rfc4120

Le protocole Kerberos v5

Le protocole Kerberos

Kerberos fournit un moyen de vérifier les identités des principaux sur un réseau non-sûr. Ceci est fait sans s'appuyer sur la confiance des assertions par le système hôte, sans confiance basé sur les adresses de l'hôte, sans nécessiter de sécurité physique des hôtes sur le réseau, et en assumant qui les paquets sur le réseau peuvent être lus, modifiés et insérés. Kerberos effectue une authentification dans ces conditions en tant que service d'authentification tiers en utilisant la cryptographie conventionnelle. Les extensions Kerberos peuvent permettre l'utilisation de chiffrements à clé publique durant certaines phases de l'authentification.

Le processus d'authentification de base de Kerberos se déroule comme suit : Un client envoie une demande au serveur d'authentification (AS) pour les accréditations pour un serveur donné. l'AS répond avec ces accréditations, chiffrés avec la clé du client. les accréditations consistent d'un ticket pour le serveur et une clé de chiffrement temporaire (souvent appelé un clé de session). Le client transmet le ticket (qui contient l'identité du client et une copie de la clé de session, le tout chiffré avec la clé du serveur) au serveur. La clé de session (maintenant partagée par le client et le serveur) est utilisé pour authentifier le client et peut optionnellement être utilisé pour authentifier le serveur. Il peut être utilisé également pour chiffrer d'autres communications entre les 2 parties ou pour échanger une clé de sous-session séparée à utiliser pour chiffrer d'autres communications.

L'implémentation du protocole de base consiste d'un ou plusieurs serveurs d'authentification fonctionnant sur un hôte sécurisé. Les serveurs d'authentification maintiennent une base de données de principaux et leur clés secrète. Pour ajouter l'authentification à ses transactions, une application réseaux ajoute des appels à la librairie Kerberos directement ou via GSS-API. Ces appels résultent en la transmission des messages nécessaires pour accomplir l'authentification.

Le protocole de base consiste de nombreux sous-protocoles (ou échanges). Il y a 2 méthodes de base pour lequel un client peut demander des accréditations. Dans la première approche, le client envoie une requête en texte clair pour le serveur désiré au serveur AS. La réponse est envoyée chiffrée avec la clé secrète du client. Généralement cette requête est pour un TGT (Ticket-Granting Ticket), qui peut être utilisé ultérieurement avec le TGS (Ticket-Granting Server). Dans la seconde méthode, le client envoie une requête au TGS. Le client utilise de TGT pour s'authentifier lui-même auprès du TGS de la même manière que s'il avait contacté un serveur d'application qui nécessitait une authentification Kerberos. La réponse est chiffrée dans la clé de session du TGT. Bien que la spécification du protocole décrive l'AS et le TGT comme des serveurs séparés, en pratique il sont souvent implémentés comme points d'entrée différents du protocole dans seul serveur Kerberos.

Une fois obtenues, les accréditations 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 est libre de choisir qu'elle protection est nécessaire.

Pour vérifier les identités des principaux dans une transaction, le client transmet le ticket au serveur d'application. Parce que le ticket est envoyé en clair (de partie sont chiffrés, mais ce chiffrement ne déjoue pas la répétition) et peut être intercepté et réutilisé par un attaquant, des informations additionnelles 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 inclus un horodatage. l'horodatage prouve que le message a été récemment envoyés et n'est pas rejouée. Chiffrer l'authentifiant dans la clé de session prouve qu'il a été généré par un partie possédant la clé de session. Vu que personne excepté le principal et le serveur ne connaît la clé de session (qui n'est jamais envoyée sur le réseau en clair), cela garantis l'identité du client.

L'intégrité des messages échangés entre les principaux peut également être garantis en utilisant la clé de session (passée dans le ticket et contenus dans les accréditations). Cette approche fournis la détection contre les attaques à répétition et les attaques par modification de flux. C'est accomplis en générant et en transmettant un checksum à l'épreuve des collisions du message du client, entrée avec la clé de session. La confidentialité et l'intégrité des messages échangés entre les principaux peuvent êrte sécurisé en chiffrant le données en utilisant la clé de session contenus dans le ticket ou la clé de sous-session trouvée dans l'authentifiant.

les échanges d'authentification mentionnés plus haut nécessitent un accès lecture seule à la base de données Kerberos. Parfois, cependant, les entrées dans la base de données doit être modifiée, par exemple pour ajouter des principaux ou changer la clé d'un principal. Pour cela on utilise un protocole entre un client et un serveur Kerberos tiers, le serveur d'administration Kerberos (KADM). Il y a aussi un protocole pour maintenir plusieurs copies de la base Kerberos. Aucun de ces protocoles n'est décris dans ce document.

Fonctionnement inter-domaine

Le protocole Kerberos est conçus pour opérer au delà des limites organisationnelle. Un client dans une organisation peut être authentifié pour un serveur dans une autre organisation. Chaque organisation souhaitant utiliser un serveur Kerberos établis son propre "domaine" (realm). Le nom de ce domaine dans lequel un client est enregistré fait partie du nom du client et peut être utilisé par le service final pour décider d'honorer ou non une requête.

En établissant des clé inter-domaine, les administrateurs de 2 domaines peuvent permettre à un client authentifié dans le domaine local de prouver son identité aux serveurs dans les autres domaines. Les échanges de clé inter-domaines (une clé séparée peut être utilisé pour chaque direction) enregistrent le service d'allocation de ticket de chaque domaine comme un principale dans l'autre domaine. Un client est ainsi capable d'obtenir un TGT pour le service d'allocation de ticket du royaume distant depuis le domaine local. Quand ce TGT est utilisé, le service d'allocation de ticket distant utilise la clé inter-domaine (qui généralement diffère de sa propre clé TGS normale) pour déchiffrer le TGT ; ainsi il est certain que le ticket a été fournis par le TGS du client. Les tickets fournis par le service d'allocation de ticket distant va indiquer au service final que le client a été authentifié depuis un autre domaine.

Sans opération inter-domaine et avec les permissions appropriées, le client peut l'enregistrement d'un principal nommé séparément dans en royaume distant et engager un échange normal avec ce domaine. Cependant, même pour un petit nombre de clients, cela devient lourd à gérer, et méthodes plus automatiques sont nécessaires.

Un domaine est dit communiquer avec un autre si les 2 domaines partagent un clé inter-domaine, ou si le domaine local partage une clé inter-domaine avec un domaine intermédiaire qui communique avec le domaine distant. Un chemin d'authentification est la séquence de domaines intermédiaire qui servent de transit d'un domaine à un autre.

Les domaines peuvent être organisés hiérarchiquement. Chaque domaine partage une clé avec son parent et une clé différente avec chaque enfant. Si une clé inter-domaine n'est pas directement partagée par 2 domaines, l'organisation hiérarchique permet de construire facilement un chemin d'authentification. Si une organisation hiérarchique n'est pas utilisée, il peut être nécessaire de consulter une base de données pour construire un chemin d'authentification entre les domaines.

Bien que les domaines sont typiquement hiérarchiques, les domaines intermédiaire peuvent être bypassés pour effectuer une authentification inter-domain au travers d'un chemin alternatif. Il est important pour le service final de connaître les domaines traversés. Pour simplifier cette décision, un champ dans chaque ticket contient les noms des domaines qui ont servis dans l'authentification du client.

Le serveur d'application est responsable au final d'accepter ou non l'authentification et devrait vérifier le champ de transit. Le serveur d'application peut choisir de s'appuyer sur le KDC pour la vérification de ce champ. Le KDC du serveur d'application va mettre le flag **TRANSITED-POLICY-CHECKED** dans ce cas. Les KDC des domaines intermédiaire peuvent également vérifier ce champ vu qu'ils fournissent des TGT pour d'autre domaines, mais ils sont encouragés à ne pas le faire. Un client peut demander que les KDC ne vérifient pas ce champ en mettant le flag **DISABLE-TRANSITED-CHECK**. Les KDC devraient honorer ce flag.

Choisir un principal avec lequel communiquer

Le protocole Kerberos fournis un moyen de vérifier que l'entité avec laquelle il communique est la même que celle enregistrée avec le KDC en utilisant l'identité revendiquée (le principal). Il est nécessaire de déterminer que cette identité correspond à l'entité avec laquelle on tente de communiquer.

Quand des données appropriées ont été échangées en avance, l'application peut effectuer cette détermination syntaxiquement basé sur la spécification du protocole de l'application, les informations fournies par l'utilisateur, et les fichiers de configuration. Par exemple, le nom principal du serveur (incluant le domaine) pour un serveur telnet peut être dérivé du nom d'hôte spécifié par l'utilisateur (depuis la ligne de

commande), le préfixe "host/" spécifié dans la spécification du protocole de l'application, et un mappage à un domaine Kerberos dérivé syntaxiquement depuis la partie domaine du nom d'hôte et des informations spécifiées depuis la base de domaines Kerberos locale.

On peut également s'appuyer sur les tiers de confiance qui font cette détermination, mais seulement quand les données obtenues depuis un tiers sont convenablement protégées en intégrité lorsqu'elles résident sur le serveur du tiers de confiance et lors de leur transmission. Par exemple, on ne devrait pas pas se fier à un enregistrement DNS non protégé pour

Les implémentations de Kerberos et les protocoles basés sur Kerberos ne doivent pas utiliser de requêtes DNS non sécurisés pour canoniser les composants du nom d'hôte des noms de principal de service. Dans un environnement sans service de nom sécurisé, les auteurs d'application peuvent ajouter un nom de domaine configuré statiquement pour les noms d'hôtes non qualifiés avant de passer ce nom aux mécanismes de sécurité. Les facilités de service de nom sécurisés, si disponible, peuvent être trustés pour la canonisation de nom d'hôte, mais une telle canonisation par le client ne devrait pas être requise par les implémentations du KDC.

Authorisation

En tant que service d'authentification, Kerberos fournis un moyen de vérifier l'identité des principaux sur un réseau. L'authentification est généralement la première étape dans le processus d'autorisation, déterminant si un client peut utiliser un service, à quels objets le client à accès, et le type d'accès permis pour chacun. Kerberos ne fournis pas par lui-même d'autorisation. La possession d'un ticket client pour un service fournis uniquement l'authentification de ce client pour ce service, et en l'absence de procédure d'autorisation, une application ne devrait pas lui autoriser l'utilisation de ce service.

Des méthodes d'autorisation séparés peuvent être implémentés comme fonctions de contrôle d'accès spécifique à l'application et peut utiliser des fichiers sur le serveur d'application, sur des accréditations d'autorisation fournis séparément tels que ceux fournis dans les proxys. Ces accréditations peuvent être embarqués dans une donnée d'authentification du ticket lorsqu'il est encapsulé par l'élément de données d'autorisation produit par le KDC.

Les applications ne devraient pas accepter la délivrance 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.

Étendre Kerberos sans rupture d'intéropérabilité

Avec la base d'implémentation de Kerberos déployée grandissante, étendre Kerberos devient de plus en plus important. Malheureusement, certaines extensions du protocole existant créent des problèmes d'intéropérabilité à cause de l'incertitude au regard du traitement de certaines options d'extentions par certaines implémentations.

Kerberos fournis un mécanisme général pour l'extensibilité du protocole. Certains messages du protocole contiennent des trous typés – des sous-messages qui contiennent une chaîne d'octet et un entier qui définis comment interpréter cette chaîne d'octet. Les types entier sont enregistrés centralement, mais ils peuvent être utilisés par les vendeurs d'extension et pour les extensions standardisés.

Dans ce document, le mot extension réfère à une extension en définissant un nouveau type à insérer dans un trou typé existant dans un message du protocole. Il ne réfère pas à l'extension en ajoutant de nouveaux champs ASN.1, sauf mention.

Envoyer des messages extensibles

If faut s'assurer que les anciennes implémentations peuvent comprendre les messages envoyés, même si elles ne comprennent pas une extension utilisée. À moins que l'emeteur sache qu'une extension est supportée, l'extension ne peut pas changer les sémantiques du coeur

du message ou des extensions définies précédemment.

Par exemple, une extension incluant des information de clé nécessaire à déchiffrer la partie chiffrée d'un KDC-REP pourrait seulement être utilisé dans les situations où le receveur est connus pour supporter l'extension. Donc en définissant de tels extensions il est important de fournir une manière pour le 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 le 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.

Hypothèse sur l'environnement

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

- Les attaques DOS ne sont pas résolues avec Kerberos. Il y a des endroits dans le protocoles où un intrus peut empêcher une application de participer aux étapes d'authentification appropriées.
- Les principaux doivent garder secrètes leurs clés secrète. Si un intrus s'empare d'une manière un d'une autre de la clé d'un principal, il sera capable de se faire passer pour ce principal ou de se déguiser en n'importe quel serveur du principal légitime.
- Les attaques en devinant le mot de passe 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.
- Chaque hôte sur le réseaux doit avoir une horloge synchronisée 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. Si les horloges sont synchronisées sur le réseaux, le protocole doit lui-même être synchronisé. Le degré d'approximation normal est de l'ordre de 5 minutes
- Les identifiants de principal ne sont pas recyclés à court terme. Un contrôle de mode d'accès normal utilise des 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 ce problème en ne réutilisant pas les identifiants de principal.

Glossaire

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 chiffrement. Le déchiffrement transforme le texte chiffré en texte 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.

Utilisation et demandes de flags de ticket

Chaque ticket Kerberos contient un ensemble de flags utilisés pour indiquer les attributs de ce ticket. La plupart des flags peuvent être demandés par un client lorsque le ticket est obtenu ; certains sont automatiquement activés et désactivés en fonction des besoins du serveur. À l'exception du flag **INVALID**, les clients doivent ignorer les flags qu'ils ne reconnaissent pas. Les KDC doivent ignorer les options KDC qui ne sont pas reconnus. Certaines mises en œuvre de la rfc1510 sont connues pour rejeter les options KDC si la demande a été rejetée alors qu'elle avait été envoyée avec des options ajoutées depuis la rfc1510. Comme les nouveaux KDC ignorent les options inconnues, les clients doivent confirmer que le ticket retourné satisfait leur 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 soit retournée.

Ticket initial, pré-authentification, et matériel authentifié

Le flag **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 flag 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 flags **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 (dans ce cas le flag **INITIAL** est établi) ou qu'il soit produit sur la base d'un TGT (le flag **INITIAL** n'est pas mis, mais **PRE-AUTHENT** et **HW-AUTHENT** sont ramenés du TGT).

Tickets invalides

Le flag **INVALID** indique qu'un ticket est invalide. Les serveur d'application doivent rejeter les tickets qui ont ce fjag. Un ticket postdaté sera toujours produit sous 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** spécifié. Le KDC ne validera le ticket qu'après l'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)

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. 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 2 heures 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 présenter au KDC un ticket renouvelable, avec l'option **RENEW** dans la demande.

Le KDC va produire un nouveau ticket avec une nouvelle clé de session et une nouvelle heure d'expiration. 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.

Le flag **RENEWABLE** dans un ticket n'est normalement interprété que par le service d'allocation de tickets. 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 flag **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. S'il est établi, le champ renew-till dans le ticket contient l'heure après laquelle le ticket ne peut plus être renouvelé.

Tickets postdatés

Les applications peuvent parfois 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. 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.

Le flag **MAY-POSTDATE** 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 flag 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** dans le KRB_AS_REQ. Ce flag 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 KRB_AS_REQ. La durée de vie d'un ticket postdaté sera la durée de vie restante du TGT au moment de la demande. Sauf si l'option **RENEWABLE** est aussi établie, auquel cas elle peut être la durée de vie complète du TGT. Le KDC peut limiter la durée d'un ticket postdaté.

Le flag **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 ces tickets, ou 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.

Tickets mandatables et mandataires

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 flags proxy et proxiable est utilisé pour fournir des accéditifs 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 paragraphes suivant pour transmettre un TGT.

Le flag **PROXIABLE** 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 flag dit au TGS qu'il est d'accord pour produire un nouveau ticket (mais pas un TGT) avec une adresse réseau différente fondée sur ce ticket. Ce flag 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 gu'il soit rétabli lorsqu'il demande tout autre ticket.

Ce flag permet à un client de passer au serveur proxy 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 le mandat est produit pour être utilisé sur toute adresse.

Le flag **PROXY** est établi dans un ticket par le TGS lorsqu'il produit un ticket proxy. Les serveurs d'application peuvent vérifier ce flag ; et optionnellement peuvent demander une authentification supplémentaire de la part de l'agent qui présente de mandat afin de fournir une trace d'audit.

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. Exemple : 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 flag **FORWARDABLE** 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 flag **FORWARDABLE** a une interprétation similaire à **PROXIABLE**, sauf que les TGT peuvent aussi être produits avec des adresses réseau différentes. Ce flag est ré-initialisé par défaut, mais les utilisateurs peuvent demander qu'il soit établi en mettant l'option **FORWARDABLE** dans la demande d'AS lorsqu'ils demandent le TGT initial.

Le flag **FORWARDED** est mis par le TGS lorsqu'un client présente un ticket avec le flag **FORWARDABLE** mis 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 flag **FORWARDED** mis. Les serveurs d'application peuvent choisir de traiter les tickets **FORWARDED** différemment.

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.

Vérification des stratégies 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 ?DC 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 l'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 flag **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 pars le champ en mettant le flag **DISABLE-TRANSITED-CHECK**. Les KDC devraient respecter ce flag.

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** sois mis.

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 flag **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 flags de ticket dans la partie chiffrée de la réponse du KDC peut avoir le flag **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 flag 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 flag. Lorsqu'il établit ce flag, 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.

Autres options de KDC

Il y a 3 options supplémentaires qui peuvent être établies dans une demandes de KDC du client

Renewable-OK

L'option **REMEWABLE-OK** indisque 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 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.

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 TGT. 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.

Authentification hardware 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 l'AS. S'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.

Échange de messages

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

Échange service d'authentification

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 la forme de base, la clé secrète du client est utilisée pour le chiffrement et le déchiffrement. Cet échange est normalement utilisé à l'initialisation d'une session de connexion pour obtenir des accréditifs pour un serveur d'allocation de tickets (TGS). qui seront utilisés ultérieurement pour obtenir des accréditifs pour d'autres serveurs sans 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 TGS (le service de changement de mot de passe refuse les demandes si le demanteur ne peut pas démontrer qu'il a connaissance du vieux mot de passe).

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 2 messages : **KRB_AS_REQ** et **KRB_AS_REP** ou **KRB_ERROR**.

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 temporaire généré de façon aléatoire qui peut être utilisé pour détecter des répétitions et pour associer les réponses aux demandes correspondantes. Ce nom temporaire doit être généré de façon aléatoire par le client et mémorisé pour vérification par rapport au nom aléatoire 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

supplémentaire sont chiffrées avec la clé secrète du client. La partie ciffrée du message **KRB_AS_REP** contient aussi le nom temporaire qui doit correspondre au nom temporaire provenant du message **KRB_AS_REQ**.

Sans pré-authentification, l'AS 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 puisque personne sauf le principal dont l'identité a été donnée dans la demande se sera capable d'utiliser la réponse. Ses information 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 nécessaire pour l'échange initial. Ce champ devrait être utilisé pour la pré-authentification.

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. Ce message 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, 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.

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 ; s'il devrait être postdaté ou permettre de 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.

Réception du message KRB_AS_REQ

Si tout va bien, le traitement du message **KRB_AS_REP** résultera en la création d'un ticket que le client présentera au serveur.

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.

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**.

S'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 avec le code **KDC_ERR_PREAUTH_FAILES**. 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 inclure habituellement les éléments **PA-ETYPE-INFO** et/ou **PA-ETYPE-INFO2**. 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 s'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.

En réponse à une demande d'AS, s'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. S'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 de l'utilisateur est générée à partir d'un mot de passe ou d'une passphrase, la fonction string-to-key pour le type de clé de chiffrement particulier est utilisé, comme spécifié dans la rfc 3961. La valeur de salt et les paramètres supplémentaires pour la fonction string-to-key ont des valeurs par défaut qui peuvent être remplacés par les données de pré-authentification (**n (PA-PW-SALT, PA-AFS3-SALT, PA-ETYPE-INFO, PA-ETYPE-INFO2, etc.**). Comme le KDC est supposé mémoriser une copie seulement de la clé résultant, ces valeurs ne devraient pas être changées pour des clés fondées dus 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 un **KDC-ERROR** ou dans un **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** ne comporte aucun type de chiffrement plus récent), il doit envoyer les deux **PA-ETYPE-INFO2** et **PA-ETYPE-INFO** (tous 2 avec une entrée pour chaque enctype). Un enctype "plus récent" est tout enctype spécifié officiellement en premier concurrement ou ultérieurement à la publication de la présent 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 utilisateur de façon fiable selon une passphrase donnée sans contacter le KDC, car on ne pourra pas savoir si des valeurs de salt 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 une fenêtre acceptable pour le KDC et si l'option **POSTDATE** n'a pas été spécifié, l'heure de début du ticket est réglée à l'heure en cours du serveur d'authentification. S'il indique une heure future au delà de la fenêtre acceptable, mais si l'option **POSTDATED** n'a pas été spécifiée, l'erreur **KDC_ERR_CANNOT_PORTDATE** est retournée. Autrement, l'heure de début demandée est vérifiée par rapport à la politique du domaine local, et si l'heure de début du ticket est acceptable, il est réglé comme demandé, et le flag **INVALID** est mis dans le nouveau ticket.

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 demandée dans le message **KRB_AS_REQ**s
- L'heure de début du ticket plus la durée de vie maximum admissible associée au principal 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
- L'heure de début du ticket plus la durée de vie maximum réglée par la stratégie du domaine local

Si l'heure d'expiration demandée moins l'heure de début est inférieur 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 demandé, le flag **RENEWABLE** est alors mis dans le nouveau ticket, et le champ renew-till 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 plus la durée de vie renouvelable maximum réglée par la politique du domaine local

Le champ des flags du nouveau ticket aura les options suivantes établies s'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, son flag **INVALID** est également mis.

Si tout ce qui précède réussit, le serveur va chiffrer la partie de ciphertext du ticket en utilisant la clé de chiffrement extraite de l'enregistrement du principal du serveur dans la base de données Kerberos en utilisant le type de chiffrement associé à la clé du principal du serveur (ce choix n'est pas affecté par le etype dans la requête). Il forge alors un message **KDC_AS_REP**, 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 ciphertext 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.

Génération du message KDC_ERROR

Plusieurs erreurs peuvent survenir, et le serveur d'authentification répond en envoyant un message d'erreur, **KDC_ERROR**, au client, avec les champs code d'erreur et e-text réglés aux valeurs appropriées.

Réception du message KRB_AS_REP

Si le type du message de réponse est **KDC_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 temporaire 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 des valeurs attendues), et que le champ d'adresse d'hôte est correcte. Il mémorise alors le ticket, la clé de sessions, 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'utilisateur 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** (en comparant le nom temporaire avec celui envoyé), le client sait que l'heure courante du KDC est celle lue dans le champ authtime de la partie chiffrée de la réponse. Le client peut optionnellement utiliser cette valeur pour synchroniser l'horloge dans les messages suivant en enregistrant avec le ticket la différence entre la valeur de authtime et l'horloge locale. Cet offset peut ainsi être utilisé par le même utilisateur pour ajuster le temps lu de l'horloge système lors de la génération des messages.

Cette technique doit être utilisée en ajustant 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'utilisateur dans la clé secrète a été utilisée, mais pas auprès du système ou de la workstation.

Le déchiffrement correcte du message **KRB_AS_REP** n'est pas suffisant à l'hôte peut vérifier l'identité de l'utilisateur qui se connecte sur la workstation. L'utilisateur 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'utilisateur, il doit exiger que l'utilisateur 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.

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.

Échange d'authentification client/serveur

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.

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 supplémentaire. Le ticket est insuffisant par lui-même pour authentifier un client, car les tickets passent à travers le réseau. 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".

Génération d'un message KRB_AP_REQ

Lorsqu'un client souhaite initier l'authentification auprès d'un serveur, il obtient (soit avec un cache d'arréditifs, 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 rejetés s'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 en mettant le ou les flags 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 information supplémentaires spécifiques de l'application.

Réception du message KRB_AP_REQ

L'authentification se fonde sur l'heure courante du serveur, 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.

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 flag **USE-SESSION-KEY** est mis dans le champ **ap-options**, il indique au serveur l'utilisation de l'authentification d'utilisateur à utilisateur. et le chiffrement du ticket avec la clé de session provenant du TGT du serveur plutôt qu'avec la clé secrète du serveur.

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. S'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 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 au delà de l'admissible, l'erreur **KRB_AP_ERR_SKEW** est retournée.

À moins que le serveur d'application ne fournisse ses propres moyens pour se protéger contre la répétition (par exemple, une séquence challenge-réponse initiée par le serveur après l'authentification, ou l'utilisation d'une sous-clé de chiffrement générée par le serveur), le serveur doit utiliser un cache pour mémoriser tout authentifiant présenté dans la plage de temps admissible. Une analyse soigneuse du protocole et de la mise en œuvre de l'application est recommandée avant d'éliminer ce cache. Le cache mémorise au moins les champ de nom du serveur, de nom du client, et d'heures tirés des authentifiants vus le plus récemment, et si une correspondance est trouvée, 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 du serveur ne devraient pas voir leurs authentifiants rejetés si les champ heure et microseconde se trouvent correspondre à d'autres authentifiants du client.

Si un serveur perd la trace des authentifiants présentés pendant l'horaire admissible, il doit rejeter toutes les demandes jusqu'à la fin de la plage de temps, lui donnant l'assurance que tout authentifiant perdu ou répété se retrouvera hors plage.

Note de mise en œuvre : Si un client génère plusieurs demandes au KDC avec le même horodatage, y compris le champ timestamp, 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 timestamp dans l'horodatage lorsque l'horloge retourne la même heure pour plusieurs demandes.

Si plusieurs serveur (par exemple, différents services sur une machine, ou un seul service mis en œuvre sur plusieurs machines) partagent un principal de service (pratique non recommandée en général), ils doivent partager ce cache, 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 contre la répétition, un authentifiant utilisé avec un tel service pourrait être répété ultérieurement dans un service différent avec le même principal de service mais sans protection contre la répétition.

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 plus tard ou l'utilise pour aider à générer son propre choix d'une sous-clé à retourner dans un message **KRB_AP_REP**.

Le serveur calcule l'âge 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 la plage horaire admissible, ou si le flag **INVALID** est mis, 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 la plage admissible, l'erreur **KRB_AP_ERR_TKT_EXPIRED** est retournée.

Si toutes ces vérification ont réussi sans erreur, le serveur est assuré 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 gérer l'autorisation 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.

Génération d'un message KRB_AP_REP

Normalement, la demande l'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 doivent 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.

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. S'ils correspondent, le client est alors assuré que le serveur est authentique. Le numéro de séquence est la sous-clé (si 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 présent dans l'authentification).

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 utilisation 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 aléatoire chez le client, il est vivement recommandé que toute clé déduite par une application pour utilisation ultérieure incluent 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 eut 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**. Le message **KRB_SAFE** peut être utilisé pour s'assurer de l'intégrité.

Échange TGS

L'échange TGS entre un client et le TGS est initié par un client lorsqu'il cherche à obtenir des accréditifs d'authentification pour un

serveur donné (qui peut être enregistré dans un domaine distant), lorsqu'il cherche à renouveler ou valider un ticket existant, ou lorsqu'il cherche à obtenir un ticket mandataire. Dans le premier cas, le client doit déjà avoir acquis un ticket pour le TGS auprès de l'AS (Le TGT est généralement obtenu quand un client quand un client s'authentifie sur le système). Le format pour l'échange TGS est presque identique à l'échange AS. La différence principale est que le chiffrement et le déchiffrement dans l'échange TGS n'est pas effectuée dans le clé du client. À la place, le clé de session du TGT ou du ticket renouvelable, ou de la clé de sous-session d'un authentifiant est utilisé. Comme c'est le cas pour tous les serveur d'application, les tickets expirés ne sont pas acceptés par le TGS, donc une fois une TGT ou un renouvelable expiré, le client doit utiliser un échange séparé pour obtenir des tickets valides.

L'échange TGS consiste de 2 messages : une requête (**KRB_TGS_REQ**), et une réponse (**KRB_TGS_REP** ou **KRB_ERROR**). Le message **KRB_TGS_REQ** inclus les informations authentifiant le client plus une requête pour des accréditifs. Les information d'authentification consistent de l'en-tête d'authentification (**KRB_AP REQ**), qui inclus le ticket TGT, renouvelable ou invalide du client précédemment obtenus. Dans les cas de TGT et de proxy, la requête peut inclure un ou plusieurs éléments : une liste d'adresses réseaux, une collection de données d'autorisation typées, ou des tickets supplémentaires. La réponse TGS (**KRB_TGS_REP**) contient es 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. 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.

Génération du message KRB_TGS_REQ

Avant d'envoyer une demande au TGS, 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, peut être mémorisé dans un serveur de noms, ou peut être obtenu d'un fichier de configuration. Si le client connaît le nom et le domaine du principal du service et s'il ne possède pas encore un TGT pour le domaine approprié, il doit alors en obtenir un. Il 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 récurrent 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. Noter que dans ces cas, la mauvaise configuration du serveur Kerberos peut causer les 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 les 2 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 cache ce ticket et le réutiliser dans des échange **KRB_TGS** futures 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 s'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 serveur 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 de DOS 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'options **ENC-TKT-IN-SKEY** utilisée pour l'authentification d'utilisateur à utilisateur. 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 même champs qu'utilisés dans le message **KRB_AS_REQ** avec plusieurs champs facultatifs : le champ **enc-authorization-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 avec 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.

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 vérification 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 TGS, 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, pus 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'utilisateur 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 **KRB_AP_ERR_MODIFIED**) ou si la somme de contrôle n'est pas à l'épreuve des collisions (avec un code d'erreur **KRB_AP_ERR_INAPP_CKSUM**). Si le type de somme de contrôle n'est pas accepté, l'erreur **KDC_ERR_SUMTYPE_NOSUPP** est retournées.

Si un des déchiffrements indique un échec de vérification d'intégrité, l'erreur **KRB_AP_ERR_BAD_INTEGRITY** est retournée. Le KDC doit envoyer un message **KRB_TGS_REP** valide s'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, de KDC devrait alors retourner **KRB_AP_ERR_REPEAT**.

Génération du message KRB_TGS_REP

Le message **KRB_TGS_REP** partage son format avec **KRP_AS_REP**, mais avec son champ de typee réglé à **KRB_TGS_REP**.

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 s'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 besoins d'être mis à jour, mais que le type de transit n'est par 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'heure de début du TGT plus le minimum de la durée de vie maximum pour le serveur '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 flag **FORWARDABLE** est mis 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 flag **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 plage horaire admissible pour le KDC et si l'option **POSTDATE** n'est pas spécifiée, l'heure de début du ticket est réglée à l'heure en cours de l'AS. Si elle indique une heure dans le futur au delà de la plage horaire admissible, mais l'option **POSTDATE** n'est pas spécifiée ou si le flag **MAY-POSTDATE** n'est pas mis dans le TGT, l'erreur **KDC_ERR_CANNOT_POSTDATE** est alors retournée. Sinon, si le TGT a le flag **MAY-POSTDATE** mis, 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 flag **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, de fin, ou de fin de **renew-till** d'un ticket postdaté nouvellement produit ne peut être étendue au delà de l'heure de **renew-till** du TGT.

Si l'option **ENC-TGT-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'utilisateur à utilisateur pour prouver son identité à un serveur qui n'a pas d'accès à une clé persistante. 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 de ce KDC, et (c) l'option **RENEW** est demandé, donc le KDC va vérifier que le flag **RENEWABLE** est mis dans le ticket, que le flag **INVALID** n'est pas mis, et que le temp **renew-till** est dans le future. Si l'option **VALIDATE** est requis, le KDC va vérifier que le starttime a passé et que le flag **INVALID** est mis. Si l'option **PROXY** est demandé, le KDC va vérifier que le flag **PROXIABLE** est mis dans le ticket. Si le test réussis et que le ticket passe toutes les vérifications décris dans la prochaine section, le KDC va fournir le nouveau ticket approprié.

La parie ciphertext de la réponse dans le message **KRB_TGS_REP** est chiffrée dans la clé de sous-session de l'authentifiant, si présent, ou dans la clé de session du TGT. Elle n'est pas chiffré 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 de clé sont laissés de côté car ces valeurs sont stockées avec les enregistrements de la base de données du client, et il n'y a pas besoin de cet enregistrement pour satisfaire une demande sur la base d'un TGT.

Recherche de tickets révoqués

Chaque fois qu'une demande est faite au TGS, le ou les tickets présentés sont confrontés à une liste rouge de tickets annulés. Celle liste 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 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 l'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 se 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.

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. Se 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 par le domaine du principal client, sera ajouté au champ de transit du ticket à produire. Ceci est

accomplis en lisant le champ de transit d'après de 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 pourrais 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) est décris ci-dessous.

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

Un nom de domaine se terminant par un "." est interprété comme étant ajouté au domaine précédent. Par exemple, ou 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 un "/" 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 un "/" doit être autonome, il devrait alors être précédé d'un espace (" "). Par exemple, on pour 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 :

"/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 traversées. ",EDU, /COM," signifie que tous les domaines depuis le domaine du client jusqu'à EDU (dans une hiérarchie de type 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.

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, s'il a été spécifié dans la demande, ou la clé de session provenant du TGT, plutôt que la clé secrète du client. Le nom du serveur retourné dans la réponse est le vrai nom du principal du service.

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 une somme de contrôle à l'épreuve des collisions des données utilisateur et de quelques informations de contrôle. La somme de contrôle est assortie 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.

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 assortie mandatée pour être mise en œuvre ainsi que le système de chiffrement 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 mise 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 missage **KDB_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 2 mécanismes. Ce choix devrait se fonder sur les besoins du protocole d'application.

Les numéros de séquence sont utiles lorsque tous les messages envoyés sont 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.

Réception du message KRB_SAFE

Lorsqu'une application reçoit un message **KRB_SAFE**, elle le vérifie comme suit. Si une erreur survient, un code d'erreur est retourné.

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 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. D'adresse de l'expéditeur doit être incluse dans les informations de contrôle ; le receveur vérifie que le rapport du système d'exploitation de l'adresse de l'expéditeur correspond à l'adresse de l'expéditeur 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 expéditeurs peuvent utiliser le type d'adresse directionnel pour l'adresse d'expéditeur 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 **timestamp** et **usec** et/ou de numéro de séquence sont vérifiés. Si timestamp et usec sont attendus et absents, ou s'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 timestamp soient strictement ordonnées ; il est seulement exigé qu'ils soient dans la plage permise. 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 timestamp 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 te de contrôle, et si elle ne correspond pas à la somme d contrôle reçue, une erreur **KRB_AP_ERR_MODIFIED** est générée.

Si toutes les vérification réussissent, l'application est sûr 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 checksum qui a à la fois la sécurité adéquate et des clés compatibles avec la clé de session ou de sous-session. Les chechsums non keyed ou qui ne sont pas à l'épreuve des collisions ne conviennent pas pour cette utilisation.

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.

Génération du message KRB_PRIV

Lorsqu'une application souhaite envoyer un message **KRB_PRIV**, elle collecte ses données et les informations de contrôle appropriées 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 timestamp soit un numéro de séquence (ou les 2). 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.

Réception du message KRB_PRIV

Lorsqu'une application reçoit un messagse **KRB_PRIV**, elle le vérifie comme suit. Si une erreur survient, un code d'erreur est rapporté.

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'éxpéditeur doit être incluse dans les informations de contrôle ; le receveur vérifie que le rapport du système d'exploitation de l'adresse de l'expéditeur correspond à l'adresse de l'expéditeur dans le message. Si une adresse de receveur est spécifiée ou si le receveur exige une adresse, une des adresse du receveur doit également apparaître comme adresse de receveur dans le message. Lorsqu'une adresse d'expéditeur ou de receveur peut ne par correspondre 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 directionnel à la place de l'adresse réseau réelle.

Après cela intervient la vérification des champs timestamp et usec et/ou de numéro de séquence. Si le timestamp et usec sont attendus et ne sont pas présents, ou s'ils sont présents mais ne sont pas ceux en cours, l'erreur **KRB_AP_ERR_SKEW** est générée. Si les champs de nom du serveurs, ainsi que le nom du client, l'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 timestamp, 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.

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.

Génération d'un message KRB_CRED

Lorsqu'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 information 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_CRÉD**. L'heure courante et, s'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 l'inclure dans un jeton de contexte initial GSS-API, 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'intéropérabilité avec MIT, la mise en œuvre Microsoft ne chiffrera pas le **KRB_CRED** dans un jeton GSS-API si elle utilise une clé de session DES. MIT Kerberos peut recevoir et décoder des jetons **KRB_CRED** chiffrés ou non dans l'échange GSS-API. La mise en œuvre Heimdal de Kerberos peut aussi accepter des messages **KRB_CRED** chiffrés ou non. Comme le message **KRB_CRED** dans un jeton GSS-API 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.

Réception d'un message KRB_CRED

Lorsqu'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'expéditeur correspond à l'adresse de l'expéditeur dans le message, et qu'une des adresse 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 environnement de traducteur d'adresses réseau (NAT). 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 timestamp et usec (et nonce, s'il est exigé) sont vérifiés ensuite. Si timestamp et usec ne sont pas présents, ou s'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**.

Échanges d'authentification user to user

L'authentification d'utilisateur à utilisateur 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 ex : un serveur Windows) comme utilisateur sur une workstation. Dans un tel cas, le serveur peut avoir accès au TGT obtenu lorsque l'utilisateur se connecte sur la workstation, mais comme le serveur fonctionne comme un utilisateur non privilégié, il peut ne pas avoir accès aux clés du système. Des situations similaires peuvent survenir en faisant fonctionnes des applications paire-à-paire.

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. 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 passe pour le vérificateur.

Le premier message est 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'utilisateur à utilisateur, 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 et spécifie l'option **ENC-TKT-IN-SKEY** dans sa demande.

S'il est validé, le ticket d'application retourné au client va être chiffré en utilisant la cé 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 l'utilisateur à utilisateur, le client doit spécifier le flag **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'application.

Spécifications de chiffrement et de checksum

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 KDC 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 d'identité du service.

L'opération string-to-key fournie par la rfc3961 est utilisée pour produire une clé à long terme pour un principal (généralement pour un utilisateur). La chaîne salt par défaut, si fournis 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 des paramètres opaque de string-to-key par défaut comme définis dans la rfc3961.

Les données chiffrées, et les checksums sont transmis en utilisant les objets de données **EncryptedData**, **EncryptionKey**, et **Checksum**. Les opérations de chiffrement, déchiffrement, et de checksums 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

KeyUsage spécifiées dans la description de chaque objet EncryptedDate 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 utilisations de clé sont des entiers non signés 32-bits, le 0 n'est pas permis. Les valeurs d'utilisation de clé de 512 à 1023 sont réservés pour des utilisation internes à une mise en œuvre de Kerberos. (par exemple, alimenter un générateur de nombre pseudo-aléatoire avec une valeur produite en chiffrant quelque chose avec une clé de session et une valeur d'utilisation de clé non utilisée pour un autre objet.) Les valeurs d'utilisation de clé de 1024 à 2047 sont réservés à l'usage des applications ; les applications devraient utiliser des valeurs paires pour le chiffrement et des valeurs impaires pour les checksums dans cette gamme.

Il peut exister d'autres documents qui définissent les protocoles dans les termes des types de chiffrement ou de checksums de la rfc1510. Ces documents ne connaissent rien sur les utilisations de clé. Afin que ces spécifications continuent d'avoir un sens en attendant leur mise à jour, si aucune valeur d'utilisation de clé n'est spécifié, les utilisation de clé 1024 et 1025 doivent être utilisés pour déduire les clés pour le chiffrement et les checksums, respectivement. (ceci ne s'applique pas aux protocoles qui ont leur propre chiffrement indépendemment 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 checksum Kerberos devraient utiliser leurs propres valeurs d'utilisation de clé.

Sauf indication contraire, aucun chaînage d'état de chiffrement n'est fait d'une opération de chiffrement à un 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 de données). 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 des types de clé de session suffisamment fort au KDC et espérer que tout type retourné par le KDC sera compris et pris en charge par le serveur d'application.

Spécifications de message

Le protocole Kerberos est défini ici en termes de notation ASN.1, qui fournit une syntaxe de spécification à la fois de la disposition abstraite des messages du protocole et de leurs codages. Les mises en œuvre 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 correcte de la mise en œuvre.

Noter qu'en plusieurs endroits, ds changements des types abstraits ont été faits par rapport à la rfc1510. 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 rfc1510 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 méta-langage distinctif (DER, Distinguished Encoding Rules). Ce peu 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 d'utilisation du BIT STRING), les codages résultant de l'utilisation de DER restent identiques entre les types définis dans la rfc1510 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 d'identifiant d'objet est spécifiée par erreur comme ayant la valeur de "5". Dans le

cas de la rfc1964, l'utilisation de la valeur d'OID correcte résulterait en un changement du protocole du câble ; elle reste donc inchangée pour l'instant.

Noter qu'ailleurs dans le 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 souligner dans les identifiants, aussi dans les définitions ASN.1 réelles, les soulignés sont remplacés par le traits d'union (-). De plus, les noms de membre de la structure et les valeur définies en ASN.1 doivent commencer par une lettre minuscule, alors que les noms du types doivent commencer par une majuscule.

Notes de compatibilité 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 pas les variantes standard de l'ASN.1. L'objet de ces notes est plutôt de décrire quelques bizarreries historiques et non la 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.

Règles de codage distinct ASN.1

Les codage des messages du protocole Kerberos doit obéir aux règles de codage en DER 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) les plus générales ; en particulier, cet mises en œuvre envoient des codages de longueur indéfinie. Les mises en œuvre peuvent accepter de tels codages dans l'intérêt de la rétro compatibles, bien qu'on doive prévenir les mises en œuvre que le décodage du BER le plus général est plein de périls.

Champs d'entier facultatifs

Certaines mises en œuvre de distinguent par 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 micro-secondes, 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.

Types 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 OPTIONAL, mais devraient les accepter comme équivalents à un type OPTIONAL 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.

Tag Numbers 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 messages que les parties comprennent ; par exemple, au moyen d'un flag 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 le tag incorrecte est envoyé sur TCP. Les KDC ne devraient pas répondre aux messages reçus avec un tag inconnus sur UDP afin d'éviter les attaques DOS. 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.

Tag Number supérieurs à 30

Une mise en œuvre 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 sont codés en utilisant plus d'un octet. Les révisions futures du présent protocole pourraient utiliser les numéros supérieurs à 30, et les mises en œuvre devraient être préparées à retourner une erreur, le cas échéant, lorsqu'elles ne reconnaissent pas l'étiquette.

Types Kerberos de base

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

KerberosString

La spécification d'origine du protocole Kerberos dans la rfc1510 utilise GeneralString à de nombreux endroits pour les données de chaîne human-readable. Les mise 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 pour commuter les ensembles de caractères, et l'ensemble de caractères par défaut qui est conçus désigné sous le nom de G0 est la version de référence internationale (IRV) - US ASCII de ISO-646/ECMA-6.

La nomes ISO-2022/ECMA-35 définit 4 éléments de code d'ensemble de caractèrse (G0 à G3) et 2 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.

En pratique, de nombreuses mises en œuvre traitent GeneralStrings comme chaînes 8 bits par défaut quel que soit le jeu de caractères, sans considération de l'utilisation correcte des séquences d'échappement. Le jeu de caractères par défaut est souvent déterminé par la localisation du systèmes d'exploitation de l'utilisateur habituel. Au moins une mise en œuvre majeure place des caractères Unicode codés en UTF-8 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 client, 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 implémentations 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 implémentations 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.

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 se doivent pas contenir de caractère avec le code 0 (US-ASCII NULL). 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 "/", dans le style des noms X.500. Un PrincipalName est une séquences typée de composants comportant les sous champs suivants :

name-type Ce champ spécifie le type de nom qui suit. Ce champ devrait être traité comme conseil.

name-string Ce champ code une séquence de composants qui forment un nom, chaque composant étant codé comme un KerberosString. Pris ensemble, un PrincipalName et un domaine forment un identifiant de principal.

KerberosTime

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

Les horodatages utilisés dans Kerberos sont codés comme GeneralizedTime. Une valeur KerberosTime ne doit pas inclure de portions fractionnée de 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 aprés 21h le 6 novembre 1985 est 19851106210627Z.

Contraintes sur les types entiers

Certains membres de type entier devraient être contraints à des valeur représentables en 32-bits, pour la compatibilité avec des limites raisonnables d'implémentation.

```
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 rfc1510, le codage DER ne devrait pas être altéré. Les anciennes implémentations étaient normalement limitées à des valeurs d'entier 32-bits, et les numéros alloués devraient entrer dans l'espace des valeurs entières représentables en 32-bits afin de promouvoir l'intéropérabilité. Plusieurs champs d'entiers dans les messages sont contraints à des valeurs fixes.

pvno comme TGT-VNO ou AUTHENTICATOR-VNO, ce champ récurant vaut toujours 5.

msg-type Ce champ est habituellement identique au tagNumber d'application du type de message qui le contient

HostAddress et HostAddresses

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

Note :HostAddress est toujours utilisé comme champ OPTIONAL et ne devrait pas être vide. Le codage d'adresse d'hôte comporte 2 champs :

addr-type Ce champ spécifie le type d'adresse suivant.

address Code une seule adresse du type addr-type.

AuthorizationData

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 Contient des données d'authorisation à interpréter conformément à la valeur du chanp ad-type correspondant

ad-type Spécifie le format du sous-champ ad-data.

Chaque séquence de type et données est perçue comme un élément d'autorisation. Les éléments peuvent être spécifiques à l'application ; cependant, il y a un jeu commun d'éléments récursifs qui devraient être compris par toutes les implémentations. Ces éléments contiennent d'autres éléments embarqués, et l'interprétation de l'élément encapsulé détermine lequel des éléments embarqués doit être interprétés, et ceux qui peuvent être ignorés.

Ces éléments de données d'autorisation communs sont définis récursivement, signifiant que **ad-data** pour ces types va lui-même contenir une donnée de séquence d'autorisation. dont l'interprétation est affectée par l'élément encapsulant. En fonction de la signification de l'élément encapsulant, les éléments encapsulés peuvent être ignorés, pourraient être interprétés produit directement par le KDC, ou pourraient être stockés dans une partie plaintext séparé du ticket. Les types d'éléments encapsulant sont spécifiés comme partie de la spécification Kerberos parce que comportement fondé sur ces valeurs devrait être compris te toutes les implémentations, alors que d'autre éléments n'ont besoin d'être compris que par les applications qu'ils affectent.

Les éléments de données d'autorisation sont considérés comme critiques s'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 un **AP-REQ** ou dans un ticket contenu dans un **AP-REP**, alors, sauf s'il est encapsulé dans un élément de données d'autorisation commun 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**.

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
```

IF-RELEVANT

AD-IF-RELEVANT : := AuthorizationData

Les éléments 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 implémentations qui peuvent avoir des extensions locales pour l'autorisation.

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

KDCIssued

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

ad-checksum ckecksum cryptographique calculés sur le codage DER de AuthorizationData dans le champ "elements", keyed avec la clé de session. Son checksumtype est le type de checksum obligatoire pour le type de chiffrement de la clé de sessio, et sa valeur d'utilisation de clé est 19.

i-realm, i-sname Le nom du principal qui émet, s'il est différent de celui du KDC. Ce champ devrait être utilisé lorsque le KDC peut vérifier l'authenticité des éléments signés par le principal qui émet, et permet au 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 un checksum 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 par copiés directement sur un ticket dérivé. S'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'application, et KDC qui n'implémentent pas cet élément.

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 foins le nombre d'éléments encapsulés spécifié dans condition-count est satisfait. Donc, cet élément peut être utilisé 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)

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)

PA-DATA

Historiquement, les PA-DATA étaient connues comme des données de pré-authentification, ce qui signifie qu'elles étaient utilisée 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 protocoles 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 utilisée pour une interprétation répertoriée du type d'élément.

padata-value Contient habituellement le codage 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 avec Kerberos.

PA-TGS-REQ

Dans le cas de demandes de tickets supplémentaire **KRB_TGS_REQ**, padata-value contient une **AP-REQ** coddée. 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.

Pré-authentification à timestamp chiffré

Il y a des types de pré-authentification qui peuvent être utilisés pour pré-authentifier un client au moyen d'un timestamp 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'utilisation de clé de 1.

PA-PW-SALT

padata-value pour ce type de pré-authentification contient de salt pour le string-to-key à utiliser par le client pour obtenir la clé pour déchiffrer 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 à la localisation.

Dans l'exemple trivial, une chaîne de salt de longueur 0 est très courant pour les domaines qui ont converti leur bases de données de principal depuis une version 4. Un KDC ne devrait par envoyer **PA-PW-SALT** lorsqu'il produit un message **KRB-ERROR** qui demande une pré-authentification supplémentaire. Note d'implémentation : Certaines implémentations 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 **KDC-ERROR** qui demande une pré-authentification supplémentaire. Un KDC ne doit pas envoyer

PA-PW-SALT lorsque le **AS-REQ** du client comporte au moins un etype "newer".

PA-ETYPE-INFO

Le type de pré-authentification **ETYPE-INFO** est envoyé par le KDC dans un **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 timestamp chiffré pour les besoins de l'envoi d'une valeur **PA-ENC-TIMESTAMP** de pré-authentification. Il peut aussi être envoyé dans un **AS-REP** pour fournir des informations au client sur le salt de clé à utiliser pour le string-to-key que le client doit utiliser pour obtenir la clé de déchiffrement de la partie chiffrée du **AS-REP**.

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

Le salt, comme celui de **PA-PW-SALT**, est aussi complètement non spécifié par rapport au jeu de caractères et est probablement spécifique à la localisation. Si **ETYPE-INFO** est envoyé dans un **AS-REP**, il doit être exactement un **ETYPE-INFO-ENTRY**, et son etype doit correspondre à celui du enc-part dans **AS-REP**. Un KDC ne doit pas envoyer **PA-ETYPE-INFO** lorsque le **AS-REQ** du client comporte un moins un etype "newer".

PA-ETYPE-INFO2

Le type de pré-authentification **ETYPE-INFO2** est envoyé par le KDC dans un **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 timestamp chiffré pour les besoins de l'envoi d'une valeur **PA-ENC-TIMESTAMP** de pré-authentification. Il peut aussi être envoyé dans un **AS-REP** pour fournir des informations au client sur le salt de clé à utiliser pour le string-to-key que le client doit utiliser pour obtenir la clé de déchiffrement de la partie chiffrée du **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 de salt est un KerberosString, mais les installations existantes peuvent avoir des caractères spécifiques de la localisation mémorisés dans le chaînes de salt, et les développeurs peuvent choisir de les traiter. L'interprétation de s2kparams est spécifiée dans la description du cryptosystem associé au etype. Chaque cryptosystem 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 un **AS-REP**, il doit être exactement un **ETYPE-INFO2-ENTRY**, et son etype doit correspondre à celui du enc-part dans le **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**. Un KDC ne doit pas envoyer de **ETYPE-INFO** ou **PW-SALT** lorsque **AS-REQ** comporte au moins un etype "newer".

KerberosFlags

```
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 rfc1510 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 RFC1510.

Les chaînes binaires ASN.1 ont plusieurs utilisations. L'utilisation la plus simple d'une chaîne 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 8 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 implémentations de langage C qui choisissent naturellement de mémoriser les vecteur booléens comme des entiers de 32bits.

Par exemple, si la notation de KDCOptions devait inclure la notation "NamedBit", comme dans la rfc1510, et si la valeur KDCOptions à coder avait seulement le bit "transmissible" (bit numéro 1) mis, le codage DER doit n'inclure que 2 bits : le premier est à 0 et est réservé, et le bit de valeur 1 pour transmissible.

La plupart des implémentations de Kerberos envoient inconditionnellement 32 bits sur le réseau lors du codage de chaînes binaires utilisée comme vecteurs booléens. Ce comportement viole la syntaxe ASN.1 utilisée pour les valeurs de flag dans la rfc1510, 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 relèguant en commentaires. La contrainte de taille sur le type KerberosFlags exige qu'au moins 32 bits soient codés à tout moment, bien qu'une implémentation laxiste puisse choisir d'accepter moins de 32bits et de traiter les bits manquants comme mis à 0.

Actuellement, aucune utilisation de KerberosFlags ne spécifie plus de 32bits de flags, 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 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".

Type liés au cryptosystem

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 clé pour le déchiffrer

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

etype Ce champ spécifie l'algorithme de chiffrement utilisé pour chiffrer le texte

kvno Ce champ contient le numéro de version de clé utilisée pour chiffrer les données. n'est présent que dans les messages chiffrés avec des clés à longue durée.

cipher Contient le texte chiffré.

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

```
EncryptionKey ::= SEQUENCE {
 keytype [0] Int32 - actually encryption type -,
```

```
 keyvalue [1] OCTET STRING
}
```

keytype Spécifie le type de chiffrement de la clé de chiffrement qui suit dans le champ keyvalue.

keyvalue Contient la clé elle-même, codée comme une chaîne d'octet.

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 Checksums utilisent un hachage chiffré, bien que des exceptions existent.

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

cksumtype Indique l'algorithme utilisé pour générer la somme de contrôle qui l'accompagne

checksum Contient la somme de contrôle elle-même.

Tickets

Ce paragraphe décrit les paramètres de format et de chiffrement pour les tickets et les authentifiants. Lorsqu'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 flags suivants sont nouveaux depuis la RFC 1510
 - transited-policy-checked(12),
 - ok-as-delegate(13)
```

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

realm Spécifie le domaine qui a produit un ticket. Sert 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 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 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'utilisation de 2.

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

0 (Réserved)

1 (forwardable) Le flag FORWARDABLE n'est normalement interprété que par le TGS, et peut être ignoré par les serveurs d'extrémité. Lorsqu'il est mis, ce flag dit au TGS qu'il est ok pour produire un nouveau TGT avec une adresse réseau différente fondée sur le ticket présenté.

2 (forwarded) Lorsque mis, ce flag indique que le ticket a été transmis ou a été produit sur la base d'une authentification impliquant un TGT transmis.

3 (proxiable) Normalement interprété que par le TGS et est identique que FORWARDABLE, excepté qu'il dit au TGS que seuls des non-TGT peuvent être produits avec des adresses réseau différentes.

4 (proxy) Indique que ce ticket a été mandaté

5 (may-postdate) Normalement interprété que par le TGS et indique qu'un ticket post-daté peut être produit sur la base de ce TGT.

6 (postdated) Indique que ce ticket est post-daté

7 (invalid) Indique que ce ticket est invalide, et qu'il doit être validé par le KDC avant utilisation.

8 (renewable) Normalement interprété que par le TGS et peut être utilisé pour obtenir un ticket de remplacement qui expire à une date ultérieur.

9 (initial) Indique que ce ticket a été produit en utilisant le protocole d'AS, et non pas produit sur la base d'un TGT.

10 (pre-authent) Indique que durant l'authentification initiale, le client a été authentifié par le KDC avant la production d'un ticket.

11 (hw-authent) Indique que le protocole utilisé pour l'authentification initiale exige l'utilisation d'un matériel.

12 (transited-policy-checked) 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 flag est à 0, le serveur d'application doit alors vérifier le champ de transit lui-même, et s'il est incapable de le faire, il doit rejeter l'authentification. Si le flag est à 1, le serveur d'application peut alors s'affranchir de la vérification.

13 (ok-as-delegate) 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 flag pour l'aider à décider de déléguer des accréditifs à ce serveur.

14-31 (reserved) Réservé

key Ce champ existe dans le ticket et la réponse du KDC et il est utilisé pour passer la clé de session 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'utilisateur à qui ce ticket a été produit. Il ne spécifie par l'ordre dans lequel les domaines ont été 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 transposé en éléments dans ce champ en utilisant la transposition définis dans la rfc2253.

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.

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

endtime Ce champ contient l'heure après laquelle le ticket ne sera plus honoré.

renew-till Ce champ n'est présent que dans les ticket qui on le flag RENEWABLE mis. 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 pour les renouvellements.

caddr Ce champ dans un ticket contient zéro (s'il est omis) ou plusieurs adresses d'hôte. Ce sont les adresses à partir desquelles le ticket peut être utilisé. S'il n'y a pas d'adresse, 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 adresse 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 pouvant l'être, un agresseur qui a compromis la workstation 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'un localisation sûre.

authorization-data Ce champ est utilisé pour passer les données d'autorisation du principal au nom duquel un ticket a été produit au service d'application. Si aucune données 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".

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 lorsqu'un nouveau ticket est obtenu durant l'échange TGS, ou elle peuvent être ajoutées durant une délégation chaînée utilisant le champ de données d'autorisation de l'authentifiant.

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. Bien que Kerberos ne soit pas concerné par le format du contenu des sous-champs, il en porte les information 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 lorsqu.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, de 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.

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

Spécification pour les échanges AS et TGS

Ce paragraphe spécifie le format des messages utilisés dans l'échange entre le client et le serveur Kerberos.

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 2 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)
```

pvno Inclus dans chaque message de protocole, et spécifie le numéro du protocole. Doit être à 5

msg-type Indique le type d'un message. 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**. Ce champ contient une séquence d'informations d'authentification qui peuvent être nécessaire avant que les accréditifs puissent être produits ou déchiffrés.

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

kdc-options Ce champ apparaît dans les demandes **KRB_AS_REQ** et **KRB_TGS_REQ** et indique les flags 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 flag 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 cahmp kdc_options est un champ binaire, où les options choisies sont indiquées par le bit mis, et les option non choisies et les champs réservés ne sont pas mis. La signification des options est la suivante :

0 (Réserved)

1 (FORWARDABLE) Indique que le tickest est fournis avec le flag forwardable mis. Ne peut seulement être mis dans la demande initiale, ou dans une sous-demande si le TGT sur lequel il est basé est également forwardable.

2 (FORWARDED) Seulement spécifié dans une demande au TGS et sera honoré si le TGT dans la demande a le bit FORWARDABLE mis.

3 (PROXIABLE) Indique que le tickest est fournis avec le flag proxiable mis. Ne peut seulement être mis dans la demande initiale, ou dans une sous-demande si le TGT sur lequel il est basé est également proxiable.

4 (PROXY) Indique que c'est une demande pour un proxy et sera honoré si le TGT dans la demande a le bit PROXIABLE mis.

5 (ALLOW-POSTDATE) Indique que le ticket à fournir doit avoir MAY-POSTDATE mis. Ne peut seulement être mis dans la demande initiale, ou dans une sous-demande si le TGT sur lequel il est basé est également MAY-POSTDATE.

6 (POSTDATED) Indique que c'est une demande pour un ticket post-daté. Sera honoré si le TGT sur lequel il est basé a son MAY-POSTDATE mis. Le ticket résultant aura également le flag INVALID mis.

7 (RESERVED)

8 (RENEWABLE) Indique que le ticket à fournir doit avoir son flag RENEWABLE mis. Peut seulement être mis dans la demande initiale, ou dans une sous-demande si le TGT sur lequel il est basé est également renewable

9 (RESERVED)

10 (RESERVED)

11 (RESERVED)

12-25 (RESERVED)

26 (DISABLE-TRANSITED-CHECK) Indique au KDC de ne pas vérifier le champ transited

27 (RENEWABLE-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 renew-till égal à l'heure de fin demandée.

28 (ENC-TKT-IN-SKEY) N'est utilisée que par le TGS. indique que le ticket pour le serveur d'extrémité est à chiffrer avec la clé de session provenant du TGT supplémentaire fourni.

29 (RESERVED)

30 (RENEW) N'est utilisé que par le TGS. Indique que la demande est pour un renouvellement. Le ticket fournis est chiffré dans la clé secrète pour le serveur sur lequel il est valide. Ne sera honoré que si le ticket à renouveler a son flag RENEWABLE mis et si renew-till n'est pas passé.

31 (VALIDATE) N'est utilisé que par le TGS. Indique que la demande est pour un ticket post-daté à valider.

cname et sname Ces champs sont les mêmes que ceux décrits pour le ticket plus haut. Le sname ne peut être absent que lorsque l'option **ENC-TGT-IN-SKEY** est spécifié. 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 Si présent (et ne peut l'être que sous la forme de TGS_REQ), est un codage des authorizations désirées chiffré avec la sous-clé de session si elle est présent dans l'authentifiant, ou avec la clé de session dans le TGT (l'authentifiant et le TGT viennent tous deux du champ padata dans le KRB_TGS_REQ). La valeur d'utilisation 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 Spécifie la partie domaine de l'identifiant de principal du serveur. Dans l'échange 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 post-dater. Il spécifie l'heure de début de validité pour le ticket demandé. Si omis, le KDC devrait uiliser l'heure en cours.

till 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 aura l'heure maximum permise par le KDC.

rtime Ce champ est l'heure de renew-till envoyée au KDC dans une demande de ticket. facultatif.

nonce 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 attaquant. Les noms occasionnels ne doivent jamais être réutilisés.

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

addresses Ce champ est inclus dans la demande de ticket, et il est facultativement inclus dans les demandes de tickets supplémentaire provenant du serveur d'allocation de tickets. Il spécifie les adresse à partir desquelles le ticket demandé doit être valide. Normalement, il inclut les adresse de l'hôte du client. Si un proxy est demandé, ce champ contiendra d'autres adresse. Ce champ est généralement copié par le KDC dans le champ cappr du ticket résultant.

additional-tickets Des tickets supplémentaire peuvent être facultativement inclus dans une demande au TGS. Si l'option ENC-TGT-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'utilisateur à utilisateur, 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 supplémentaire a été spécifiée, les tickets supplémentaire sont alors utilisés dans l'ordre spécifié par l'ordre des bits des options bits.

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 chifrée de la réponse dépend du type de message. Pour **KRB_AS_REP**, le texte 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**, et texte est chiffré avec la clé de session provenant du TGT utilisé dans la demande. Dans ce cas, aucun numéro de version ne sera présent dans la séquence EncryptedData.

Les champs de message sont les 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 Décrits dans **KRB_KDC_REQ**. msg-type est KRB_AS_REP ou KRB_TGS_REP

padata Décrits dans **KRB_KDC_REQ**. Une utilisation possible est de coder une chaîne salt à utiliser avec un algorithme de string-to-key. C'est utile pour faciliter les transitions si un nom de domaine doit changer ; dans un tel cas, toutes les entrées déduites de mot de passe existantes dans la base de données Kerberos auraient un flag marquant qu'elles ont besoin d'une chaîne salt spécial jusqu'au prochain changement de mot de passe.

crealm, cname, srealm, et sname Décrits dans **KRB_KDC_REQ**.

ticket C'est le ticket nouvellement produit

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'utilisation 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'utilisation de clé est 8 si la clé de session TGS est utilisée, ou 9 si une sous-clé d'authentifiant TGS est utilisée.

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

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. I peut aussi couvrir tous les serveurs d'un domaine, ou juste un serveur particulier. Certaines implémentations peuvent afficher ces informations à l'utilisateur 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 tamps partagé.

lr-typeCe champ indique comment interpréter le champ lr-value suivant. Les valeurs négatives indiquent que ces informations n'app

0 Aucune information n'est portée par le sous-champ lr-value.

1 le sous-champ lr-value est l'heure de la dernière demande initiale pour un TGT.

2 Le sous-champ lr-value est l'heure de la dernière demande initiale.

3 Le sous-champ lr-value est l'heure de la production du plus récent TGT utilisé

4 Le sous-champ lr-value est l'heure de la demande du dernier renouvellement

5 Le sous champ lr-value est l'heure de la dernière demande (de tout type).

6 Le sous-champ lr-value est l'heure de l'arrivée à expiration du mot de passe.

7 Le sous-champ lr-value est l'heure de l'arrivée à expiration du compte.

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

nonce Décrits dans **KRB_KDC_REQ**.

key-expiration Ce champ 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. S'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 déconseillé, 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 (généralement le programme de connexion) de prendre les mesures appropriées 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, 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 proxy ou un ticket transmis, ou si l'utilisateur 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.

Spécifications de message client-serveur

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

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 utilisée, et le ticket et l'authentifiant eux-mêmes. Le message **KRB_AP_REQ** est souvent désigné comme un 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 Décrits dans **KRB_KDC_REQ**. 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 bis mis, et les options non choisies à 0. La signification des options est la suivante :

(reserved) 1

(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 serve
2

(mutual-required) Dit au serveur que le serveur exige l'authentification mutuele, et qu'il doit répondre par un message KRB_AP_
3-31

(reserved)

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

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

L'authentifiant chiffré est inclus dans le 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'utilisation 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.

```
- 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 Spécifie le numéro de version du format de l'authentifiant. Vaut 5

crealm et cname Décrit dans le paragraphe ticket

cksum Contient un checksum des données d'application qui accompagnent le **KRB_AP_REQ**. calculée en utilisant une valeur d'utilisation de clé de 10 dans les échanges d'application normaux, ou 6 lorsqu'utilisé dans le champ **TGS-REQ PA-TGS-REQ AP-DATA**

cusec Ce champ contient la partie microsecondes du timestamp du client. sa valeur va de 0 à 999999.

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 optionnel comporte le numéro de séquence initial à utiliser par **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 **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 puis reviennent à 0.

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 message **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.

authorization-data Ce champ est décrit au paragraphe ticket. il est optionnel 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.

Définition de KRB_AP_REP

Le message **KRB_AP_REP** contient le numéro de version du protocole Kerberos, le type de message, en un timestamp 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
}
```

nc-part La partie codée **EncAPRepPart** est chiffrée avec la clé de session partagée du ticket. Le champ de 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 Décrits dans **KRB_KDC_REQ**. msg-type est**KRB_AP_REP**

enc-part Décrits dans **KRB_KDC_REQ**. Est calculé avec une valeur d'utilisation de clé de 12

ctime Heure courant sur l'hôte du client

cusec partie micro-seconde du timestamp

subkey Contient une clé de chiffrement à utiliser pour protéger cette session d'application spécifique.

seq-number Décrit au paragraphe ticket

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. Le message **KRB_SAFE** contient des données d'utilisateur 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,
 timestamp [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 Décrits dans **KRB_KDC_REQ**. msg-type est **KRB_SAFE**

safe-body Ce champ est un fourre tout pour le corps du message **KRB_SAFE**

cksum Ce champ contient le checksum des données d'application, calculée avec une valeur d'utilisation 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.

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.

timestamp Ce champ fait partie des messages KRB_SAFE et KRB_PRIV. Heure courante le l'émetteur.

usec partie micro-seconde du timestamp

seq-number Ce champ est décrit au paragraphe ticket

s-address Spécifie l'adresse utilisée par l'eméteur du message

r-address 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 on été incorrectement ou malicieusement délivrés à un mauvais receveur.

Spécification du message KRB_PRIV

Ce paragraphe spécifie le format d'un message qui peut être utilisé par un client et un serveur d'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 message KRB_AP_REQ/KRB_AP_REP. Le message KRB_PRIV contient des données d'utilisateur chiffrées avec la clé de session.

Les champs de messages sont les suivant :

```
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 Décrits dans **KRB_KDC_REQ**. msg-type est **KRB_PRIV**

enc-part Ce champ détient un codage de la séquence EncKrbPrivPart chiffrée avec le clé de session, avec une valeur d'utilisation 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 Décrits dans **KRB_SAFE**

seq-number Décrits dans la section ticket

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 subordonnées. Il suppose qu'une clé de session a déjà été échangée, peut-être en utilisant les message KRB_AP_REQ/KRB_AP_REP. 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,
 timestamp [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 Décrits dans **KRB_KDC_REQ**. msg-type est **KRB_PRIV**

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

enc-part Ce champ étient un codage de la séquence EncKrbCredPart chiffrée avec la clé de session partagée par l'envoyeur et le receveur prévu, avec une valeur d'utilisation de 14. Ce codage chiffré est utilisé pour le champ enc-part du message KRB-CRED.

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

timestamp 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 Décrit dans **KRB_SAFE**

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.

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

lags, authtime, starttime, endtime, renew-till, srealm, sname, et caddr facultatifs. 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 messages KDC-REP.

Spécification du message KRB_ERROR

Ce paragraphes spécifie 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 attaquant de le synthétiser ou le 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 utilisateur des raisons d'un échec.

Les champs de message sont les 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 Décrits dans **KRB_KDC_REQ**. msg-type est **KRB_ERROR**

ctime et cusec Ces champs sont décris dans **KRB_AP REP**.Si les valeurs de ces champs sont connues de l'entité qui génère l'erreur, ils devraient être remplis dans la KRB-ERROR. Se 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 du timestamp.

error-code Ce champ contient le code d'erreur retourné par Kerberos ou le serveur.

crealm et cname Ces champs sont décrit au paragraphe ticket. Lorsque l'entité qui génère l'erreur connaît ces valeur, elle 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 dans le paragraphe ticket

e-text Ce champ contient du texte supplémentaire pour aider à expliquer le code d'erreur associé à l'échec de demande.

e-data Ce champ contient des données supplémentaires sur l'erreur à utiliser par l'application. Si le codes d'erreur est KDC_ERR_PREAUTH_REQUIRED, le champ e-data contiendra alors un odage 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 l'implémentation. De même, pour les codes d'erreur futurs, le format et le contenu du champ e-data sera défini par l'implémentation sauf spécification contraire. Qu'ils soient définis par l'implémentation 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
}
```

Tag Number d'application

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

tag Number - (type name) - 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 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 protocoles qui utilise Kerberos.

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'intéropérabilité 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-domaine 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 3 styles de noms de domaine : domaine, X500, et autre :

domain : ATHENA.MIT.EDU
X500 : C=US/O=OSF
autre : NAMETYPE :rest/of.name=without-restrictions

Les noms de style domaine doivent ressembler à des noms de domaine : Ils comportent des composants séparés par des points et ils ne contiennent ni " :" ni "/". 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 " :" 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 implémentations de Kerberos fondées sur la rfc1510, mais elle est différente du séparateur recommandé dans la rfc2253.

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 " :" 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 3 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 d'un nom de domaine Internet ou d'un nom X.500, ou, dans le cas où un tel nom n'est pas enregistré, l'autorité pour utiliser un nom de domaine peut être dérivé de l'autorité du domaine parent. Par exemple, s'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'existe pas un nom identique au nom de domaine de l'enfant sauf s'il est alloué à la même entité comme nom de domaine.

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 information 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 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, 2 noms ne peuvent être les mêmes. Les types de nom suivants sont définis :

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

Lorsqu'un nom n'implique pas d'informations autres que son unicité à un moment particulier, 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 n'être jamais ré-alloué, le type de nom d'UID devrait alors être utilisé.

Si le premier composant d'un nom identifie un service et si des composants restants identifient une instance du service d'un 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 TGS 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 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 être utilisé. Ce type peut être utilisé pour identifier les serveurs sur des hôtes qui ont des noms X.500, et où "/" pourrait être ambiguë.

Un type de nom NT-X500-PRINCIPAL devrait être utilisé lorsqu'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 rfc2253.

Un type de nom de SMTP permet à un nom d'être d'un 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 nom, 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 TGS.

Nom des principaux de serveur

L'identifiant de principal pour un serveur sur un hôte sera généralement composé de 2 parties : Le domaine du KDC avec lequel le serveur est enregistré, et un nom à 2 composants 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 qui permet "/" 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, le nom de l'hôte doit être en minuscule. 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 peut être la chaîne host au lieu d'un identifiant spécifique du service.

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 sont des quantités de 32bits codées dans l'ordre MSB. Une adresse IPv4 de bouclage ne devrait pas apparaître dans un PDU Kerberos. Le type des adresses IPv4 est (2)

Adresses Internet (IPv6) Les adresses IPv6 sont des guantités de 128bits codées danfs l'ordre MSB. Le type des adresses IPv6 est (24). Les adresses suivantes ne doivent pas apparaître dans un PDU Kerberos : l'adresse non spécifiée, la loopback, les adresses de lien locaux.

Ces restrictions s'appliquent à l'inclusion dans les champs d'adresse des PDU, mais pas aux champs d'adresse des paquets qui pourraient porter ces PDU. La restriction est nécessaire parce que l'utilisation d'une adresse d'une portée non mondial pourrait permettre l'acceptation d'un message envoyé à partir d'un nœud qui pourrait avoir la même adresse, mais qui ne serait par l'hôte prévu. Si le type d'adresse local doit être utilisé pour une communication, la restriction d'adresse dans les tickets ne doit pas être utilisée. 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 (12).

Adresses Netbios Les adresses Netbios sont des adresses de 16 octets normalement composées de caractères alphanumériques de 1 à 15 et complétées 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 (20)

Adresses directionnelles Inclure l'adresse de l'émetteur 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 4 octets dans l'ordre des octets du réseau. Si le message est généré par la pratie qui envoie le message KRB_AP_REQ original, une adresse de 0 devrait être utilisé. 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 no 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 utilisée que quand le protocoles d'application exige leur prise en charge. Les adresses directionnelles dont du type (3)

Échange de messages KDC : transports IP

Kerberos définis 2 mécanismes de transport IP pour communiquer entre clients et serveurs : UDP/IP et TCP/IP.

UDP/IP

Les KDC qui prennent en charge les transport IP doivent accepter les demandes UDP et devraient les écouter sur le port 88. 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 client devraient utiliser la découverte de KDC pour identifier l'adresse et le port IP auquel ils veulent envoyer leur demande.

Lorsqu'il contacte un KDC pour un KRB_KDC_REQ en utilisant UDP, le client doit envoyer un UDP ne contenant qu'un codage de la demande au KDC. Le KDC répondra avec un datagramme contenant seulement un encodage de la réponse (soi un KRB_ERROR, soit un KRB_KDC_REP) 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. 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.

TCP/IP

Les KDC qui prennent en charge les transports IP doivent accepter les demandes TCP et devraient les écouter sur le port 88. 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 pour identifier l'adresse et le port IP auquel ils enverront leur demande.

Lorsque le message KRB_KDC_REQ est envoyé au KDC sur un flux TCP, la réponse doit être retournées au client sur le même flux TCP qui était établis pour le demandes. Le KDC peut fermer le flux TCP après avoir envoyé une réponse, mais peut laisser le flux ouvert pendant une durées raisonnable s'il attend une suite. Il faut veiller, dans la gestion des connexions TCP avec le KDC, à empêcher les attaques DOS fondées sur le nombre de connexions TCP ouvertes.

Le client doit être prêt à voir le flux fermé 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 peut envoyer plusieurs demandes avant de recevoir des réponses, bien qu'il doive être prêt à traiter la fermeture 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 4 octets dans l'ordre des octets du réseau. Le MSB 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 MSB mis dans le codage de longueur, reçoit une demande avec le MSB mis, il doit retourner un message KRB-ERROR avec l'erreur KRB_ERR_FIELD_TOOLONG et doit clore le flux TCP.

Découverte de KDC sur les réseaux IP

Les implémentations de client Kerberos doivent fournir un moyen pour que le client détermine la localisation des KDC. Traditionnellement, les implémentations Kerberos mémorisent de 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.

DNS

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 majuscule, 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.

Spécification de localisation de KDC avec DNS SRV

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

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

Le proto peut être udp ou tcp. Si ces enregistrements de SRV doivent être utilisé, les enregistrements upd et tcp doivent tous 2 être spécifiés pour tous les développements de KDC. Le Realm est de 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 standar définis dans la rfc2782.

Conformément à cette rfc, le numéro de port utilisé pour les enregistrement de SRV "_udp" et "_tcp" devraient être la valeur allouée à "kerberos" par l'IANA : 88, sauf si le KDC est configuré pour écouter sur un autre accès TCP.

Découverte de KDC pour les Realms name stype domaine

Ce sont des enregistrement DNS pour un domaine Kerberos EXAMPLE.COM. Il y a 2 serveurs Kerberos kdc1.example.com et kdc2.example.con. Les requêtes devraient d'abord être dirigées sur kdc1.example.com conformément à la priorité spécifée. Les pondérations ne sont pas utilisées dans cet 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.

Nom du TGS

L'identifiant de principal du TGS doit être composé de 3 parties : le domaine du KDC qui produit les tickets TGS, et un nom en 2 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 utilisé pour 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 utilisé pour obtenir des tickets du domaine MIT.EDU a l'identifiant de principal "ATHENA.MIT.EDU" (domaine), ("krbtgt", "MIT.EDU") (nom).

OID pour Kerberosv5

Cet OID peut être utilisé pour identifier les messages de protocoles Kerberos encapsulés dans d'autres protocoles :

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

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 spécifiés les gammes qui limitent les valeurs des constantes pour lesquelles les valeurs sont définies ici.

Valeurs d'utilisation de clé

La spécification du chiffrement et de checksum exige en entrée un numéro d'utilisation de clé, pour altérer la clé de chiffrement utilisée dans tout message spécifique afin de rendre plus difficiles certains types d'attaques.

1. timestamp padata AS-REQ PA-ENC-TIMESTAMP, chiffré avec la clé de client

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

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

4. Données d'autorisation TGS-REQ KDC-REQ-BODY, chiffrées avec la clé de session TGS

5. Données d'autorisation TGS-REQ KDC-REQ-BODY, chiffrées avec la sous-clé d'authentifiant TGS

6. Somme de contrôle d'authentifiant AP-REQ de padata TGS-REQ PA-TGS-REQ, frappée avec la clé de session TGS

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

8. Partie chiffrée de TGS-REP (inclut la clé de session d'application), chiffrée avec la clé de session TGS

9. Partie chiffrée de TGS-REP (inclut la clé de session d'application), chiffrée avec la sous-clé d'authentifiant de TGS

10. Somme de contrôle d'authentifiant AP-REQ, frappée avec la clé de session d'application

11. Authentifiant AP-REQ (inclut la sous-clé d'authentifiant d'application), chiffré avec la clé de session d'application

12. Partie chiffrée de AP-REP (inclut la sous-clé de session d'application), chiffré avec la clé de session d'application

13. Partie chiffrée de KRB-PRIV, chiffrée avec une clé choisie par l'application

14. Partie chiffrée de KRB-CRED, chiffrée avec une clé choisie par l'application

15. Somme de contrôle KRB-SAFE, frappé avec une clé choisie par l'application

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)

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.

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

Types d'adresse

IPv4 2
Directionnelle 3
ChaosNet 5
XNS 6
ISO 7
DECNET Phase IV 12
AppleTalk DDP 16
NetBios 20
IPv6 24

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)

Types de codages de transit

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

Numéro de version du protocole

pvno (5) Numéro de version en cours du protocole Kerberos

Types de message Kerberos

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

Types de noms

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

Codes d'erreur

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

Exigences d'intéropérabilité

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'utilisateur à utilisateur ; la prise en change de proxy ; le format des noms de domaines ; le traitement des données d'autorisation ; et la transmission, le post-datage, et le renouvellement de tickets.

Afin d'assurer l'intéropérabilité 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 ultérieurs.

Spécification 2

Ce paragraphe définit la seconde spécification des options. Les implémentations qui sont configurées de cette façon peuvent être considérées comme prenant en charge la spécification 2 de Kerberos v5 (5.2).

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 implémentations peuvent aussi accepter d'autres mécanismes, mais les mécanismes supplémentaires ne peuvent être utilisés qu'en communication avec des principaux connus pour les accepter aussi.

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 le domaine demandé.

Codage de champ de transit DOMAIN-X500-COMPRESS 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 salt et de string-to-key. 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 flags 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 augmente 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 implémentations 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 flag 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.

Valeurs de KDC recommandées

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

proxiable, etc. Admis