresse directionnelle donne le moyen d'éviter les attaques des adresses dans le transport et en réflexion. Les adresses directionnelles sont codées comme des entiers non signés de quatre octets dans l'ordre des octets du réseau. Si le message est généré par la partie qui envoie le message KRB_AP_REQ original, une adresse de 0 DEVRAIT alors être utilisée. Si le message est généré par la partie à qui le message KRB_AP_REQ était envoyé, l'adresse 1 DEVRAIT alors être utilisée. Les applications qui impliquent plusieurs parties peuvent spécifier l'utilisation des autres adresses.

Les adresses directionnelles DOIVENT être utilisées uniquement pour le champ d'adresse d'envoyeur dans les messages KRB_SAFE ou KRB_PRIV. Elles NE DOIVENT PAS être utilisées comme adresse de ticket ou dans un message KRB_AP_REQ. Ce type d'adresse DEVRAIT être utilisé seulement dans des situations où la partie qui envoie sait que la partie qui reçoit prend en charge le type d'adresse. Cela signifie généralement que les adresses directionnelles ne peuvent être utilisées que quand le protocole d'application exige leur prise en charge. Les adresses directionnelles sont du type (3).

7.2 Échange de messages du KDC : Transports IP

Kerberos définit deux mécanismes de transport IP pour communiquer entre clients et serveurs : UDP/IP et TCP/IP.

7.2.1 Transport UDP/IP

Les serveurs Kerberos (KDC) qui prennent en charge les transports IP DOIVENT accepter les demandes UDP et DEVRAIENT les écouter sur l'accès 88 (décimal) sauf s'ils sont spécifiquement configurés pour écouter sur un autre accès UDP. Des accès de remplacement PEUVENT être utilisés quand plusieurs KDC fonctionnent sur plusieurs domaines sur le même hôte.

Les clients Kerberos qui prennent en charge les transports IP DEVRAIENT prendre en charge l'envoi des demandes UDP. Les clients DEVRAIENT utiliser la découverte de KDC [7.2.3] pour identifier l'adresse et le port IP auquel ils veulent envoyer leur demande.

Lorsque il contacte un KDC pour une demande KRB_KDC_REQ en utilisant le transport UDP/IP, le client doit envoyer un doit UDP ne contenant qu'un codage de la demande au KDC. L KDC répondra avec un doit de réponse ne contenant qu'un codage du message de réponse (soi un KRB_ERROR, soit un KRB_KDC_REP) à l'accès d'envoi à l'adresse IP de l'envoyeur. La réponse à une demande faite au moyen d'un transport UDP/IP DOIT aussi utiliser le transport UDP/IP. Si la réponse ne peut être traitée en utilisant UDP (par exemple, parce qu'elle est trop grande), le KDC DOIT retourner KRB_ERR_RESPONSE_TOO_BIG, forçant le client à réessayer la demande en utilisant le transport TCP.

7.2.2 Transport TCP/IP

Les serveurs Kerberos (KDC) qui prennent en charge les transports IP DOIVENT accepter les demandes TCP et DEVRAIENT les écouter sur l'accès 88 (décimal) sauf s'ils sont spécifiquement configurés pour écouter sur une accès TCP de remplacement. Les accès de remplacement PEUVENT être utilisés lorsque plusieurs KDC fonctionnent sur plusieurs domaines sur le même hôte.

Les clients DOIVENT prendre en charge l'envoi des demandes TCP, mais PEUVENT choisir d'essayer une demande en utilisant initialement le transport UDP. Les clients DEVRAIENT utiliser la découverte de KDC [7.2.3] pour identifier l'adresse et le port IP auquel ils enverront leur demande. Note de mise en œuvre : Certaines extensions au protocole Kerberos ne réussiront pas si aucun client ou KDC prenant en charge le transport TCP n'est impliqué. Les mises en œuvre de la RFC 1510 n'étaient pas obligées de prendre en charge les transports TCP/IP.

Lorsque le message KRB_KDC_REQ est envoyé au KDC sur un flux TCP, la réponse (message KRB_KDC_REP ou KRB_ERROR) DOIT être retournée au client sur le même flux TCP qui était établis pour la demande. Le KDC PEUT clore le flux TCP après avoir envoyé une réponse, mais PEUT laisser le flux ouvert pendant une durée raisonnable si il attend une suite. Il faut veiller, dans la gestion des connexions TCP/IP avec le KDC, à empêcher les attaques de déni de service fondées sur le nombre de connexions TCP/IP ouvertes.

Le client DOIT être prêt à voir le flux clos par le KDC à tout moment après la réception d'une réponse. Une clôture de flux NE DEVRAIT PAS être traitée comme une erreur fatale. Plutôt, si plusieurs échanges sont exigés (par exemple, par certaines formes de pré-authentification), le client peut avoir besoin d'établir une nouvelle connexion quand il est prêt à envoyer les messages suivants. Un client PEUT clore le flux après réception d'une réponse, et DEVRAIT clore le flux s'il ne prévoit pas d'envoyer de messages de suite.

Un client PEUNT envoyer plusieurs demandes avant de recevoir des réponses, bien qu'il doive être prêt à traiter la clôture de la connexion après la première réponse.

Chaque demande (KRB_KDC_REQ) et réponse (KRB_KDC_REP ou KRB_ERROR) envoyée sur le flux TCP est précédée par la longueur de la demande en quatre octets dans l'ordre des octets du réseau. Le plus fort bit de la longueur est réservé à une expansion future et DOIT actuellement être mis à zéro. Si un KDC qui ne comprend pas comment interpréter un bit de plus fort poids établi *(mis à 1)* dans le codage de longueur, reçoit une demande avec le bit de plus fort poids établi, il DOIT retourner un message KRB-ERROR avec l'erreur KRB_ERR_FIELD_TOOLONG et DOIT clore le flux TCP.

Si plusieurs demandes sont envoyées sur une seule connexion TCP et si le KDC envoie plusieurs réponses, le KDC n'est pas obligé d'envoyer les réponses dans l'ordre des demandes correspondantes. Cela peut permettre à certaines mises en œuvre d'envoyer chaque réponse aussitôt qu'elles sont prêtes, même si des demandes antérieures sont toujours en cours de traitement (par exemple, en attente d'une réponse d'un appareil ou base de données externe).

7.2.3 Découverte de KDC sur les réseaux IP

Les mises en œuvre de client Kerberos DOIVENT fournir un moyen pour que le client détermine la localisation des centres de distribution de clés Kerberos (KDC). Traditionnellement, les mises en œuvre de Kerberos mémorisent des telles informations de configuration dans un fichier sur chaque machine cliente. L'expérience a montré que cette méthode de mémorisation des informations de configuration pose des problèmes d'informations périmées et d'évaluation, tout particulièrement lors de l'utilisation de l'authentification inter domaine. Ce paragraphe décrit une méthode d'utilisation du système de noms de domaine de la [RFC1035] pour mémoriser les informations de localisation des KDC.

7.2.3.1 DNS contre Kerberos : Sensibilité à la casse des noms de domaine

Dans Kerberos, les noms de domaine sont sensibles à la casse. Bien qu'il soit fortement recommandé que tous les noms de domaine soient en majuscules, cette recommandation n'a pas été adoptée par tous les sites. Certains sites utilisent des noms tout en minuscules et d'autres utilisent une casse mélangée. DNS, d'un autre côté, est insensible à la casse pour les interrogations. Parce que les noms de domaine "MYREALM", "myrealm", et "MyRealm" sont tous différents, mais se résolvent de la même façon dans le système de noms de domaine, il est nécessaire qu'une seule des combinaisons possibles de caractères majuscules et minuscules soit utilisée dans les noms de domaine.

7.2.3.2 Spécification des informations de localisation de KDC avec les enregistrements DNS SRV

Les informations de localisation de KDC sont à mémoriser en utilisant le RR SRV de DNS [RFC2782]. Le format de ce RR est le suivant :

_Service._Proto.Realm TTL Class SRV Priority Weight Port Target

Le nom de Service pour Kerberos est toujours "kerberos".

Le Proto peut être "udp" ou "tcp". Si ces enregistrements de SRV doivent être utilisés, les enregistrements "udp" et "tcp" DOIVENT tous deux être spécifiés pour tous les développements de KDC.

Le Realm est le domaine Kerberos auquel cet enregistrement correspond. Le domaine DOIT être un nom de domaine de style domaine.

TTL, Class, SRV, Priority, Weight, et Target ont la signification standard définie dans la RFC 2782.

Conformément à la RFC 2782, le numéro de port utilisé pour les enregistrements de SRV "_udp" et "_tcp" DEVRAIT être la valeur allouée à "kerberos" par l'Autorité d'allocation des numéros Internet (IANA) : 88 (décimal), sauf si le KDC est configuré pour écouter sur un autre accès TCP.

Note de mise en œuvre : De nombreuses mises en œuvre de client existantes ne prennent pas en charge la découverte de KDC et sont configurées pour envoyer des demandes sur l'accès alloué par l'IANA (88 décimal), aussi il est vivement recommandé que les KDC soient configurés pour écouter sur cet accès.

7.2.3.3 Découverte de KDC pour les noms de domaine de style Domaine sur les réseaux IP

Ce sont des enregistrements DNS pour un domaine Kerberos EXAMPLE.COM. Il y a deux serveurs Kerberos, kdc1.example.com et kdc2.example.com. Les interrogations devraient d'abord être dirigées sur kdc1.example.com conformément à la priorité spécifiée. Les pondérations ne sont pas utilisées dans ces échantillons d'enregistrements.

```
_kerberos._udp.EXAMPLE.COM.   IN   SRV   0   0   88   kdc1.example.com.
_kerberos._udp.EXAMPLE.COM.   IN   SRV   1   0   88   kdc2.example.com.
_kerberos._tcp.EXAMPLE.COM.   IN   SRV   0   0   88   kdc1.example.com.
_kerberos._tcp.EXAMPLE.COM.   IN   SRV   1   0   88   kdc2.example.com.
```

7.3 Nom du TGS

L'identifiant de principal du service d'allocation de tickets doit être composé de trois parties : le domaine du KDC qui produit le ticket TGS, et un nom en deux parties du type NT-SRV-INST, dont la première partie est "krbtgt" et la seconde partie est le nom du domaine qui va accepter le TGT. Par exemple, un TGT produit par le domaine ATHENA.MIT.EDU pour être utilisé à obtenir des tickets du KDC ATHENA.MIT.EDU a un identifiant de principal de "ATHENA.MIT.EDU" (domaine), ("krbtgt", "ATHENA.MIT.EDU") (nom). Un TGT produit par le domaine ATHENA.MIT.EDU pour être utilisé à obtenir des tickets du domaine MIT.EDU a l'identifiant de principal de "ATHENA.MIT.EDU" (domaine), ("krbtgt", "MIT.EDU") (nom).

7.4 OID Arc pour KerberosV5

Cet OID PEUT être utilisé pour identifier les messages de protocole Kerberos encapsulés dans d'autres protocoles. Il désigne aussi l'OID arc pour les IOD qui se rapportent à KerberosV5 qui seront alloués par une action de l'IETF.
Note de mise en œuvre : La RFC 1510 avait une valeur incorrecte (5) pour "dod" dans son OID.

```
id-krb5  OBJECT IDENTIFIER ::= {
         iso(1) identified-organisation(3) dod(6) internet(1) security(5) kerberosV5(2)
  }
```

L'allocation des OID en –dessous de id-krb5 arc doit être obtenue en contactant le registraire pour l'arc id-krb5, ou son représentant. Au moment de la publication de la présente RFC, de telles inscriptions peuvent être obtenues en contactant krb5-oid-registrar@mit.edu.

7.5 Constantes du protocole et valeurs associées

Les tableaux qui suivent font la liste des constantes utilisées dans le protocole et définissent leur signification. Dans la partie "spécification" sont spécifiées les gammes qui limitent les valeurs des constantes pour lesquelles les valeurs sont

définies ici. Cela permet aux mises en œuvre de faire des hypothèses sur les valeurs maximales qui seront reçues pour ces constantes. Les mises en œuvre recevant des valeurs en-dehors de la gamme spécifiée dans la partie "spécification" PEUVENT rejeter la demande, mais elles DOIVENT se récupérer proprement.

7.5.1 Numéros d'utilisation de clé

La spécification du chiffrement et de la somme de contrôle dans la [RFC3961] exige en entrée un "numéro d'usage de clé", pour altérer la clé de chiffrement utilisée dans tout message spécifique afin de rendre plus difficiles certains types d'attaques cryptographique. Ce sont les valeurs d'usage de clé allouées dans le présent document :

1. Horodatage padata AS-REQ PA-ENC-TIMESTAMP, chiffré avec la clé de client (paragraphe 5.2.7.2)
2. Tickets AS-REP et TGS-REP (inclut la clé de session TGS ou la clé de session d'application), chiffrés avec la clé de service (paragraphe 5.3)
3. Partie chiffrée de AS-REP (inclut la clé de session TGS ou la clé de session d'application), chiffrée avec la clé de client (paragraphe 5.4.2)
4. Données d'autorisation TGS-REQ KDC-REQ-BODY, chiffrées avec la clé de session TGS (paragraphe 5.4.1)
5. Données d'autorisation TGS-REQ KDC-REQ-BODY, chiffrées avec la sous-clé d'authentifiant TGS (paragraphe 5.4.1)
6. Somme de contrôle d'authentifiant AP-REQ de padata TGS-REQ PA-TGS-REQ, frappée avec la clé de session TGS (paragraphe 5.5.1)
7. Authentifiant AP-REQ de padata TGS-REQ PA-TGS-REQ (inclut la sous-clé d'authentifiant de TGS), chiffré avec la clé de session TGS (paragraphe 5.5.1)
8. Partie chiffrée de TGS-REP (inclut la clé de session d'application), chiffrée avec la clé de session TGS (paragraphe 5.4.2)
9. Partie chiffrée de TGS-REPt (inclut la clé de session d'application), chiffrée avec la sous-clé d'authentifiant de TGS (paragraphe 5.4.2)
10. Somme de contrôle d'authentifiant AP-REQ, frappée avec la clé de session d'application (paragraphe 5.5.1)
11. Authentifiant AP-REQ (inclut la sous-clé d'authentifiant d'application), chiffré avec la clé de session d'application (paragraphe 5.5.1)
12. Partie chiffrée de AP-REP (inclut la sous-clé de session d'application), chiffré avec la clé de session d'application (paragraphe 5.5.2)
13. Partie chiffrée de KRB-PRIV, chiffrée avec une clé choisie par l'application (paragraphe 5.7.1)
14. Partie chiffrée de KRB-CRED, chiffrée avec une clé choisie par l'application (paragraphe 5.8.1)
15. Somme de contrôle KRB-SAFE, frappé avec une clé choisie par l'application (paragraphe 5.6.1)

16-18. Réservés à une utilisation future dans Kerberos et les protocoles qui s'y rapportent.

19. Somme de contrôle AD-KDC-ISSUED (ad-checksum au paragraphe 5.2.6.4)

20-21. Réservés à une utilisation future dans Kerberos et les protocoles qui s'y rapportent.

22-25. Réservés à une utilisation future dans les mécanismes GSS-API de Kerberos Version 5 [RFC4121].

26-511. Réservés à une utilisation future dans Kerberos et les protocoles qui s'y rapportent.

512-1023. Réservés à des utilisations internes à une mise en œuvre Kerberos.

1024. Chiffrement à usage d'application dans les protocoles qui ne spécifient pas de valeurs d'usage de clés.

1025. Sommes de contrôle à usage d'application dans les protocoles qui ne spécifient pas de valeurs d'usage de clés

1026-2047. Réservé à l'usage d'application.

7.5.2 Types de données de pré-authentification

Padata et type de données	Padata-type	Commentaire
PA-TGS-REQ	1	
PA-ENC-TIMESTAMP	2	
PA-PW-SALT	3	
[reserved]	4	
PA-ENC-UNIX-TIME	5	(déconseillé)
PA-SANDIA-SECUREID	6	
PA-SESAME	7	
PA-OSF-DCE	8	
PA-CYBERSAFE-SECUREID	9	

PA-AFS3-SALT	10	
PA-ETYPE-INFO	11	
PA-SAM-CHALLENGE	12	(sam/otp)
PA-SAM-RESPONSE	13	(sam/otp)
PA-PK-AS-REQ_OLD	14	(pkinit)
PA-PK-AS-REP_OLD	15	(pkinit)
PA-PK-AS-REQ	16	(pkinit)
PA-PK-AS-REP	17	(pkinit)
PA-ETYPE-INFO2	19	(remplace pa-etype-info)
PA-USE-SPECIFIED-KVNO	20	
PA-SAM-REDIRECT	21	(sam/otp)
PA-GET-FROM-TYPED-DATA	22	(incorporé dans typed-data)
TD-PADATA	22	(incorpore padata)
PA-SAM-ETYPE-INFO	23	(sam/otp)
PA-ALT-PRINC	24	(crawdad@fnal.gov)
PA-SAM-CHALLENGE2	30	(kenh@pobox.com)
PA-SAM-RESPONSE2	31	(kenh@pobox.com)
PA-EXTRA-TGT	41	Réservé extra TGT
TD-PKINIT-CMS-CERTIFICATES	101	CertificateSet du CMS
TD-KRB-PRINCIPAL	102	PrincipalName
TD-KRB-REALM	103	Domaine
TD-TRUSTED-CERTIFIERS	104	de PKINIT
TD-CERTIFICATE-INDEX	105	de PKINIT
TD-APP-DEFINED-ERROR	106	spécifique de l’application
TD-REQ-NONCE	107	ENTIER
TD-REQ-SEQ	108	ENTIER
PA-PAC-REQUEST	128	(jbrezak@exchange.microsoft.com)

7.5.3 Types d’adresse

Type d’adresse	**Valeur**
IPv4	2
Directionnelle	3
ChaosNet	5
XNS	6
ISO	7
DECNET Phase IV	12
AppleTalk DDP	16
NetBios	20
IPv6	24

7.5.4 Types de données d’autorisation

Types de données d’autorisation	**Valeur de Ad-type**
AD-IF-RELEVANT	1
AD-INTENDED-FOR-SERVER	2
AD-INTENDED-FOR-APPLICATION-CLASS	3
AD-KDC-ISSUED	4
AD-AND-OR	5
AD-MANDATORY-TICKET-EXTENSIONS	6
AD-IN-TICKET-EXTENSIONS	7
AD-MANDATORY-FOR-KDC	8
Valeurs réservées	9-63
OSF-DCE	64
SESAME	65
AD-OSF-DCE-PKI-CERTID	66 (hemsath@us.ibm.com)
AD-WIN2K-PAC	128 (jbrezak@exchange.microsoft.com)
AD-ETYPE-NEGOTIATION	129 (lzhu@windows.microsoft.com)

7.5.5 Types de codage traversés

Type de codage traversé	Valeur de Tr-type
DOMAIN-X500-COMPRESS	1
Valeurs réservées	Toutes les autres

7.5.6 Numéro de version du protocole

Étiquette	Valeur	Signification ou Code MIT
pvno	5	Numéro de version en cours du protocole Kerberos

7.5.7 Types de message Kerberos

Type de message	Valeur	Signification
KRB_AS_REQ	10	Demande d'authentification initiale
KRB_AS_REP	11	Réponse à demande KRB_AS_REQ
KRB_TGS_REQ	12	Demande d'authentification fondée sur un TGT
KRB_TGS_REP	13	Réponse à demande KRB_TGS_REQ
KRB_AP_REQ	14	Demande d'application au serveur
KRB_AP_REP	15	Réponse à KRB_AP_REQ_MUTUAL
KRB_RESERVED16	16	Réservé à une demande krb_tgt_request d'usager à usager
KRB_RESERVED17	17	Réservé à une réponse krb_tgt_reply d'usager à usager
KRB_SAFE	20	Message d'application sûr (avec somme de contrôle)
KRB_PRIV	21	Message d'application privé (chiffré)
KRB_CRED	22	Message privé (chiffré) pour transmission d'accréditifs
KRB_ERROR	30	Réponse d'erreur

7.5.8 Types de nom

Type de nom	Valeur	Signification
KRB_NT_UNKNOWN	0	Type de nom inconnu
KRB_NT_PRINCIPAL	1	Juste le nom du principal comme dans DCE, ou pour utilisateurs
KRB_NT_SRV_INST	2	Service et autre instance unique (krbtgt)
KRB_NT_SRV_HST	3	Service avec nom d'hôte comme instance (telnet, rcommands)
KRB_NT_SRV_XHST	4	Service avec hôte comme composants restants
KRB_NT_UID	5	ID unique
KRB_NT_X500_PRINCIPAL	6	Nom distinctif X.509 codé [RFC2253]
KRB_NT_SMTP_NAME	7	Nom en forme de nom de messagerie électronique SMTP (par exemple, user@example.com)
KRB_NT_ENTERPRISE	10	Nom d'entreprise ; peut être transposé en nom de principal

7.5.9 Codes d'erreur

Code d'erreur	Valeur	Signification
KDC_ERR_NONE	0	Pas d'erreur
KDC_ERR_NAME_EXP	1	L'entrée du client dans la base de données a expiré
KDC_ERR_SERVICE_EXP	2	L'entrée du serveur dans la base de données a expiré
KDC_ERR_BAD_PVNO	3	Numéro de version du protocole demandé non accepté
KDC_ERR_C_OLD_MAST_KVNO	4	La clé du client est chiffrée avec la vieille clé maîtresse
KDC_ERR_S_OLD_MAST_KVNO	5	La clé du serveur est chiffrée avec la vieille clé maîtresse
KDC_ERR_C_PRINCIPAL_UNKNOWN	6	Client non trouvé dans la base de données Kerberos
KDC_ERR_S_PRINCIPAL_UNKNOWN	7	Serveur non trouvé dans la base de données Kerberos
KDC_ERR_PRINCIPAL_NOT_UNIQUE	8	Plusieurs entrées du principal dans la de données
KDC_ERR_NULL_KEY	9	Le client ou le serveur a une clé nulle
KDC_ERR_CANNOT_POSTDATE	10	Ticket non éligible au postdatage
KDC_ERR_NEVER_VALID	11	Heure de début demandée postérieure à l'heure de fin
KDC_ERR_POLICY	12	La politique du KDC rejette la demande
KDC_ERR_BADOPTION	13	Le KDC ne peut pas traiter l'option demandée
KDC_ERR_ETYPE_NOSUPP	14	Le KDC ne prend pas en charge le type de chiffrement
KDC_ERR_SUMTYPE_NOSUPP	15	Le KDC n'accepte pas le type de somme de contrôle

KDC_ERR_PADATA_TYPE_NOSUPP	16	Le KDC ne prend pas en charge le type de padata
KDC_ERR_TRTYPE_NOSUPP	17	Le KDC ne prend pas en charge le type transité
KDC_ERR_CLIENT_REVOKED	18	Les accréditifs du client ont été révoqués
KDC_ERR_SERVICE_REVOKED	19	Les accréditifs du serveur ont été révoqués
KDC_ERR_TGT_REVOKED	20	Le TGT a été révoqué
KDC_ERR_CLIENT_NOTYET	21	Client pas encore valide ; réessayer plus tard
KDC_ERR_SERVICE_NOTYET	22	Serveur pas encore valide ; réessayer plus tard
KDC_ERR_KEY_EXPIRED	23	Mot de passe expiré ; changer le mot de passe pour recommencer
KDC_ERR_PREAUTH_FAILED	24	Informations de pré-authentification invalides
KDC_ERR_PREAUTH_REQUIRED	25	Pré-authentification supplémentaire exigée
KDC_ERR_SERVER_NOMATCH	26	Le serveur demandé et le ticket ne correspondent pas
KDC_ERR_DOIIVENT_USE_USER2USER	27	Principal de serveur valide seulement d'usager à usager
KDC_ERR_PATH_NOT_ACCEPTED	28	La politique du KDC rejette le chemin de transit
KDC_ERR_SVC_UNAVAILABLE	29	Un service n'est pas disponible
KRB_AP_ERR_BAD_INTEGRITY	31	Échec de vérification d'intégrité sur le champ déchiffré
KRB_AP_ERR_TKT_EXPIRED	32	Ticket expiré
KRB_AP_ERR_TKT_NYV	33	Ticket pas encore valide
KRB_AP_ERR_REPEAT	34	La demande est une répétition
KRB_AP_ERR_NOT_US	35	Le ticket n'est pas pour nous
KRB_AP_ERR_BADMATCH	36	Ticket et authentifiant ne correspondent pas
KRB_AP_ERR_SKEW	37	Biais d'horloge trop grand
KRB_AP_ERR_BADADDR	38	Adresse réseau incorrecte
KRB_AP_ERR_BADVERSION	39	Discordance de version de protocole
KRB_AP_ERR_MSG_TYPE	40	Type de message invalide
KRB_AP_ERR_MODIFIED	41	Flux de message modifié
KRB_AP_ERR_BADORDER	42	Message pas à son ordre
KRB_AP_ERR_BADKEYVER	44	Version de clé spécifiée non disponible
KRB_AP_ERR_NOKEY	45	Clé de service non disponible
KRB_AP_ERR_MUT_FAIL	46	Échec d'authentification mutuelle
KRB_AP_ERR_BADDIRECTION	47	Direction de message incorrecte
KRB_AP_ERR_METHOD	48	Autre méthode d'authentification exigée
KRB_AP_ERR_BADSEQ	49	Numéro de séquence incorrect dans le message
KRB_AP_ERR_INAPP_CKSUM	50	Type de somme de contrôle inapproprié dans le message
KRB_AP_PATH_NOT_ACCEPTED	51	La politique rejette le chemin de transit
KRB_ERR_RESPONSE_TOO_BIG	52	Réponse trop grosse pour UDP ; ressayer avec TCP
KRB_ERR_GENERIC	60	Erreur générique (description dans e-text)
KRB_ERR_FIELD_TOOLONG	61	Le champ est trop long pour cette mise en œuvre
KDC_ERROR_CLIENT_NOT_TRUSTED	62	Réservé pour PKINIT
KDC_ERROR_KDC_NOT_TRUSTED	63	Réservé pour PKINIT
KDC_ERROR_INVALID_SIG	64	Réservé pour PKINIT
KDC_ERR_KEY_TOO_WEAK	65	Réservé pour PKINIT
KDC_ERR_CERTIFICATE_MISMATCH	66	Réservé pour PKINIT
KRB_AP_ERR_NO_TGT	67	Pas de TGT disponible pour valider USER-TO-USER
KDC_ERR_WRONG_REALM	68	Réservé pour utilisation future
KRB_AP_ERR_USER_TO_USER_REQUIRED	69	Le ticket doit être pour USER-TO-USER
KDC_ERR_CANT_VERIFY_CERTIFICATE	70	Réservé pour PKINIT
KDC_ERR_INVALID_CERTIFICATE	71	Réservé pour PKINIT
KDC_ERR_REVOKED_CERTIFICATE	72	Réservé pour PKINIT
KDC_ERR_REVOCATION_STATUS_UNKNOWN	73	Réservé pour PKINIT
KDC_ERR_REVOCATION_STATUS_UNAVAILABLE	74	Réservé pour PKINIT
KDC_ERR_CLIENT_NAME_MISMATCH	75	Réservé pour PKINIT
KDC_ERR_KDC_NAME_MISMATCH	76	Réservé pour PKINIT

8 Exigences d'interopérabilité

La version 5 du protocole Kerberos accepte une myriade d'options. Parmi celles-ci, plusieurs types de chiffrement et de sommes de contrôle ; des schéma de codage de remplacement pour le champ de transit ; des mécanismes facultatifs pour la pré-authentification ; le traitement de tickets sans adresse ; des options pour l'authentification mutuelle ;

l'authentification d'usager à usager ; la prise en charge de mandataires ; le format des noms de domaine ; le traitement des données d'autorisation ; et la transmission, le postdatage, et le renouvellement des tickets.

Afin d'assurer l'interopérabilité des domaines, il est nécessaire de définir une configuration minimale qui doit être prise en charge par toutes les mises en œuvre. Cette configuration minimale est sujette à changement, comme le font les technologies. Par exemple, si ultérieurement il est découvert qu'un des algorithmes de chiffrement ou de somme de contrôle exigés n'est pas sûr, il sera remplacé.

8.1 Spécification 2

Ce paragraphe définit la seconde spécification de ces options. Les mises en œuvre qui sont configurées de cette façon peuvent être considérées comme prenant en charge la spécification 2 de Kerberos Version 5 (5.2). La spécification 1 (déconseillée) peut être trouvée dans la RFC 1510.

Transport
Le transport TCP/IP et UDP/IP DOIVENT être pris en charge par les clients et KDC qui revendiquent la conformité à la spécification 2.

Méthodes de chiffrement et de somme de contrôle
Les mécanismes de chiffrement et de somme de contrôle suivants DOIVENT être pris en charge :

Chiffrement : AES256-CTS-HMAC-SHA1-96 [RFC3962]
Somme de contrôle : HMAC-SHA1-96-AES256 [RFC3962]

Les mises en œuvre DEVRAIENT aussi accepter d'autres mécanismes, mais les mécanismes supplémentaires ne peuvent être utilisés qu'en communicant avec des principaux connus pour les accepter aussi. Les mécanismes suivants provenant de la [RFC3961] et de la [RFC3962] DEVRAIENT être pris en charge :

Chiffrement : AES128-CTS-HMAC-SHA1-96, DES-CBC-MD5, DES3-CBC-SHA1-KD
Somme de contrôle : DES-MD5, HMAC-SHA1-DES3-KD, HMAC-SHA1-96-AES128

Les mises en œuvre PEUVENT aussi accepter d'autres mécanismes, mais les mécanismes supplémentaires ne peuvent être utilisés qu'en communicant avec des principaux connus pour les accepter aussi.

Note de mise en œuvre : Les premières mises en œuvre de Kerberos génèrent des messages qui utilisent les méthodes de somme de contrôle CRC-32 et RSA-MD5. Pour l'interopérabilité avec ces premières livraisons, les développeurs PEUVENT envisager de prendre en charge ces méthodes de somme de contrôle, mais devraient en analyser attentivement les implications sur la sécurité afin de limiter les situations dans lesquelles ces méthodes sont acceptées.

Noms de domaine
Toutes les mises en œuvre DOIVENT comprendre la hiérarchie des domaines à la fois dans le domaine Internet et dans le style X.500. Lorsque est demandé un TGT pour un domaine inconnu, le KDC DOIT être capable de déterminer les noms des domaines intermédiaires entre le domaine du KDC et l domaine demandé.

Codage de champ de transit
DOMAIN-X500-COMPRESS (décrit au paragraphe 3.3.3.2) DOIT être pris en charge. D'autres codages PEUVENT être acceptés, mais ils ne peuvent être utilisés lorsque ce codage est accepté par TOUS les domaines intermédiaires.

Méthodes de pré-authentification
La méthode TGS-REQ DOIT être prise en charge. Elle n'est pas utilisée sur la demande initiale. La méthode PA-ENC-TIMESTAMP DOIT être acceptée par les clients, mais savoir si elle est activée par défaut PEUT être déterminé domaine par domaine. Si la méthode n'est pas utilisée dans la demande initiale et si l'erreur KDC_ERR_PREAUTH_REQUIRED est retournée, spécifiant PA-ENC-TIMESTAMP comme méthode acceptable, le client DEVRAIT ressayer la demande initiale en utilisant la méthode de pré-authentification PA-ENC-TIMESTAMP. Les serveurs n'ont pas besoin de prendre en charge la méthode PA-ENC-TIMESTAMP, mais si elle n'est pas acceptée, le serveur DEVRAIT ignorer la présence de la pré-authentification PA-ENC-TIMESTAMP dans une demande.

La méthode ETYPE-INFO2 DOIT être acceptée ; cette méthode est utilisée pour communiquer l'ensemble des types de chiffrement acceptés, et les paramètres correspondants de sel et de chaîne à clé. La méthode ETYPE-INFO DEVRAIT

être acceptée pour l'interopérabilité avec les mises en œuvre les plus anciennes.
Authentification mutuelle
L'authentification mutuelle (via le message KRB_AP_REP) DOIT être acceptée.

Adresses et fanions de ticket
Tous les KDC DOIVENT passer les tickets qui ne portent pas d'adresse (c'est-à-dire que si un TGT ne contient pas d'adresse, le KDC retournera des tickets dérivés). Les mises en œuvre DEVRAIENT par défaut demander des tickets sans adresse, car cela augment de façon significative l'interopérabilité avec la traduction d'adresse réseau. Dans certains cas, les domaines ou les serveurs d'application PEUVENT exiger que les tickets aient une adresse.

Les mises en œuvre DEVRAIENT accepter le type d'adresse directionnelle pour les messages KRB_SAFE et KRB_PRIV et DEVRAIENT inclure des adresses directionnelles dans ces messages quand d'autres types d'adresse ne sont pas disponibles.

Les tickets mandataires et transmis DOIVENT être acceptés. Les domaines et serveurs d'application individuels peuvent y régler leur propre politique lorsque de tels tickets seront acceptés.

Toutes les mises en œuvre DOIVENT reconnaître les tickets renouvelables et postdatés, mais ils n'ont pas besoin de les mettre réellement en œuvre. Si ces options ne sont pas prises en charge, l'heure de début et l'heure de fin dans le ticket DEVRONT spécifier l'entière durée de vie utile d'un ticket. Lorsque un ticket postdaté est décodé par un serveur, toutes les mises en œuvre DEVRONT rendre visible la présence du fanion postdaté pour le serveur appelant.

Authentification d'usager à usager
La prise en charge de l'authentification d'usager à usager (via l'option ENC-TKT-IN-SKEY KDC) DOIT être fournie par les mises en œuvre, mais les domaines individuels PEUVENT décider au titre de leur politique de rejeter de telles demandes principal par principal ou domaine par domaine.

Données d'autorisation
Les mises en œuvre DOIVENT passer tous les sous-champs de données d'autorisation des TGT à tout ticket dérivé sauf si ils sont amenés à supprimer un sous-champ au titre de la définition de ce type de sous-champ enregistré. (Il n'est jamais incorrect de passer sur un sous-champ, et actuellement, aucun type de sous-champ enregistré ne spécifie la suppression au KDC.)

Les mises en œuvre DOIVENT rendre disponible le contenu de tout sous-champ de données d'autorisation au serveur lorsque le ticket est utilisé. Les mises en œuvre ne sont pas obligées de permettre aux clients de spécifier le contenu des champs de données d'autorisation.

Gammes de constantes
Toutes les constantes du protocole sont contraintes à des valeurs de 32 bits (signés) sauf contraintes supplémentaires provenant de la définition du protocole. Cette limite est donnée pour permettre aux mises en œuvre de faire des hypothèses sur les valeurs maximales qui seront reçues pour ces constantes. Les mises en œuvre recevant des valeurs hors de ces gammes PEUVENT rejeter la demande, mais elles DOIVENT se récupérer proprement.

8.2 Valeurs de KDC recommandées

Ci-après est fournie une liste de valeurs recommandées pour une configuration de KDC.

Durée de vie minimum	5 minutes
Durée de vie maximum renouvelable	1 semaine
Durée de vie maximum de ticket	1 jour
Biais d'horloge admissible	5 minutes
Adresses vides	Admis
Mandatable, etc.	Admis

9 Considérations relatives à l'IANA

La section 7 du présent document spécifie des constantes du protocole et autres valeurs définies exigées pour

l'interopérabilité de plusieurs mises en œuvre. Jusqu'à ce qu'une RFC ultérieure ne spécifie autre chose, ou que le groupe de travail Kerberos ne soit clos, l'allocation de constantes de protocole supplémentaires et autres valeurs définies exigées pour des extensions au protocole Kerberos sera administrée par le groupe de travail Kerberos. Suivant les recommandations formulées dans la [RFC2434], les lignes directrices fournies par l'IANA sont les suivantes :

Les types de nom de domaine "réservés" du paragraphe 6.1 et les types de domaine "autres" excepté ceux commençant par "X-" ou "x-" ne seront pas enregistrés sans un e action de normalisation de l'IETF, moment auquel des lignes directrices pour une allocation ultérieure seront spécifiées. Les types de nom de domaine commençant par "X-" ou "x-" sont pour utilisation privée.

Pour les types d'adresse d'hôte décrits au paragraphe 7.1, les valeurs négatives sont pour utilisation privée. L'allocation de numéros positifs supplémentaires est soumise à révision par le groupe de travail Kerberos ou autre groupe d'experts.

Les numéros d'usage de clé supplémentaires, comme défini au paragraphe 7.5.1, seront alloués sous réserve de révision par le groupe de travail Kerberos ou autre groupe d'experts.

Les valeurs supplémentaires de type de données de pré-authentification, comme définies au paragraphe 7.5.2, seront alloués sous réserve de révision par le groupe de travail Kerberos ou autre groupe d'experts.

Les types supplémentaires de données d'autorisation, comme définis au paragraphe 7.5.4, seront alloués sous réserve de révision par le groupe de travail Kerberos ou autre groupe d'experts. Bien qu'on puisse supposer qu'il pourrait y avoir une demande significative de types d'utilisation privée, il n'est intentionnellement pas fait de place à la portion d'utilisation privée dans l'espace de nom parce que les conflits entre valeurs allouées privatives pourraient avoir des implications préjudiciables à la sécurité.

Les types supplémentaires de codage de transit, comme définis au paragraphe 7.5.5, présentent des problèmes particuliers pour l'interopérabilité avec les mises en œuvre existantes. Comme tels, ces allocations ne seront faites que par des actions de normalisation, sauf si le groupe de travail Kerberos ou un autre groupe de travail ayant une compétence dans ce domaine fait des allocations préliminaires pour des documents qui passent par le processus de normalisation.

Les types supplémentaires de message Kerberos, comme définis au paragraphe 7.5.7, seront alloués sous réserve de révision par le groupe de travail Kerberos ou autre groupe d'experts.

Les types de nom supplémentaires, comme décrits au paragraphe 7.5.8, seront alloués sous réserve de révision par le groupe de travail Kerberos ou autre groupe d'experts.

Des codes d'erreur supplémentaires à ceux décrits au paragraphe 7.5.9 seront alloués sous réserve de révision par le groupe de travail Kerberos ou autre groupe d'experts.

10 Considérations sur la sécurité

Comme service d'authentification, Kerberos donne les moyens de vérifier l'identité des principaux sur un réseau. Par lui-même Kerberos ne fournit pas d'autorisation. Les applications ne devraient pas accepter que la délivrance d'un ticket de service par le serveur Kerberos accorde l'autorité d'utiliser le service, car de telles applications peuvent devenir vulnérables à l'outrepassement de cette vérification d'autorisation dans un environnement où ils interopèrent avec d'autres KDC ou dans lesquels sont fournies d'autres options d'authentification d'application.

Les attaques de déni de service ne sont pas résolues par Kerberos. Il y a des endroits dans les protocoles où un intrus peut empêcher une application de participer aux étapes d'authentification appropriées. Comme l'authentification est une étape exigée pour l'utilisation de nombreux services, des attaques réussies de déni de service contre un serveur Kerberos peuvent résulter en un déni d'autres services réseau qui s'appuient sur Kerberos pour l'authentification. Kerberos est vulnérable à de nombreuses sortes d'attaques de déni de service : celles contre le réseau, qui empêcheraient les clients de contacter le KDC ; celles contre le système de noms de domaine, qui pourraient empêcher un client de trouver les adresses IP du serveur Kerberos ; et celles en surcharge du KDC Kerberos lui-même par des demandes répétées.

Les conflits d'interopérabilité causés par une utilisation de jeu de caractères incompatible (voir au paragraphe 5.2.1) peuvent résulter en un déni de service pour les clients qui utilisent dans des chaînes Kerberos des jeux de caractères

autres que ceux mémorisés dans la base de données du KDC.

Les serveurs d'authentification maintiennent une base de données de principaux (c'est-à-dire, utilisateurs et serveurs) et leurs clés secrètes. La sécurité des machines de serveur d'authentification est critique. Une brèche dans la sécurité d'un serveur d'authentification compromettrait la sécurité de tous les serveurs qui s'appuient sur le KDC compromis, et compromettrait l'authentification de tout principal enregistré dans le domaine du KDC compromis.

Les principaux doivent tenir leurs clés secrètes secrètes. Si un intrus vole d'une façon ou d'une autre la clé d'un principal, il sera capable de se faire passer pour ce principal ou de faire passer tout serveur pour le principal légitime.

Les attaques par mot de passe deviné ne sont pas résolues par Kerberos. Si un utilisateur choisit un mot de passe faible, il est possible qu'un attaquant réussisse une attaque de dictionnaire hors ligne en essayant de façon répétitive de déchiffrer, avec les entrées successives d'un dictionnaire, les messages obtenus qui sont chiffrés avec une clé dérivée du mot de passe de l'usager.

Sauf si les options de pré-authentification sont exigées par la politique d'un domaine, le KDC ne saura pas si une demande d'authentification réussit. Un attaquant peut demander une réponse avec des accréditifs pour n'importe quel principal. Ces accréditifs ne seront vraisemblablement que de peu d'usage à l'attaquant s'il ne connaît pas la clé secrète du client, mais la disponibilité de la réponse chiffrée avec la clé secrète du client donne à l'attaquant le texte chiffré qui peut être utilisé pour monter des attaques en force ou de dictionnaire pour déchiffrer les accréditifs, en devinant le mot de passe de l'usager. Pour cette raison, il est fortement recommandé que les domaines Kerberos exigent l'utilisation de la pré-authentification. Même avec la pré-authentification, les attaquants peuvent essayer les attaques en force ou de dictionnaire contre des accréditifs qui sont observés par espionnage sur le réseau.

Parce qu'un client peut demander un ticket pour tout principal de serveur et peut tenter une attaque en force ou de dictionnaire contre la clé du principal de serveur en utilisant ce ticket, il est fortement recommandé que ces clés soient générées de façon aléatoire (plutôt que générées à partir de mots de passe) pour tous les principaux qui sont utilisables comme principal de cible pour un message KRB_TGS_REQ ou KRB_AS_REQ [RFC4086].

Bien que les méthodes de chiffrement DES-CBC-MD5 et de somme de contrôle DES-MD5 soient mentionnées comme DEVRAIENT être mises en œuvre pour la rétro compatibilité, le seul algorithme de chiffrement (DES) sur lequel elles se fondent est faible, et des algorithmes plus forts devraient être utilisés chaque fois que possible.

Chaque hôte sur le réseau doit avoir une horloge approximativement synchronisée à l'heure des autres hôtes ; cette synchronisation est utilisée pour réduire les besoins de tenue de registres des serveurs d'application lorsqu'ils font la détection e répétitions. Le degré d'approximation peut être configuré serveur par serveur, mais il est normalement de l'ordre de 5 minutes. Si les horloges sont synchronisées sur le réseau, le protocole de synchronisation d'horloges DOIT être lui-même sécurisé contre les attaquants du réseau.

Les identifiants de principal ne doivent pas être recyclés à court terme. Un mode normal de contrôle d'accès utilisera des listes de contrôle d'accès (ACL) pour accorder les permissions à des principaux particuliers. Si une entrée d'ACL périmée reste d'un principal périmé et si l'identifiant de principal est réutilisé, le nouveau principal va hériter des droits spécifiés dans l'entrée d'ACL périmée. En ne réutilisant pas les identifiants de principal, le danger d'accès par inadvertance est supprimé.

Le déchiffrement approprié d'un message KRB_AS_REP provenant du KDC n'est pas suffisant pour que l'hôte vérifie l'identité de l'usager ; l'usager et un attaquant pourraient coopérer pour générer un message de format KRB_AS_REP qui se déchiffre correctement mais ne viendrait pas du bon KDC. Pour authentifier l'inscription d'un usager sur un système local, les accréditifs obtenus dans l'échange d'AS peuvent d'abord être utilisés dans un échange de TGS pour obtenir des accréditifs pour un serveur local. Ces accréditifs doivent ensuite être vérifiés par un serveur local grâce à l'achèvement réussi de l'échange client/serveur.

De nombreuses mises en œuvre conformes à la RFC 1510 ignorent les éléments de données d'autorisation inconnus. S'appuyer sur ces mises en œuvre pour honorer des restrictions de données d'autorisation peut créer une faiblesse de la sécurité.

Les accréditifs Kerberos contiennent des informations en clair qui identifient les principaux auxquels elles s'appliquent. Si la confidentialité de ces informations est nécessaire, cet échange devrait être encapsulé dans un protocole pourvoyant à la confidentialité sur l'échange de ces accréditifs.

Les applications doivent veiller à protéger les communications qui suivent l'authentification, soit en utilisant les messages KRB_PRIV ou KRB_SAFE selon de qui est approprié, soit en appliquant leurs propres mécanismes de confidentialité ou d'intégrité sur de telles communications. L'achèvement de l'échange KRB_AP_REQ et KRB_AP_REP sans utilisation ultérieure des mécanismes de confidentialité et d'intégrité ne fournit que l'authentification des parties à la communication et pas à la confidentialité et l'intégrité de la communication suivante. Les applications qui appliquent des mécanismes de protection de la confidentialité et de l'intégrité autres que KRB_PRIV et KRB_SAFE doivent s'assurer que l'étape d'authentification est liée de façon appropriée avec le canal de communication protégé qui est établi par l'application.

A moins que le serveur d'application fournisse ses propres moyens convenables pour protéger contre la répétition (par exemple, une séquence de mise en cause-réponse initiée par le serveur après authentification, ou l'utilisation d'une sous-clé de chiffrement générée par un serveur), le serveur doit utiliser une mémoire cache de répétition pour se souvenir de tous les authentifiants présentés dans la durée du biais d'horloge admissible. Tous les services qui partagent une clé on besoin d'utiliser la même mémoire cache de répétition. Si des mémoires cache de répétition séparées sont utilisées, un authentifiant utilisé avec un tel service pourrait être répété plus tard sur un service différent avec le même principal de service.

Si un serveur perd trace des authentifiants présentés durant le biais d'horloge admissible, il doit rejeter toutes les demandes jusqu'à ce que l'intervalle de biais d'horloge soit passé, donnant l'assurance que tout authentifiant perdu ou répété tombera en dehors du biais d'horloge admissible et ne pourra plus être répété avec succès.

Les mises en œuvre de Kerberos ne devraient pas utiliser de serveurs de répertoire qui ne soient pas de confiance pour déterminer le domaine d'un hôte. Permettre cela autoriserait que la compromission du serveur de répertoire donne à un attaquant la possibilité d'amener le client à accepter l'authentification avec le mauvais principal (c'est-à-dire, un principal avec un nom similaire, mais dans un domaine avec lequel l'hôte légitime n'est pas enregistré).

Les mises en œuvre de Kerberos ne doivent pas utiliser DNS pour transposer un nom en un autre (canoniser) afin de déterminer la partie hôte du nom de principal avec laquelle il va communiquer. Permettre cette canonisation permettrait une compromission du DNS qui résulterait en ce que le client obtiendrait des accréditifs et s'authentifierait correctement au mauvais principal. Bien que le client sache avec qui il communique, cela ne sera pas le principal avec lequel il avait l'intention de communiquer.

Si le serveur Kerberos retourne un TGT pour un domaine 'plus proche' que le domaine désiré, le client peut utiliser la configuration de politique locale pour vérifier que le chemin d'authentification utilisé est acceptable. Autrement, un client peut choisir son propre chemin d'authentification plutôt que de s'appuyer sur le serveur Kerberos pour un choisir un. Dans l'un et l'autre cas, toute information de politique ou de configuration utilisée pour choisir ou valider des chemins d'authentification, par le serveur Kerberos ou par le client, doit être obtenue d'une source de confiance.

Dans sa forme de base, le protocole Kerberos ne fournit pas un secret parfait des communications. Si le trafic a été enregistré par un espion, les messages chiffrés en utilisant le message KRB_PRIV, ou les messages chiffrés en utilisant un chiffrement spécifique de l'application avec des clés échangées en utilisant Kerberos peuvent alors être déchiffrés si la clé de l'usager, du serveur d'application, ou du KDC est ultérieurement découverte. C'est pourquoi la clé de session utilisée pour chiffrer de tels messages, lorsqu'ils sont transmis sur le réseau, est chiffrée avec la clé du serveur d'application. Elle est aussi chiffrée avec la clé de session provenant du TGT de l'usager lorsque elle est retournée à l'usager dans le message KRB_TGS_REP. La clé de session provenant du TGT est envoyée à l'usager dans le message KRB_AS_REP chiffré avec la clé secrète de l'usager et incorporée dans le TGT, qui était chiffré avec la clé du KDC. Les applications qui exigent un secret de transmission parfait doivent échanger les clés par des mécanismes qui fournissent une telle assurance, mais peuvent utiliser Kerberos pour l'authentification du canal chiffré établi avec ces autres moyens.

11 Remerciements

Le présent document est une révision de la RFC 1510 dont le co-auteur était John Kohl. La spécification du protocole Kerberos décrite dans le présent document est le résultat de nombreuses années d'effort. Sur cette période, de nombreux individus ont contribué à la définition du protocole et à la rédaction de la spécification. Il n'est malheureusement pas possible de faire la liste de tous les contributeurs comme auteurs du présent document, bien que nombre de ceux qui ne figurent pas sur cette liste soient des auteurs en esprit, y compris ceux qui ont fourni des textes de contribution pour des parties de certaines sections, ceux qui ont contribué à la conception de parties du protocole, et ceux qui ont contribué significativement à la discussion du protocole dans le groupe de travail technique

d’authentification commune (CAT) de l’IETF et le groupe de travail Kerberos.

Parmi ceux qui ont contribué au développement et à la spécification de Kerberos figurent Jeffrey Altman, John Brezak, Marc Colan, Johan Danielsson, Don Davis, Doug Engert, Dan Geer, Paul Hill, John Kohl, Marc Horowitz, Matt Hur, Jeffrey Hutzelman, Paul Leach, John Linn, Ari Medvinsky, Sasha Medvinsky, Steve Miller, Jon Rochlis, Jerome Saltzer, Jeffrey Schiller, Jennifer Steiner, Ralph Swick, Mike Swift, Jonathan Trostle, Theodore Ts'o, Brian Tung, Jacques Vidrine, Assar Westerlund, et Nicolas Williams. De nombreux autres membres du projet Athena du MIT, du groupe réseautage du MIT, et des groupes de travail Kerberos et CAT de l’IETF ont contribué mais ne figurent pas sur la liste.

Annnexe A. Module ASN.1

```
KerberosV5Spec2 {
        iso(1) identified-organisation(3) dod(6) internet(1)
        security(5) kerberosV5(2) modules(4) krb5spec2(2)
} DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- OID arc pour KerberosV5
--
-- Cet OID peut être utilisé pour identifier les messages du protocole Kerberos messages encapsulés dans d’autres protocoles.
--
-- Cet OID désigne aussi l’arc OID pour les OID se rapportant à KerberosV5.
--
-- NOTE : La RFC 1510 avait une valeur incorrecte (5) pour "dod" dans son OID.
id-krb5         OBJECT IDENTIFIER ::= {
        iso(1) identified-organisation(3) dod(6) internet(1)
        security(5) kerberosV5(2)
}

Int32   ::= INTEGER (-2147483648..2147483647)
        -- valeurs signées représentables en 32 bits

UInt32  ::= INTEGER (0..4294967295)
        -- valeurs non signées de 32 bits

Microseconds    ::= INTEGER (0..999999)
        -- microsecondes

KerberosString  ::= GeneralString (IA5String)

Realm   ::= KerberosString

PrincipalName   ::= SEQUENCE {
        name-type           [0] Int32,
        name-string         [1] SEQUENCE OF KerberosString
}

KerberosTime    ::= GeneralizedTime -- sans fraction de seconde

HostAddress     ::= SEQUENCE  {
        addr-type           [0] Int32,
        address  [1] OCTET STRING
}

-- NOTE : HostAddresses est toujours utilisé comme champ FACULTATIF et ne devrait pas être vide.
HostAddresses   -- NOTE : légèrement différent de la rfc1510, mais a une transposition de valeur et se code de la même façon
```

```
        ::= SEQUENCE OF HostAddress

-- NOTE : AuthorizationData est toujours utilisé comme champ FACULTATIF et ne devrait pas être vide.
AuthorizationData       ::= SEQUENCE OF SEQUENCE {
        ad-type  [0] Int32,
        ad-data  [1] OCTET STRING
}

PA-DATA         ::= SEQUENCE {
        -- NOTE : la première étiquette est [1], pas [0]
        padata-type     [1] Int32,
        padata-value    [2] OCTET STRING -- peut être codé AP-REQ
}

KerberosFlags   ::= BIT STRING (SIZE (32..MAX))
        -- nombre minimum de bits qui devra être envoyé, mais pas moins de 32

EncryptedData   ::= SEQUENCE {
        etype   [0] Int32 -- EncryptionType --,
        kvno    [1] UInt32 OPTIONAL,
        cipher  [2] OCTET STRING – texte chiffré
}

EncryptionKey   ::= SEQUENCE {
        keytype         [0] Int32 – en fait type de chiffrement --,
        keyvalue        [1] OCTET STRING
}

Checksum        ::= SEQUENCE {
        cksumtype       [0] Int32,
        checksum        [1] OCTET STRING
}

Ticket                  ::= [APPLICATION 1] SEQUENCE {
        tkt-vno         [0] INTEGER (5),
        realm           [1] Realm,
        sname           [2] PrincipalName,
        enc-part        [3] EncryptedData -- EncTicketPart
}

-- Partie chiffrée du ticket
EncTicketPart           ::= [APPLICATION 3] SEQUENCE {
        flags           [0] TicketFlags,
        key             [1] EncryptionKey,
        crealm          [2] Realm,
        cname           [3] PrincipalName,
        transited       [4] TransitedEncoding,
        authtime        [5] KerberosTime,
        starttime       [6] KerberosTime OPTIONAL,
        endtime         [7] KerberosTime,
        renew-till      [8] KerberosTime OPTIONAL,
        caddr           [9] HostAddresses OPTIONAL,
        authorization-data  [10] AuthorizationData OPTIONAL
}

-- Champ de transit codé
TransitedEncoding       ::= SEQUENCE {
        tr-type         [0] Int32 -- doit être enregistré --,
        contents        [1] OCTET STRING
}
```

```
TicketFlags     ::= KerberosFlags
        -- reserved(0),
        -- forwardable(1),
        -- forwarded(2),
        -- proxiable(3),
        -- proxy(4),
        -- may-postdate(5),
        -- postdated(6),
        -- invalid(7),
        -- renewable(8),
        -- initial(9),
        -- pre-authent(10),
        -- hw-authent(11),
-- les suivants sont nouveaus depuis la rfc 1510
        -- transited-policy-checked(12),
        -- ok-as-delegate(13)

AS-REQ          ::= [APPLICATION 10] KDC-REQ

TGS-REQ         ::= [APPLICATION 12] KDC-REQ

KDC-REQ         ::= SEQUENCE {
        -- NOTE : la première étiquette est [1], pas [0]
        pvno                [1] INTEGER (5) ,
        msg-type            [2] INTEGER (10 -- AS -- | 12 -- TGS --),
        padata              [3] SEQUENCE OF PA-DATA OPTIONAL
        -- NOTE : pas vide --,
        req-body            [4] KDC-REQ-BODY
}

KDC-REQ-BODY            ::= SEQUENCE {
        kdc-options         [0] KDCOptions,
        cname               [1] PrincipalName OPTIONAL   -- Utilisé seulement dans AS-REQ --,
        realm               [2] Realm       -- Domaine de serveur et aussi de client dans AS-REQ --,
        sname               [3] PrincipalName OPTIONAL,
        from                [4] KerberosTime OPTIONAL,
        till                [5] KerberosTime,
        rtime               [6] KerberosTime OPTIONAL,
        nonce               [7] UInt32,
        etype               [8] SEQUENCE OF Int32          -- EncryptionType par ordre de préférence --,
        addresses           [9] HostAddresses OPTIONAL,
        enc-authorization-data      [10] EncryptedData OPTIONAL    -- AuthorizationData --,
        additional-tickets  [11] SEQUENCE OF Ticket OPTIONAL -- NOTE : non vide
}

KDCOptions              ::= KerberosFlags
         -- reserved(0),
        -- forwardable(1),
        -- forwarded(2),
        -- proxiable(3),
        -- proxy(4),
        -- allow-postdate(5),
        -- postdated(6),
        -- unused7(7),
        -- renewable(8),
        -- unused9(9),
        -- unused10(10),
        -- opt-hardware-auth(11),
        -- unused12(12),
```

```
        -- unused13(13),
-- 15 est réservé pour la canonisation
-- unused15(15),
-- 26 était inutilisé dans la rfc 1510
        -- disable-transited-check(26),
--
        -- renewable-ok(27),
        -- enc-tkt-in-skey(28),
        -- renew(30),
        -- validate(31)

AS-REP::= [APPLICATION 11] KDC-REP

TGS-REP                 ::= [APPLICATION 13] KDC-REP

KDC-REP                 ::= SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (11 -- AS -- | 13 -- TGS --),
        padata          [2] SEQUENCE OF PA-DATA OPTIONAL
                        -- NOTE : non vide --,
        crealm          [3] Realm,      cname   [4] PrincipalName,
        ticket  [5] Ticket,
        enc-part [6] EncryptedData      -- EncASRepPart ou EncTGSRepPart, selon le cas approprié
}

EncASRepPart    ::= [APPLICATION 25] EncKDCRepPart

EncTGSRepPart           ::= [APPLICATION 26] EncKDCRepPart

EncKDCRepPart           ::= SEQUENCE {
        key             [0] EncryptionKey,
        last-req        [1] LastReq,
        nonce           [2] UInt32,
        key-expiration  [3] KerberosTime OPTIONAL,
        flags           [4] TicketFlags,
        authtime        [5] KerberosTime,
        starttime       [6] KerberosTime OPTIONAL,
        endtime         [7] KerberosTime,
        renew-till      [8] KerberosTime OPTIONAL,
        srealm          [9] Realm,
        sname           [10] PrincipalName,
        caddr           [11] HostAddresses OPTIONAL
}

LastReq                 ::=    SEQUENCE OF SEQUENCE {
        lr-type         [0] Int32,
        lr-value        [1] KerberosTime
}

AP-REQ                  ::= [APPLICATION 14] SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (14),
        ap-options      [2] APOptions,
        ticket          [3] Ticket,
        authenticator   [4] EncryptedData -- Authentifiant
}

APOptions               ::= KerberosFlags
        -- reserved(0),
        -- use-session-key(1),
```

```
        -- mutual-required(2)

-- Authentifiant non chiffré
Authenticator               ::= [APPLICATION 2] SEQUENCE  {
        authenticator-vno   [0] INTEGER (5),
        crealm              [1] Realm,
        cname               [2] PrincipalName,
        cksum               [3] Checksum OPTIONAL,
        cusec               [4] Microseconds,
        ctime               [5] KerberosTime,
        subkey              [6] EncryptionKey OPTIONAL,
        seq-number          [7] UInt32 OPTIONAL,
        authorization-data  [8] AuthorizationData OPTIONAL
}

AP-REP                      ::= [APPLICATION 15] SEQUENCE {
        pvno                [0] INTEGER (5),
        msg-type            [1] INTEGER (15),
        enc-part            [2] EncryptedData -- EncAPRepPart
}

EncAPRepPart                ::= [APPLICATION 27] SEQUENCE {
        ctime               [0] KerberosTime,
        cusec               [1] Microseconds,
        subkey              [2] EncryptionKey OPTIONAL,
        seq-number          [3] UInt32 OPTIONAL
}

KRB-SAFE                    ::= [APPLICATION 20] SEQUENCE {
        pvno                [0] INTEGER (5),
        msg-type            [1] INTEGER (20),
        safe-body           [2] KRB-SAFE-BODY,
        cksum               [3] Checksum
}

KRB-SAFE-BODY               ::= SEQUENCE {
        user-data           [0] OCTET STRING,
        horodatage          [1] KerberosTime OPTIONAL,
        usec                [2] Microseconds OPTIONAL,
        seq-number          [3] UInt32 OPTIONAL,
        s-address           [4] HostAddress,
        r-address           [5] HostAddress OPTIONAL
}

KRB-PRIV                    ::= [APPLICATION 21] SEQUENCE {
        pvno                [0] INTEGER (5),
        msg-type            [1] INTEGER (21),       -- NOTE : il n'y a pas d'étiquette [2]
        enc-part            [3] EncryptedData       -- EncKrbPrivPart
}

EncKrbPrivPart              ::= [APPLICATION 28] SEQUENCE {
        user-data           [0] OCTET STRING,
        horodatage          [1] KerberosTime OPTIONAL,
        usec                [2] Microseconds OPTIONAL,
        seq-number          [3] UInt32 OPTIONAL,
        s-address           [4] HostAddress             -- adresse de l'envoyeur --,
        r-address           [5] HostAddress OPTIONAL    -- adresse du receveur --,
}

KRB-CRED                    ::= [APPLICATION 22] SEQUENCE {
```

```
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (22),
        tickets         [2] SEQUENCE OF Ticket,
        enc-part        [3] EncryptedData -- EncKrbCredPart
}

EncKrbCredPart          ::= [APPLICATION 29] SEQUENCE {
        ticket-info     [0] SEQUENCE OF KrbCredInfo,
        nonce           [1] UInt32 OPTIONAL,
        horodatage      [2] KerberosTime OPTIONAL,
        usec            [3] Microseconds OPTIONAL,
        s-address       [4] HostAddress OPTIONAL,
        r-address       [5] HostAddress OPTIONAL
}

KrbCredInfo             ::= SEQUENCE {
        key             [0] EncryptionKey,
        prealm          [1] Realm OPTIONAL,
        pname           [2] PrincipalName OPTIONAL,
        flags           [3] TicketFlags OPTIONAL,
        authtime        [4] KerberosTime OPTIONAL,
        starttime       [5] KerberosTime OPTIONAL,
        endtime         [6] KerberosTime OPTIONAL,
        renew-till      [7] KerberosTime OPTIONAL,
        srealm          [8] Realm OPTIONAL,
        sname           [9] PrincipalName OPTIONAL,
        caddr           [10] HostAddresses OPTIONAL
}

KRB-ERROR               ::= [APPLICATION 30] SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (30),
        ctime           [2] KerberosTime OPTIONAL,
        cusec           [3] Microseconds OPTIONAL,
        stime           [4] KerberosTime,
        susec           [5] Microseconds,
        error-code      [6] Int32,
        crealm          [7] Realm OPTIONAL,
        cname           [8] PrincipalName OPTIONAL,
        realm           [9] Realm                   -- domaine de service --,
        sname           [10] PrincipalName          -- nom de service --,
        e-text          [11] KerberosString OPTIONAL,
        e-data          [12] OCTET STRING OPTIONAL
}

METHOD-DATA             ::= SEQUENCE OF PA-DATA

TYPED-DATA              ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
        data-type       [0] Int32,
        data-value      [1] OCTET STRING OPTIONAL
}

-- Suit le matériel de pré authentification

PA-ENC-TIMESTAMP        ::= EncryptedData -- PA-ENC-TS-ENC

PA-ENC-TS-ENC           ::= SEQUENCE {
        patimestamp     [0] KerberosTime            -- heure du client --,
        pausec          [1] Microseconds OPTIONAL
}
```

```
ETYPE-INFO-ENTRY      ::= SEQUENCE {
        etype         [0] Int32,
        salt          [1] OCTET STRING OPTIONAL
}

ETYPE-INFO            ::= SEQUENCE OF ETYPE-INFO-ENTRY

ETYPE-INFO2-ENTRY     ::= SEQUENCE {
        etype         [0] Int32,
        salt          [1] KerberosString OPTIONAL,
        s2kparams     [2] OCTET STRING OPTIONAL
}

ETYPE-INFO2           ::= SEQUENCE SIZE (1..MAX) OF ETYPE-INFO2-ENTRY

AD-IF-RELEVANT        ::= AuthorizationData

AD-KDCIssued          ::= SEQUENCE {
        ad-checksum   [0] Checksum,
        i-realm       [1] Realm OPTIONAL,
        i-sname       [2] PrincipalName OPTIONAL,
        elements      [3] AuthorizationData
}

AD-AND-OR             ::= SEQUENCE {
        condition-count [0] Int32,
        elements      [1] AuthorizationData
}

AD-MANDATORY-FOR-KDC    ::= AuthorizationData

END
```

Annexe B Changements depuis la RFC 1510

Le présent document remplace la RFC 1510 et clarifie la spécification d'éléments qui n'étaient pas complètement spécifiés. Lorsque des changements à des choix de mise en œuvre recommandés ont été faits, ou lorsque de nouvelles options sont ajoutées, ces changements sont décrits au sein du document et figurent dans la présente section. Plus significative, la "Spécification 2" à la Section 8 change les méthodes exigées de chiffrement et de somme de contrôle pour les aligner avec les meilleures pratiques courantes et déconseiller des méthodes qui ne sont plus considérées comme suffisamment fortes.

Un exposé a été ajouté à la Section 1 sur la validité de s'appuyer sur le KDC pour vérifier le champ de transit, et sur l'inclusion d'un fanion dans un ticket indiquant que cette vérification a été effectuée. C'est une nouvelle capacité qui n'était pas présente dans la RFC 1510. Les mises en œuvre préexistantes peuvent ignorer ou non l'établissement de ce fanion sans implications négatives sur la sécurité.

La définition de la clé secrète dit que dans le cas d'un usager, la clé peut être déduite d'un mot de passe. Dans la RFC 1510, il est dit que la clé est déduite du mot de passe. Ce changement a été fait pour s'accommoder de situations où la clé de l'usager pourrait être mémorisée sur une carte à mémoire, ou obtenue autrement, indépendamment du mot de passe.

L'introduction mentionne l'utilisation de la cryptographie à clé publique pour l'authentification initiale dans Kerberos par référence. La RFC 1510 n'incluait pas une telle référence.

Le paragraphe 1.3 a été ajouté pour expliquer qu'alors que Kerberos fournit l'authentification d'un principal désigné, il est toujours de la responsabilité de l'application de s'assurer que le nom authentifié est celui de l'entité avec laquelle l'application souhaite communiquer.

L'exposé sur l'extensibilité a été ajouté dans l'introduction.

L'exposé sur la façon dont l'extensibilité affecte les fanions de ticket et les options du KDC a été ajouté à l'introduction de la Section 2. Aucun changement n'a été fait aux options et fanions existants spécifiés dans la RFC 1510, bien que certains des paragraphes de la spécification aient été renumérotés, et que le texte ait été révisé pour rendre plus claire la description et l'intention des options existantes, en particulier par rapport à l'option ENC-TKT-IN-SKEY (maintenant au paragraphe 2.9.2) qui est utilisée pour l'authentification d'usager à usager. La nouvelle option et le fanion de ticket de vérification de politique de transit (paragraphe 2.7) ont été ajoutés.

Un avertissement concernant la génération de clés de session à l'usage des applications a été ajouté à la Section 3, insistant pour l'inclusion de l'entropie de la clé tirée de la clé de session générée par le KDC dans le ticket. Un exemple de l'utilisation de la sous-clé de session a été ajouté au paragraphe 3.2.6. Les descriptions des éléments de données de pré-authentification pa-etype-info, pa-etype-info2, et pa-pw-salt ont été ajoutées. La recommandation pour l'utilisation de la pré-authentification a été changée de "PEUT" en "DEVRAIT" et une note a été ajoutée concernant les attaques de texte en clair connues.

Dans la RFC 1510, la Section 4 décrivait la base de données dans le KDC. Cet exposé n'était pas nécessaire pour l'interopérabilité et constituait une contrainte non nécessaire pour la mise en œuvre. L'ancienne Section 4 a été retirée.

L'actuelle Section 4 était anciennement la Section 6 sur la spécification du chiffrement et des sommes de contrôle. La majeure partie de cette section a été mise à jour pour prendre en charge de nouvelles méthodes de chiffrement, et déplacée dans un document distinct. Les quelques aspects restants de la spécification du chiffrement et des sommes de contrôle spécifiques de Kerberos sont maintenant spécifiés dans la Section 4.

Des changements significatifs ont été faits à la disposition de la Section 5 pour préciser le comportement correct pour les champs facultatifs. Beaucoup de ces changements étaient nécessaires parce que une description impropre de l'ASN.1 dans la spécification Kerberos originale laissait le comportement correct mal spécifié. De plus, la formulation de cette section a été resserrée chaque fois que possible pour s'assurer que les mises en œuvre se conformant à la présente spécification seront extensibles à l'ajout des nouveaux champs dans les futures spécifications.

On a ajouté du texte pour décrire les problèmes de time_t=0 en ASN.1. Également pour préciser la question des mises en œuvre qui traitent les entiers facultatifs omis comme zéro. Également pour préciser le comportement de SEQUENCE ou SEQUENCE OF facultatif qui peut être vide. Une discussion a été ajoutée sur les numéros de séquence et le comportement de certaines mises en œuvre, y compris le comportement "zéro" et les nombres négatifs. Une note de compatibilité a été ajoutée au sujet de l'envoi inconditionnel de EncTGSRepPart sans considération du type de réponse qui l'incorpore. Des changements mineurs ont été faits à la description du type HostAddresses. Des contraintes ont été mises à des types d'entier. KerberosString a été défini comme une GeneralString (significativement) contrainte. KerberosFlags a été défini pour refléter le comportement des mises en œuvre existantes qui se distingue de la définition de la RFC 1510. Les fanions de ticket transited-policy-checked(12) et ok-as-delegate(13) ont été ajoutés. L'option de KDC disable-transited-check(26) a été ajoutée.

La description des PA-DATA couramment mis en œuvre a été ajoutée à la Section 5. La description de KRB-SAFE a été mise à jour pour noter le comportement de double codage des mises en œuvre existantes.

Il y avait deux définitions de METHOD-DATA dans la RFC 1510. La seconde, destinée à être utilisée avec KRB_AP_ERR_METHOD a été retirée, en laissant la définition de SEQUENCE OF PA-DATA.

La Section 7 de la RFC 1510, désignant les contraintes, a été passée en Section 6.

Quelques mots ont été ajoutés pour décrire la convention selon laquelle les noms de domaine fondés sur domaine pour les domaines nouvellement créés devraient être spécifiés en majuscules. Cette recommandation ne rend pas illégaux les noms de domaine en minuscules. Quelques mots ont été ajoutés pour souligner que les composants séparés par des barres obliques dans les noms de domaine de style X.500 sont cohérents avec les mises en œuvre existantes fondées sur la RFC 1510, mais entrent en conflit avec la recommandation générale de la représentation de nom X.500 spécifiée dans la RFC 2253.

La Section 8, transport réseau, constantes et valeurs définies, de la RFC 1510, est devenue la Section 7. Depuis la RFC 1510, la définition du transport TCP des messages Kerberos a été ajoutée, et les allocations de numéros de chiffrement et de somme de contrôle ont été déplacées dans un document distinct.

"Spécification 2" à la Section 8 du présent document change les méthodes exigées de chiffrement et de somme de contrôle pour les aligner avec les meilleurs pratiques actuelles et déconseiller les méthodes qui ne sont plus considérées comme suffisamment fortes.

Ajout de deux nouvelles sections, de considérations relatives à l'IANA et de considérations sur la sécurité.

Le pseudo-code a été retiré de l'appendice. Le pseudo-code était parfois mal interprété comme limitant les choix de mise en œuvre, et dans la RFC 1510, il n'était pas toujours cohérent avec les termes de la spécification. Des efforts ont été faits pour clarifier toutes les ambiguïtés de la spécification, plutôt que de s'appuyer sur le pseudo-code.

Un appendice a été ajouté, qui contient le module ASN.1 complet tiré de l'exposé de la Section 5 du document actuel.

FIN DES NOTES

(*TM) Projet Athena, Athena, et Kerberos sont des marques commerciales déposées de l'Institut de technologies du Massachusetts (MIT).

Références normatives

[RFC3961] Raeburn, K., "Spécifications de chiffrement et de somme de contrôle pour Kerberos 5", RFC 3961, février 2005.

[RFC3962] Raeburn, K., "Chiffrement avec la norme de chiffrement évolué (AES) pour Kerberos 5", RFC 3962, février 2005.

[ISO-646/ECMA-6] Organisation internationale de normalisation, "Ensemble de caractères codés sur 7 bits pour les échanges d'informations", ISO/CEI 646:1991.

[ISO-2022/ECMA-35] Organisation internationale de normalisation, "Structure de code de caractère et techniques d'extension", ISO/CEI 2022:1994.

[RFC1035] Mockapetris, P., "Noms de domaine – mise en œuvre et spécification", STD 13, RFC 1035, novembre 1987.

[RFC2119] Bradner, S., "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, RFC 2119, mars 1997.

[RFC2434] Narten, T. et H. Alvestrand, "Lignes directrices pour écrire une section Considérations relatives à l'IANA dans les RFC", BCP 26, RFC 2434, octobre 1998.

[RFC2782] Gulbrandsen, A., Vixie, P., et L. Esibov, "RR DNS pour spécifier la localisation des services (DNS SRV)", RFC 2782, février 2000.

[RFC2253] Wahl, M., Kille, S., et T. Howes, "Protocole léger d'accès à un répertoire (v3) : Représentation de chaîne UTF-8 des noms distinctifs", RFC 2253, décembre 1997.

[RFC3513] Hinden, R. et S. Deering, "Architecture d'adressage du protocole Internet Version 6 (IPv6)", RFC 3513, Avril 2003.

[X680] Recommandation UIT-T X.680, "Notation de syntaxe abstraite numéro un (ASN.1) : Spécification de la notation de base", (1997) Norme internationale ISO/CEI 8824-1:1998.

[X690] Recommandation UIT-T X.690, "Règles de codage ASN.1 : Spécification des règles de codage de base (BER), des règles de codage canonique (CER) et des règles de codage distinctives (DER), (1997) Norme internationale ISO/CEI 8825-1:1998.

Références informatives

[ISO-8859] Organisation internationale de normalisation, "Ensembles de caractères graphiques codés sur un seul octet de 8 bits – Alphabet latin", ISO/CEI 8859.

[RFC1964] Linn, J., "Mécanisme GSS-API de Kerberos Version 5", RFC 1964, juin 1996.

[DGT96] Don Davis, Daniel Geer, et Theodore Ts'o, "Kerberos et la dérive d'horloge : histoire, protocoles, et mise en œuvre", USENIX Computing Systems 9:1, janvier 1996.

[DS81] Dorothy E. Denning et Giovanni Maria Sacco, "Horodatages dans les protocoles de distribution de clés,"Communications de l'ACM, Vol. 24 (8), p. 533-536, août 1981.

[KNT94] John T. Kohl, B. Clifford Neuman, et Theodore Y. Ts'o, "L'évolution du système d'authentification Kerberos". In Distributed Open Systems, pages 78-94. IEEE Computer Society Press, 1994.

[MNSS87] S. P. Miller, B. C. Neuman, J. I. Schiller, et J. H. Saltzer, Section E.2.1: Authentification Kerberos et système d'autorisation, M.I.T. Projet Athéna, Cambridge, Massachusetts, décembre 21, 1987.

[NS78] Roger M. Needham et Michael D. Schroeder, "Utilisation du chiffrement pour l'authentification dans les grands réseaux d'ordinateurs," Communications de l'ACM, Vol. 21 (12), pp. 993-999, décembre 1978.

[Neu93] B. Clifford Neuman, "Autorisation fondée sur un mandataire et comptabilité pour les systèmes répartis," in Proceedings of the 13th International Conference on Distributed Computing Systems, Pittsburgh, PA, mai 1993.

[NT94] B. Clifford Neuman et Theodore Y. Ts'o, "Service d'authentification pour les réseaux d'ordinateurs," IEEE Communications Magazine, Vol. 32 (9), p. 33-38, septembre 1994.

[Pat92] J. Pato, "Utilisation de la pré-authentification pour éviter les attaques de mot de passe deviné", Open Software Foundation DCE, Request for Comments 26 (décembre 1992).

[RFC1510] Kohl, J. et C. Neuman, "Service Kerberos d'authentification de réseau (V5)", RFC 1510, septembre 1993.

[RFC4086] Eastlake, D., 3rd, Schiller, J., et S. Crocker, "Exigences d'imprévisibilité pour la sécurité", BCP 106, RFC 4086, juin 2005.

[SNS88] J. G. Steiner, B. C. Neuman, et J. I. Schiller, "Kerberos : un service d'authentification pour les systèmes de réseaux ouverts," p. 191-202, Usenix Conference Proceedings, Dallas, Texas, février 1988.

[RFC4121] Zhu, L., Jaganathan, K., et S. Hartman, "Mécanisme Kerberos Version 5 d'interface de programme d'application pour service générique de sécurité (GSS-API) : Version 2", RFC 4121, juillet 2005.

Adresse des auteurs

Clifford Neuman
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, USA
mél : bcn@isi.edu

Tom Yu
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139, USA
mél : tlyu@mit.edu

Kenneth Raeburn
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139, USA
mél : raeburn@mit.edu

Sam Hartman
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139, USA
mél : hartmans-ietf@mit.edu

Remerciement

Le financement de la fonction d'édition des RFC est actuellement fourni par Internet Society.