Gestion du chiffrement et des certificats dans Oracle® Solaris 11.2

Référence: E53957-02
Septembre 2014

ORACLE®

Table des matières

Utilisation de cette documentation

Le manuel *Gestion du chiffrement et des certificats dans Oracle® Solaris 11.2* explique comment administrer et utiliser le chiffrement et comment créer et gérer des certificats de clé privée/publique.

- **Présentation** : décrit les concepts relatifs à la structure cryptographique et à la structure de gestion des clés, ainsi que les tâches permettant d'utiliser ces technologies pour sécuriser des fichiers.
- **Public visé** : administrateurs système qui doivent mettre en oeuvre la sécurité dans l'entreprise.
- **Connaissances requises** : être familiarisé avec la terminologie et les concepts de sécurité.

Bibliothèque de documentation produit

Les informations de dernière minute et les problèmes connus pour ce produit sont inclus dans la bibliothèque de documentation accessible à l'adresse : http://www.oracle.com/pls/topic/lookup?ctx=E56338.

Accès aux services de support Oracle

Les clients Oracle ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info ou le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs si vous êtes malentendant.

Commentaires

Faites part de vos commentaires sur cette documentation à l'adresse : http://www.oracle.com/goto/docfeedback.

♦ ♦ ♦ CHAPITRE 1

Structure cryptographique

Ce chapitre décrit la fonction de structure cryptographique d'Oracle Solaris, et aborde les sujets suivants :

Pour administrer et utiliser la structure cryptographique, reportez-vous au Chapitre 3, Structure cryptographique.

Nouveautés propres à la cryptographie pour Oracle Solaris 11.2

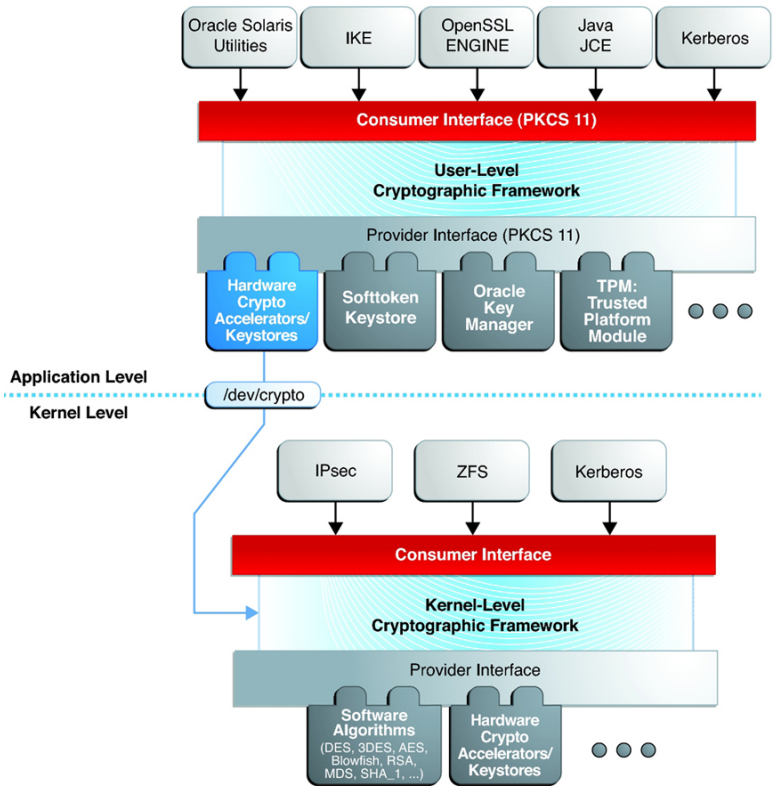
Cette section propose aux clients existants des informations sur les nouvelles fonctionnalités de prise en charge du chiffrement dans cette version.

- Oracle Solaris prend en charge les versions d'OpenSSL compatibles avec le mode FIPS et non compatibles avec le mode FIPS.
- Sur les systèmes SPARC T4 avec optimisations cryptographiques, les instructions cryptographiques sont disponibles directement dans le matériel, ce qui permet d'exécuter plus rapidement les opérations cryptographiques.
- La structure cryptographique prend en charge Camellia, un chiffrement par bloc de 128 bits similaire à AES et principalement utilisé sur le marché japonais.

Introduction à la structure cryptographique

La structure cryptographique fournit un magasin d'algorithmes et de bibliothèques PKCS #11 commun pour traiter les exigences en matière de cryptographie. Les bibliothèques PKCS #11 sont implémentées conformément au standard Cryptoki (Cryptographic Token Interface, interface de jetons cryptographiques) pour la bibliothèque PKCS #11 de RSA Security Inc.

FIGURE 1-1 Niveaux de structure cryptographique



Au niveau du noyau, la structure gère actuellement les exigences en matière de cryptographie pour ZFS, Kerberos et IPsec, ainsi que des matériels. Les consommateurs au niveau utilisateur incluent le moteur OpenSSL, JCE (extensions cryptographiques Java), `libsasl` et le protocole IKE (Internet Key Protocol). Le proxy SSL du noyau (`kssl`) utilise la structure cryptographique. Pour plus d'informations, reportez-vous à la section " Le proxy SSL au niveau du noyau chiffre les communications des serveurs Web " du manuel " Sécurisation du réseau dans Oracle Solaris 11.2 " et à la page de manuel `ksslcfg`(1M).

La loi sur les exportations aux Etats-Unis exige que les interfaces cryptographiques ouvertes soient utilisées sous licence. La structure cryptographique est conforme à la loi en vigueur en exigeant que les fournisseurs cryptographiques du noyau et PKCS 11 s'identifient. Pour plus d'informations, reportez-vous aux informations relatives à la commande `elfsign` dans “Commandes au niveau de l'utilisateur dans la structure cryptographique” à la page 12

La structure permet aux *fournisseurs* de services cryptographiques de voir leurs services utilisés par de nombreux *consommateurs* dans Oracle Solaris. Les fournisseurs sont également appelés des *plug-ins*. La structure prend en charge trois types de plug-ins :

- Plug-ins au niveau de l'utilisateur : objets partagés qui fournissent des services en utilisant les bibliothèques PKCS #11, telles que `/var/user/$USER/pkcs11_softtoken.so.1`.
- Plug-ins au niveau du noyau : modules de noyau qui fournissent l'implémentation d'algorithmes cryptographiques dans les logiciels, tels qu'AES.

  De nombreux algorithmes de la structure sont optimisés pour les architectures x86 avec le jeu d'instructions SSE2 et pour le matériel SPARC. Pour les optimisations de la série T, reportez-vous à “Structure cryptographique et serveurs de série SPARC T” à la page 17.
- Plug-ins matériels : pilotes de périphériques et leurs accélérateurs matériels associés. Les puces Niagara et les pilotes de périphériques Oracle `ncp` et `n2cp` en sont des exemples. Un accélérateur matériel décharge le système d'exploitation de fonctions cryptographiques coûteuses. La carte Sun Crypto Accelerator 6000 en est un exemple.

La structure implémente une interface standard, la bibliothèque PKCS #11, v2.20, révision 3, pour les fournisseurs au niveau de l'utilisateur. La bibliothèque peut être utilisée par des applications tierces pour atteindre les fournisseurs. Des tiers peuvent également ajouter à la structure des bibliothèques signées, des modules d'algorithme de noyau signés et des pilotes de périphériques signés. Ces plug-ins sont ajoutés lorsque Image Packaging System (IPS) installe le logiciel tiers. Pour visualiser un diagramme des principaux composants de la structure, reportez-vous à la Figure 1-1, “Niveaux de structure cryptographique”.

Concepts propres à la structure cryptographique

Notez les descriptions des concepts suivants et les exemples correspondants qui sont utiles dans le cadre de l'utilisation de la structure cryptographique.

- **Algorithmes :** les algorithmes cryptographiques sont des procédures de calcul récursives établies qui chiffrent ou hachent une entrée. Les algorithmes de chiffrement peuvent être symétriques ou asymétriques. Les algorithmes symétriques utilisent la même clé pour le chiffrement et le déchiffrement. Les algorithmes asymétriques, qui sont utilisés dans la cryptographie par clé publique, nécessitent deux clés. Les fonctions de hachage sont également des algorithmes.

  Quelques exemples d'algorithmes :

  - Algorithmes symétriques, comme AES et ECC
  - Algorithmes asymétriques, comme Diffie-Hellman et RSA

- Fonctions de hachage, comme SHA256

- **Consommateurs :** utilisateurs des services cryptographiques provenant de fournisseurs. Les consommateurs peuvent être des applications, des utilisateurs finaux ou des opérations de noyau.

  Quelques exemples de consommateurs :

  - Applications, comme IKE
  - Utilisateurs finaux, comme un utilisateur standard exécutant la commande `encrypt`
  - Opérations de noyau, comme IPsec

- **Keystore :** dans la structure cryptographique, stockage permanent pour les objets-jetons, souvent utilisé de manière interchangeable avec un **jeton**. Pour plus d'informations sur un keystore réservé, reportez-vous à **Metaslot** dans cette liste de définitions.

- **Mécanisme :** application d'un mode d'algorithme pour un objectif particulier.

  Par exemple, un mécanisme DES appliqué à l'authentification, tel que CKM_DES_MAC, est un mécanisme distinct d'un mécanisme DES appliqué au chiffrement, CKM_DES_CBC_PAD.

- **Metaslot :** connecteur individuel réunissant les capacités d'autres connecteurs chargés dans la structure. Le metaslot facilite le travail de gestion de toutes les capacités des fournisseurs disponibles par le biais de la structure. Lorsqu'une application utilisant le metaslot demande une opération, le metaslot détermine quel connecteur réel doit effectuer l'opération. Les capacités du metaslot sont configurables, mais la configuration n'est pas nécessaire. Le metaslot est activé par défaut. Pour plus d'informations, reportez-vous à la page de manuel `cryptoadm`(1M).

  Le metaslot ne dispose pas de son propre keystore. Au lieu de cela, il réserve l'utilisation d'un keystore de l'un des emplacements actuels dans la structure cryptographique. Par défaut, le metaslot réserve le keystore `Sun Crypto Softtoken`. Le keystore utilisé par le metaslot n'est pas indiqué comme l'un des emplacements disponibles.

  Les utilisateurs peuvent spécifier un autre keystore pour metaslot en définissant les variables d'environnement `${METASLOT_OBJECTSTORE_SLOT}` et `${METASLOT_OBJECTSTORE_TOKEN}` ou en exécutant la commande `cryptoadm`. Pour plus d'informations, reportez-vous aux pages de manuel `libpkcs11`(3LIB), `pkcs11_softtoken`(5) et `cryptoadm`(1M).

- **Mode :** version d'un algorithme cryptographique. Par exemple, CBC (Cipher block Chaining, enchaînement des blocs de chiffrement) est un autre mode d'ECB (Electronic Code Book, bloc de contrôle d'événement). L'algorithme AES possède deux modes : CKM_AES_ECB et CKM_AES_CBC.

- **Stratégie :** choix effectué par un administrateur de rendre des mécanismes disponibles pour l'utilisation. Par défaut, tous les fournisseurs et tous les mécanismes sont disponibles pour l'utilisation. L'activation ou la désactivation de tout mécanisme serait une application de la stratégie. Pour consulter des exemples de configuration et d'application de stratégies, reportez-vous à “Administration de la structure cryptographique” à la page 34.

- **Fournisseurs :** services cryptographiques utilisés par les consommateurs. Etant donné que les fournisseurs se connectent à la structure, ils sont également qualifiés de *plug-ins*.

Quelques exemples de fournisseurs :

- Bibliothèques PKCS #11, telles que `/var/user/$USER/pkcs11_softtoken.so`
- Modules d'algorithmes cryptographiques, comme `aes` et `arcfour`
- Pilotes de périphériques et leurs accélérateurs matériels associés, comme le pilote `mca` pour la carte Sun Crypto Accelerator 6000

- **Connecteur :** interface vers un ou plusieurs périphériques cryptographiques. Chaque emplacement, qui correspond à un lecteur physique ou à une autre interface de périphérique, peut contenir un jeton. Un jeton fournit une vue logique d'un périphérique cryptographique dans la structure.
- **Jeton :** dans un connecteur, un jeton fournit une vue logique d'un périphérique cryptographique dans la structure.

Commandes et plug-ins de structure cryptographique

La structure offre des commandes aux administrateurs, utilisateurs et développeurs qui approvisionnent les fournisseurs.

- Commandes d'administration : la commande `cryptoadm` fournit une sous-commande `list` pour répertorier les fournisseurs disponibles et leurs capacités. Les utilisateurs standard peuvent exécuter les commandes `cryptoadm list` et `cryptoadm --help`.

  Toutes les autres sous-commandes `cryptoadm` exigent que vous preniez un rôle incluant le profil de droits Crypto Management (gestion de la cryptographie) ou que vous vous connectiez en tant que superutilisateur. Les sous-commandes telles que `disable`, `install` et `uninstall` sont disponibles pour l'administration de la structure. Pour plus d'informations, reportez-vous à la page de manuel `cryptoadm`(1M).

  La commande `svcadm` est utilisée pour gérer le démon `kcfd` et actualiser la stratégie cryptographique dans le noyau. Pour plus d'informations, reportez-vous à la page de manuel `svcadm`(1M).
- Commandes au niveau de l'utilisateur : les commandes `digest` et `mac` fournissent des services d'intégrité de fichiers. Les commandes `encrypt` et `decrypt` protègent les fichiers des risques d'écoute informatique. Pour utiliser ces commandes, reportez-vous au Tableau 3-1, “Liste des tâches Protection des fichiers avec la structure cryptographique”.

Commandes d'administration dans la structure cryptographique

La commande `cryptoadm` administre une structure cryptographique en cours d'exécution. La commande fait partie du profil de droits Crypto Management (gestion de la cryptographie). Ce

profil peut être attribué à un rôle pour l'administration sécurisée de la structure cryptographique. Vous utilisez la commande `cryptoadm` pour effectuer les opérations suivantes :

- Désactiver ou activer des mécanismes du fournisseur
- Désactiver ou activer le metaslot

La commande `svcadm` est utilisée pour activer, actualiser et désactiver le démon de services cryptographiques, `kcfd`. Cette commande fait partie de l'utilitaire de gestion des services (SMF) d'Oracle Solaris. `svc:/system/cryptosvcs` est l'instance de service pour la structure cryptographique. Pour plus d'informations, reportez-vous aux pages de manuel `smf`(5) et `svcadm`(1M).

Commandes au niveau de l'utilisateur dans la structure cryptographique

La structure cryptographique fournit des commandes au niveau de l'utilisateur pour vérifier l'intégrité des fichiers et les chiffrer/déchiffrer.

- Commande `digest` : calcule une synthèse de message pour un ou plusieurs fichiers ou pour stdin. Une synthèse permet de vérifier l'intégrité d'un fichier. SHA1 et MD5 sont des exemples de fonctions digest.
- Commande `mac` : calcule un MAC pour un ou plusieurs fichiers ou pour stdin. Un code MAC associe des données à un message authentifié. Un MAC permet à un destinataire de vérifier que le message provient de l'expéditeur et qu'il n'a pas été altéré. Les mécanismes `sha1_mac` et `md5_hmac` peuvent calculer un MAC.
- Commande `encrypt` : chiffre des fichiers ou stdin avec un chiffrement symétrique. La commande `encrypt -l` répertorie les algorithmes disponibles. Les mécanismes répertoriés dans une bibliothèque au niveau de l'utilisateur sont disponibles pour la commande `encrypt`. La structure offre les mécanismes AES, DES, 3DES (triple DES) et ARCFOUR pour le chiffrement utilisateur.
- Commande `decrypt` : déchiffre des fichiers ou stdin qui ont été chiffrés avec la commande `encrypt`. La commande `decrypt` utilise les mêmes clé et même mécanisme que ceux utilisés pour chiffrer le fichier d'origine.
- Commande `elfsign` : fournit un moyen de signer les fournisseurs à utiliser avec la structure cryptographique. En règle générale, cette commande est exécutée par le développeur d'un fournisseur. La commande `elfsign` dispose de sous-commandes pour demander un certificat, signer des fichiers binaires et vérifier la signature d'un fichier binaire. Les binaires non signés ne peuvent pas être utilisés par la structure cryptographique. Les fournisseurs qui disposent de binaires signés vérifiables peuvent utiliser la structure.

Plug-ins de la structure cryptographique

Des tiers peuvent inclure leurs fournisseurs dans la structure cryptographique. Un fournisseur tiers peut être l'un des objets suivants :

- Bibliothèque partagée PKCS #11
- Module de logiciel noyau chargeable, comme un algorithme de chiffrement, la fonction MAC ou la fonction digest
- Pilote de périphérique de noyau pour un accélérateur matériel

Les objets provenant d'un fournisseur doivent être signés avec un certificat d'Oracle. La demande de certificat se base sur une clé privée sélectionnée par le tiers et un certificat fourni par Oracle. La demande de certificat est envoyée à Oracle, qui enregistre le tiers, puis émet le certificat. Le tiers signe ensuite son objet fournisseur à l'aide du certificat Oracle.

Les modules de logiciels noyau chargeables et les pilotes de périphériques de noyau pour les accélérateurs matériels doivent également être enregistrés dans le noyau. L'enregistrement s'effectue par l'intermédiaire de l'interface du fournisseur de services (SPI) de la structure cryptographique.

Services cryptographiques et zones

La zone globale et chaque zone non globale possèdent leur propre service `/system/cryptosvc`. Lorsque le service cryptographique est activé ou actualisé dans la zone globale, le démon `kcfd` démarre dans la zone globale, et la stratégie au niveau de l'utilisateur pour la zone globale et la stratégie du noyau pour le système sont définies. Lorsque le service est activé ou actualisé dans une zone non globale, le démon `kcfd` démarre dans la zone et la stratégie au niveau de l'utilisateur pour la zone est définie. La stratégie du noyau a été définie par la zone globale.

Pour plus d'informations sur les zones, reportez-vous à “ Présentation d’Oracle Solaris Zones ”. Pour plus d'informations sur l'utilisation de SMF pour gérer les applications persistantes, reportez-vous au Chapitre 1, “ Introduction à l’utilitaire de gestion des services ” du manuel “ Gestion des services système dans Oracle Solaris 11.2 ” et à la page de manuel `smf`(5).

Structure cryptographique et FIPS 140

FIPS 140 est une norme de sécurité informatique du gouvernement américain destinée aux modules cryptographiques. Les systèmes Oracle Solaris offrent deux fournisseurs d'algorithmes cryptographiques approuvés par la norme FIPS 140-2-Niveau 1.

Ces fournisseurs sont :

- La structure cryptographique d'Oracle Solaris fournit deux modules approuvés par la norme FIPS 140. Le module *utilisateur* fournit la cryptographie aux applications exécutées dans l'espace utilisateur. Le module *noyau* fournit la cryptographie aux processus au niveau du noyau.
- Le module d'objets OpenSSL fournit la cryptographie approuvée par la norme FIPS 140 aux applications SSH et Web.

Prenez en considération les éléments clés suivants :

- Etant donné que les modules de fournisseur FIPS 140 2 peuvent nécessiter un grand nombre de CPU, ils sont désactivés par défaut. En tant qu'administrateur système, vous êtes responsable de l'autorisation des fournisseurs dans le mode FIPS 140 et de la configuration des applications qui utilisent les algorithmes approuvés par FIPS.
- Si vous êtes tenu d'utiliser uniquement la cryptographie validée par la norme FIPS 140-2, vous devez exécuter Oracle Solaris 11.1 version SRU5.5 ou Oracle Solaris 11.1 version SRU3. Dans ces deux versions, Oracle a finalisé une validation de la norme FIPS 140-2 dans la structure cryptographique Solaris. Oracle Solaris 11.2 s'appuie sur cette validation et inclut des améliorations logicielles en matière de performances, de fonctionnalités et de fiabilité. Dans la mesure du possible, vous devez configurer Solaris 11.2 en mode FIPS 140-2 pour profiter de ces améliorations.

Pour plus d'informations, reportez-vous à la section " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ". Cet article comprend les sections suivantes :

- Présentation de la cryptographie FIPS 140-2 Niveau 1 dans Oracle Solaris
- Activation des fournisseurs FIPS 140
- Activation des consommateurs FIPS 140
- Exemple d'activation de deux applications en mode FIPS 140
- Références de certificat et algorithmes approuvés par FIPS 140

Les informations supplémentaires suivantes sont disponibles :

- "Basculement vers l'implémentation d'OpenSSL conforme à la norme FIPS 140" à la page 15
- "Création d'un environnement d'initialisation compatible FIPS 140 " à la page 42

Prise en charge d'OpenSSL dans Oracle Solaris

Oracle Solaris prend en charge deux implémentations d'OpenSSL :

- OpenSSL conforme à FIPS 140
- OpenSSL non conforme à FIPS 140

Ces deux implémentations ont été mises à niveau pour être compatibles avec la version la plus récente d'OpenSSL issue du projet OpenSSL, à savoir OpenSSL 1.0.1. En ce qui concerne les bibliothèques de versions, les deux sont compatibles API/ABI.

Ces deux implémentations sont présentes dans le système d'exploitation, mais une seule implémentation peut être active à la fois. Pour déterminer quelle implémentation d'OpenSSL est active sur le système, utilisez la commande `pkg mediator openssl`.

▼ Basculement vers l'implémentation d'OpenSSL conforme à la norme FIPS 140

Par défaut, l'implémentation d'OpenSSL non conforme à la norme FIPS-140 est active dans Oracle Solaris. Toutefois, vous pouvez choisir la sécurité de votre système et sélectionner l'implémentation de votre choix.

1. **Connectez-vous en tant qu'administrateur.**

2. **Assurez-vous que les deux implémentations se trouvent sur le système.**

```
$ pkg mediator -a openssl
```

Attention - L'implémentation d'OpenSSL vers laquelle vous basculez doit exister dans le système. Dans le cas contraire, si vous basculez vers une implémentation qui ne figure pas dans le système, le système risque de devenir inutilisable.

3. **Basculez vers une autre implémentation d'OpenSSL.**

```
# pkg set-mediator [--be-name name] -I implementation openssl
```

où *implementation* est soit `default` soit `fips-140` et où *name* est un nom de clone de l'environnement d'initialisation actuel. L'implémentation spécifiée sera active sur le clone.

Remarque - Lorsque l'option *--be-name* est spécifiée, la commande créé une sauvegarde de l'environnement d'initialisation actuel. Lorsque vous effectuez une réinitialisation, le système exécute le nouvel environnement d'initialisation cloné contenant la nouvelle implémentation.

Pour plus d'informations sur la commande `pkg set-mediator`, reportez-vous à la section " Modification de l'application préférée " du manuel " Ajout et mise à jour de logiciels dans Oracle Solaris 11.2 ".

4. **Réinitialisez le système.**

5. **(Facultatif) Vérifiez que le basculement a réussi et que l'implémentation d'OpenSSL que vous vouliez est active.**

```
# pkg mediator openssl
```

Exemple 1-1 Basculement vers l'implémentation d'OpenSSL conforme à la norme FIPS 140

Cet exemple change l'implémentation d'OpenSSL dans un système pour la rendre conforme à la norme FIPS-140.

```
# pkg mediator -a openssl
MEDIATOR   VER. SRC.   VERSION IMPL.   SRC. IMPLEMENTATION
openssl       vendor           vendor               default
openssl       system          system              fips-140

# pkg set-mediator --be-name BE2 -I fips-140 openssl
# reboot

# pkg mediator openssl
MEDIATOR   VER. SRC.   VERSION IMPL.   SRC. IMPLEMENTATION
openssl      vendor           vendor              default
```

♦ ♦ ♦ CHAPITRE 2

A propos des systèmes de série SPARC T et de la structure cryptographique

Ce chapitre décrit la structure cryptographique sur les serveurs de série SPARC T et les optimisations intégrées dans Oracle Solaris 11 qui améliorent les performances des fonctions cryptographiques.

Structure cryptographique et serveurs de série SPARC T

La structure cryptographique fournit aux systèmes SPARC T des mécanismes cryptographiques et optimise certains mécanismes pour ces serveurs. Ces mécanismes cryptographiques sont optimisés pour les données au repos et en transit : `AES-CBC`, `AES-CFB128` et `ARCFOUR`. Le mécanisme cryptographique DES est optimisé pour OpenSSL et, en optimisant l'arithmétique multiprécision (bignum), RSA et DSA sont également optimisés. Les autres optimisations incluent des performances de petits paquets pour les établissements de connexion et les données en transit.

Les mécanismes cryptographiques suivants sont disponibles dans cette version :

- `AES-XTS` : utilisé pour les données au repos
- `SHA-224` : mécanisme `SHA2`
- `AES-XCBC-MAC` : utilisé pour IPsec

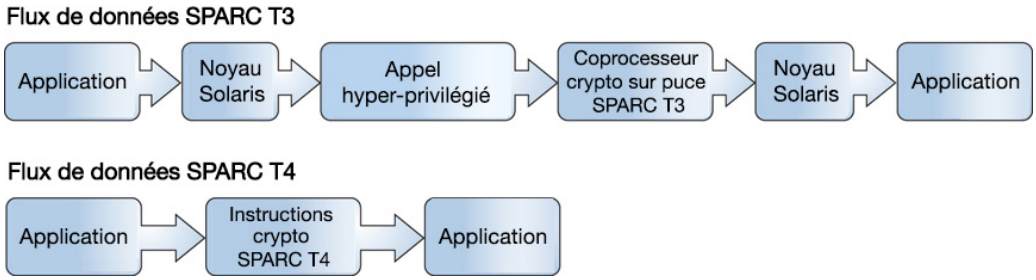
Optimisations cryptographiques dans les systèmes SPARC T-4

Depuis la mise sur le marché du microprocesseur SPARC T4, les nouvelles instructions permettant de tirer profit des fonctions cryptographiques sont disponibles directement dans le matériel. Ces instructions ne nécessitent pas de privilèges. Ainsi, tous les programmes peuvent utiliser ces instructions sans être tenus de posséder un environnement de noyau quelconque, des autorisations root ni une autre configuration spéciale. La cryptographie est

effectuée directement sur le matériel au lieu d'utiliser de nombreuses instructions de bas niveau. Par conséquent, les opérations de cryptographie sont plus rapides que les opérations sur les systèmes dont les processeurs SPARC précédents possédaient des unités centrales distinctes pour la cryptographie.

La comparaison suivante montre les différences dans le flux de données entre des systèmes SPARC T-3 et des systèmes SPARC T-4 avec optimisations cryptographiques.

FIGURE 2-1 Comparaison du flux de données entre des systèmes SPARC T



Le tableau suivant fournit une comparaison détaillée des fonctions cryptographiques dans des unités à microprocesseurs SPARC T associées à des versions d'Oracle Solaris spécifiques.

TABLEAU 2-1 Performances cryptographiques sur des serveurs de série SPARC T

Consommateur de fonctionnalités/ logiciels	Systèmes T-3 et antérieurs	Systèmes T-4 exécutant Oracle Solaris 10	Systèmes T-4 exécutant Oracle Solaris 11
SSH	Activée automatiquement avec Solaris 10 5/09 et versions ultérieures. Désactivez/activez à l'aide de la clause `UseOpenSSLEngine` dans `/etc/ssh/sshd_config`.	Patch 147707-01 requis. Désactivez/activez à l'aide de la clause `UseOpenSSLEngine` dans `/etc/ssh/sshd_config`.	Activée automatiquement. Désactivez/activez à l'aide de la clause `UseOpenSSLEngine` dans `/etc/ssh/sshd_config`.
Java/JCE	Activée automatiquement. Configurez dans `$JAVA_HOME/jre/lib/security/java.security`.	Activée automatiquement. Configurez dans `$JAVA_HOME/jre/lib/security/java.security`.	Activée automatiquement. Configurez dans `$JAVA_HOME/jre/lib/security/java.security`.
Cryptographie ZFS	Non disponible.	Non disponible.	La cryptographie HW est activée automatiquement si le jeu de données est chiffré.
IPsec	Activée automatiquement.	Activée automatiquement.	Activée automatiquement.

Consommateur de fonctionnalités/ logiciels	Systèmes T-3 et antérieurs	Systèmes T-4 exécutant Oracle Solaris 10	Systèmes T-4 exécutant Oracle Solaris 11
OpenSSL	Utilisez `-engine pkcs11`	Patch 147707-01 requis Utilisez `-engine pkcs11`	L'optimisation T4 est utilisée automatiquement. (Utilisez éventuellement `-engine pkcs11`.) `pkcs11` est recommandé pour RSA/DSA à ce stade.
KSSL (proxy Kernel SSL)	Activée automatiquement.	Activée automatiquement.	Activée automatiquement.
Oracle TDE	Non prise en charge.	Patch en attente.	Activée automatiquement avec Oracle DB 11.2.0.3 et ASO.
Apache SSL	Configurez avec `SSLCrypto Device pkcs11`	Configurez avec `SSLCrypto Device pkcs11`	Configurez avec `SSLCrypto Device pkcs11`
Domaines logiques	Affectez des unités de chiffrement aux domaines.	Fonctionnalité toujours disponible, aucune configuration requise.	Fonctionnalité toujours disponible, aucune configuration requise.

Les instructions de chiffrement T4 incluent ce qui suit :

- `aes_kexpand0`, `aes_kexpand1`, `aes_kexpand2`

  Ces instructions permettent d'effectuer une *extension de clé*. Elles permettent d'étendre la clé 128 bits, 192 bits ou 256 bits fournie par l'utilisateur en une planification de clé utilisée en interne au cours du chiffrement et du déchiffrement. L'instruction `aes_kexpand2` est utilisée uniquement pour AES-256. Les deux autres instructions `aes_kexpand` sont utilisées pour les trois longueurs de clé : AES-128, AES-192 et AES-256.

- `aes_eround01`, `aes_eround23`, `aes_eround01_l`, `aes_eround_23_l`

  Ces instructions sont utilisées pour les *cycles* de chiffrement AES ou les transformations. Conformément au standard AES dans FIPS 197, le nombre de cycles utilisés (par exemple 10, 12 ou 14) varie en fonction de la longueur de clé AES, car l'utilisation de clés de plus grande taille indique le souhait d'un chiffrement plus puissant, au prix de calculs supplémentaires.

- `aes_dround01`, `aes_dround23`, `aes_dround01_l`, `aes_dround_23_l`

  Ces instructions sont utilisées pour les cycles de déchiffrement AES de façon similaire à leur utilisation pour le chiffrement.

- Instructions pour DES/DES-3, Kasumi, Camellia, la racine carrée/multiplication de Montgomery (pour RSA Bignum) et les sommes de contrôle CRC32c
- Instructions de synthèse MD5, SHA1 et SHA2

Les instructions cryptographiques du matériel SPARC T4 sont disponibles et utilisées automatiquement sur les systèmes SPARC T4 qui exécutent Oracle Solaris 11 par le biais du moteur t4 intégré sur le microprocesseur T4 du système. A partir de Oracle Solaris 11.2, ces instructions sont désormais intégrées dans le code en amont OpenSSL. Ainsi, dans cette version, OpenSSL 1.0.1e est fourni avec un patch lui permettant d'utiliser ces instructions.

Pour plus d'informations sur les instructions T4, reportez-vous aux articles suivants.

- "SPARC T4 OpenSSL Engine" (https://blogs.oracle.com/DanX/entry/sparc_t4_openssl_engine)
- "How to tell if SPARC T4 crypto is being used?" (https://blogs.oracle.com/DanX/entry/how_to_tell_if_sparc)
- "Exciting Crypto Advances with the T4 processor and Oracle Solaris 11" (http://bubbva.blogspot.com/2011/11/exciting-crypto-advances-with-t4.html)
- "SPARC T4 Digest and Crypto Optimizations in Solaris 11.1" (https://blogs.oracle.com/DanX/entry/sparc_t4_digest_and_crypto)

Déterminer si le système prend en charge les optimisations SPARC T4

Pour déterminer si les optimisations T4 sont utilisées, utilisez la commande `isainfo`. L'inclusion de `sparcv9` et d'`aes` dans la sortie indique que le système utilise les optimisations.

```
$ isainfo -v
64-bit sparcv9 applications
        crc32c cbcond pause mont mpmul sha512 sha256 sha1 md5 camellia kasumi
        des aes ima hpc vis3 fmaf asi_blk_init vis2 vis popc
```

Identification de la version OpenSSL de votre système

Pour vérifier la version d'OpenSSL qui est en cours d'exécution sur votre système, saisissez `openssl version`. La sortie se présente de la manière suivante :

```
OpenSSL 1.0.0j 10 May 2012
```

Vérification que votre système possède OpenSSL avec les optimisations SPARC T4

Pour déterminer si votre système prend en charge OpenSSL avec les optimisations SPARC T4, consultez la bibliothèque `libcrypto.so` comme suit :

```
# nm /lib/libcrypto.so.1.0.0 | grep des_t4
[5239]  |    504192|       300|FUNC |GLOB |3    |12     |des_t4_cbc_decrypt
[5653]  |    503872|       300|FUNC |GLOB |3    |12     |des_t4_cbc_encrypt
[4384]  |    505024|       508|FUNC |GLOB |3    |12     |des_t4_ede3_cbc_decrypt
[2963]  |    504512|       508|FUNC |GLOB |3    |12     |des_t4_ede3_cbc_encrypt
[4111]  |    503712|       156|FUNC |GLOB |3    |12     |des_t4_key_expand
```

Si la commande ne génère aucune sortie, votre système ne prend pas en charge les optimisations SPARC T4 pour OpenSSL.

♦ ♦ ♦ CHAPITRE 3

Structure cryptographique

Ce chapitre décrit l'utilisation de la structure cryptographique et traite des sujets suivants :

Protection des fichiers avec la structure cryptographique

Cette section décrit la génération des clés symétriques, la création des sommes de contrôle pour l'intégrité des fichiers et la protection des fichiers contre les risques d'écoute informatique. Les commandes de cette section peuvent être exécutées par des utilisateurs standard. Les développeurs peuvent écrire des scripts qui utilisent ces commandes.

Pour configurer votre système en mode FIPS 140, vous devez utiliser des algorithmes, des modes et des longueurs de clé validés par FIPS. Voir la section “ FIPS 140 Algorithm Lists and Certificate References for Oracle Solaris Systems ” du manuel “ Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ”.

La structure cryptographique peut vous aider à protéger vos fichiers. La liste des tâches suivante présente les procédures permettant de dresser la liste des algorithmes disponibles et de protéger des fichiers par cryptographie.

TABLEAU 3-1 Liste des tâches Protection des fichiers avec la structure cryptographique

Tâche	Description	Voir
Génération d'une clé symétrique	Génère une clé de la longueur définie par l'utilisateur. Stocke éventuellement la clé dans un fichier, un keystore PKCS #11 ou un keystore NSS. Pour configurer le mode approuvé par FIPS 140, sélectionnez un type, un mode et une longueur de clé validés pour FIPS. Voir la section “ FIPS 140 Algorithms in the Cryptographic Framework ” du manuel “ Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ”.	“Génération d'une clé symétrique à l'aide de la commande `pktool`” à la page 22

Tâche	Description	Voir
Calcul d'une somme de contrôle assurant l'intégrité d'un fichier.	Vérifie que l'exemplaire d'un fichier reçu par le destinataire est identique au fichier qui a été envoyé.	“Calcul d'une synthèse d'un fichier” à la page 28
Protection d'un fichier avec un code d'authentification des messages (MAC).	Atteste au destinataire de votre message que vous en êtes l'expéditeur.	“Calcul du code MAC d'un fichier” à la page 29
Chiffrement d'un fichier, puis déchiffrement du fichier chiffré.	Protège le contenu d'un fichier en chiffrant le fichier. Fournit les paramètres de chiffrement pour déchiffrer le fichier.	“Chiffrement et déchiffrement d'un fichier” à la page 31

▼ Génération d'une clé symétrique à l'aide de la commande `pktool`

Certaines applications exigent une clé symétrique pour le chiffrement et le déchiffrement des communications. Dans cette procédure, vous créez une clé symétrique et la stockez.

Si votre site dispose d'un générateur de nombres aléatoires, vous pouvez l'utiliser pour créer un nombre aléatoire pour la clé. Cette procédure n'utilise pas le générateur de nombres aléatoires de votre site Web.

1. **(Facultatif) Si vous prévoyez d'utiliser un keystore, créez-le.**

   - **Pour créer et initialiser un keystore PKCS #11, reportez-vous à la section “Génération d'une phrase de passe à l'aide de la commande `pktool setpin`” à la page 61.**

   - **Pour créer et initialiser une base de données NSS, reportez-vous à l'Exemple 4-5, “Protection d'un keystore par une phrase de passe”.**

2. **Générez un nombre aléatoire pour l'utiliser comme clé symétrique.**
   Choisissez l'une des méthodes suivantes.

   - **Générez une clé et stockez-la dans un fichier.**
     L'avantage d'une clé stockée dans un fichier est que vous pouvez extraire la clé de ce fichier pour l'utiliser dans le fichier de clés d'une application, tel que le fichier `/etc/inet/secret/ipseckeys` ou IPsec. L'instruction d'utilisation indique les arguments.

     ```
     % pktool genkey keystore=file
     ...genkey keystore=file
     outkey=key-fn
     [ keytype=aes|arcfour|des|3des|generic ]
     [ keylen=key-size (AES, ARCFOUR or GENERIC only)]
     ```

```
[ print=y|n ]
```

outkey=*key-fn*

Nom de fichier contenant la clé.

keytype=*specific-symmetric-algorithm*

Pour une clé symétrique d'une longueur quelconque, la valeur est `generic`. Pour un algorithme spécifique, indiquez `aes`, `arcfour`, `des` ou `3des`.

Pour configurer des algorithmes approuvés par FIPS 140, sélectionnez un type de clé validé pour FIPS. Voir la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

keylen=*size-in-bits*

Longueur de la clé en bits. Le nombre doit être divisible par 8. *Ne spécifiez rien* pour `des` ou `3des`.

Pour configurer des algorithmes approuvés par FIPS 140, sélectionnez une longueur de clé validée pour FIPS. Voir la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

print=n

Imprime la clé de la fenêtre de terminal. Par défaut, la valeur de `print` est `n`.

- **Générez une clé et stockez-la dans un keystore PKCS #11.**

L'avantage du keystore PKCS #11 est que vous pouvez extraire la clé par son étiquette. Cette méthode est utile pour les clés qui chiffrent et déchiffrent des fichiers. Vous devez effectuer l'Étape 1 avant d'utiliser cette méthode. L'instruction d'utilisation indique les arguments. Les crochets qui entourent l'argument de keystore indiquent que lorsqu'il n'est pas spécifié, la clé est stockée dans le keystore PKCS #11.

```
$ pktool genkey keystore=pkcs11
...genkey [ keystore=pkcs11 ]
label=key-label
[ keytype=aes|arcfour|des|3des|generic ]
[ keylen=key-size (AES, ARCFOUR or GENERIC only)]
[ token=token[:manuf[:serial]]]
[ sensitive=y|n ]
[ extractable=y|n ]
[ print=y|n ]
```

label=*key-label*

Etiquette spécifiée par l'utilisateur pour la clé. La clé peut être récupérée à partir du keystore par son étiquette.

`keytype=`*specific-symmetric-algorithm*

Pour une clé symétrique d'une longueur quelconque, la valeur est `generic`. Pour un algorithme spécifique, indiquez `aes`, `arcfour`, `des` ou `3des`.

Pour configurer des algorithmes approuvés par FIPS 140, sélectionnez un type de clé validé pour FIPS. Voir la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

`keylen=`*size-in-bits*

Longueur de la clé en bits. Le nombre doit être divisible par 8. *Ne spécifiez rien* pour `des` ou `3des`.

Pour configurer des algorithmes approuvés par FIPS 140, sélectionnez une longueur de clé validée pour FIPS. Voir la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

`token=`*token*

Nom du jeton. Par défaut, le jeton est `Sun Software PKCS#11 softtoken`.

`sensitive=n`

Détermine la sensibilité de la clé. Lorsque la valeur est `y`, la clé ne peut pas être imprimée à l'aide de l'argument `print=y`. Par défaut, la valeur de `sensitive` est `n`.

`extractable=y`

Indique que la clé peut être extraite du keystore. Spécifiez `n` afin d'empêcher l'extraction de la clé.

`print=n`

Imprime la clé de la fenêtre de terminal. Par défaut, la valeur de `print` est `n`.

- **Générez une clé et stockez-la dans un keystore NSS.**

  Vous devez effectuer l'Étape 1 avant d'utiliser cette méthode. L'instruction d'utilisation indique les arguments.

```
$ pktool genkey keystore=nss
...genkey keystore=nss
label=key-label
[ keytype=aes|arcfour|des|3des|generic ]
[ keylen=key-size (AES, ARCFOUR or GENERIC only)]
[ token=token[:manuf[:serial]]]
[ dir=directory-path ]
[ prefix=DBprefix ]
```

`label=`*key-label*

Etiquette spécifiée par l'utilisateur pour la clé. La clé peut être récupérée à partir du keystore par son étiquette.

`keytype=`*specific-symmetric-algorithm*

Pour une clé symétrique d'une longueur quelconque, la valeur est `generic`. Pour un algorithme spécifique, indiquez `aes`, `arcfour`, `des` ou `3des`.

Pour configurer des algorithmes approuvés par FIPS 140, sélectionnez un type de clé validé pour FIPS. Voir la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

`keylen=`*size-in-bits*

Longueur de la clé en bits. Le nombre doit être divisible par 8. *Ne spécifiez rien* pour `des` ou `3des`.

Pour configurer des algorithmes approuvés par FIPS 140, sélectionnez une longueur de clé validée pour FIPS. Voir la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

`token=`*token*

Nom du jeton. Par défaut, le jeton est le jeton interne NSS.

`dir=`*directory*

Chemin d'accès au répertoire de la base de données NSS. Par défaut, *directory* est le répertoire courant.

`prefix=`*directory*

Préfixe de la base de données NSS. Par défaut, le champ de préfixe est vide.

3. (Facultatif) Vérifiez que la clé existe.

Utilisez l'une des commandes suivantes, en fonction de l'endroit où vous avez stocké la clé.

- **Vérifiez la clé dans le fichier *key-fn*.**

```
% pktool list keystore=file objtype=key [infile=key-fn]
Found n keys.
Key #1 - keytype:location (keylen)
```

- **Vérifiez la clé dans le keystore PKCS #11 ou NSS.**

```
For PKCS #11, use the following command:

$ pktool list keystore=pkcs11 objtype=key
Enter PIN for keystore:
Found n keys.
```

`Key #1 -` *keytype*`:`*location (keylen)*

Sinon, remplacez `keystore=pkcs11` par `keystore=nss` dans la commande.

Exemple 3-1 Création d'une clé symétrique à l'aide de la commande `pktool`

Dans l'exemple suivant, un utilisateur crée un keystore PKCS#11 pour la première fois, puis génère une longue clé symétrique pour une application. Enfin, l'utilisateur vérifie que la clé se trouve dans le keystore.

Notez que le mot de passe d'origine pour un keystore PKCS #11 est `changeme`. Le mot de passe initial pour un keystore NSS est un mot de passe vide.

```
# pktool setpin
Create new passphrase:      Type password
Re-enter new passphrase:      Retype password
Passphrase changed.
% pktool genkey label=specialappkey keytype=generic keylen=1024
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password

% pktool list objtype=key
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
No.      Key Type      Key Len.      Key Label
-----------------------------------------------------
Symmetric keys:
1        Symmetric     1024          specialappkey
```

Exemple 3-2 Création d'une clé approuvée par FIPS à l'aide de la commande `pktool`

Dans l'exemple suivant, une clé secrète pour l'algorithme AES est créée à l'aide d'un algorithme et d'une longueur de clé approuvés par FIPS. La clé est stockée dans un fichier local pour un déchiffrement ultérieur. La commande protège le fichier avec `400` autorisations. Si la clé est créée, l'option `print=y` affiche la clé générée dans la fenêtre de terminal.

L'utilisateur propriétaire du fichier de clés récupère la clé à l'aide de la commande `od`.

```
% pktool genkey keystore=file outkey=256bit.file1 keytype=aes keylen=256 print=y
Key Value ="aaa2df1d10f02eaee2595d48964847757a6a49cf86c4339cd5205c24ac8c8873"
% od -x 256bit.file1

0000000 aaa2 df1d 10f0 2eae e259 5d48 9648 4775
0000020 7a6a 49cf 86c4 339c d520 5c24 ac8c 8873
0000040
```

Exemple 3-3 Création d'une clé symétrique pour les associations de sécurité (SA) IPsec

Dans l'exemple suivant, l'administrateur crée manuellement les numéros de clé pour les SA IPsec et les stocke dans des fichiers. Ensuite, l'administrateur copie les clés pour le fichier `/etc/inet/secret/ipseckeys` et détruit les fichiers d'origine.

Tout d'abord, l'administrateur crée et affiche les clés requises par la stratégie IPsec :

```
# pktool genkey keystore=file outkey=ipencrin1 keytype=generic keylen=192 print=y
Key Value ="294979e512cb8e79370dabecadc3fcbb849e78d2d6bd2049"
# pktool genkey keystore=file outkey=ipencrout1 keytype=generic keylen=192 print=y
Key Value ="9678f80e33406c86e3d1686e50406bd0434819c20d09d204"
# pktool genkey keystore=file outkey=ipspi1 keytype=generic keylen=32 print=y
Key Value ="acbeaa20"
# pktool genkey keystore=file outkey=ipspi2 keytype=generic keylen=32 print=y
Key Value ="19174215"
# pktool genkey keystore=file outkey=ipsha21 keytype=generic keylen=256 print=y
Key Value ="659c20f2d6c3f9570bcee93e96d95e2263aca4eeb3369f72c5c786af4177fe9e"
# pktool genkey keystore=file outkey=ipsha22 keytype=generic keylen=256 print=y
Key Value ="b041975a0e1fce0503665c3966684d731fa3dbb12fcf87b0a837b2da5d82c810"
```

Ensuite, l'administrateur crée le fichier suivant `/etc/inet/secret/ipseckeys` :

```
##   SPI values require a leading 0x.
##   Backslashes indicate command continuation.
##
## for outbound packets on this system
add esp spi 0xacbeaa20 \
src 192.168.1.1 dst 192.168.2.1 \
encr_alg aes auth_alg sha256  \
encrkey  294979e512cb8e79370dabecadc3fcbb849e78d2d6bd2049 \
authkey  659c20f2d6c3f9570bcee93e96d95e2263aca4eeb3369f72c5c786af4177fe9e
##
## for inbound packets
add esp spi 0x19174215 \
src 192.168.2.1 dst 192.168.1.1 \
encr_alg aes auth_alg sha256  \
encrkey 9678f80e33406c86e3d1686e50406bd0434819c20d09d204 \
authkey b041975a0e1fce0503665c3966684d731fa3dbb12fcf87b0a837b2da5d82c810
```

Après avoir vérifié que la syntaxe du fichier `ipseckeys` est valide, l'administrateur détruit les fichiers clé d'origine.

```
# ipseckey -c /etc/inet/secret/ipseckeys
# rm ipencrin1 ipencrout1 ipspi1 ipspi2 ipsha21 ipsha22
```

L'administrateur copie le fichier `ipseckeys` au système de communication en utilisant la commande `ssh` ou un autre mécanisme sécurisé. Sur le système de communication, les protections sont inversées. La première entrée dans le fichier `ipseckeys` protège les paquets entrants, et la seconde entrée protège les paquets sortants. Aucune clé n'est générée sur le système de communication.

Étapes suivantes Pour continuer à utiliser la clé afin de créer un code d'authentification des messages (MAC) pour un fichier, voir “Calcul du code MAC d'un fichier” à la page 29.

▼ Calcul d'une synthèse d'un fichier

Lorsque vous calculez la synthèse d'un fichier, vous pouvez vérifier que le fichier n'a pas été altéré en comparant les résultats de la synthèse. Une synthèse n'altère pas le fichier d'origine.

1. Répertoriez les algorithmes de synthèse disponibles.

```
% digest -l
md5
sha1
sha224
sha256
sha384
sha512
```

Remarque - Dans la mesure du possible, sélectionnez un algorithme approuvé par FIPS dans la liste de la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

2. Calculez la synthèse du fichier et enregistrez la liste des synthèses.

Fournissez un algorithme avec la commande `digest`.

```
% digest -v -a algorithm input-file > digest-listing
```

`-v` — Affiche la sortie au format suivant :

algorithm (*input-file*) = *digest*

`-a` *algorithm* — Algorithme à utiliser pour calculer une synthèse du fichier. Saisissez l'algorithme lorsqu'il s'affiche dans la sortie de l'Étape 1.

Remarque - Dans la mesure du possible, sélectionnez un algorithme approuvé par FIPS et répertorié dans la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

input-file — Fichier d'entrée de la commande `digest`.

digest-listing — Fichier de sortie de la commande `digest`.

Exemple 3-4 Calcul d'une synthèse avec le mécanisme SHA1

Dans l'exemple suivant, la commande `digest` utilise le mécanisme SHA1 pour fournir une liste des répertoires. Les résultats sont placés dans un fichier.

```
% digest -v -a sha1 docs/* > $HOME/digest.docs.legal.05.07
```

```
% more ~/digest.docs.legal.05.07
sha1 (docs/legal1) = 1df50e8ad219e34f0b911e097b7b588e31f9b435
sha1 (docs/legal2) = 68efa5a636291bde8f33e046eb33508c94842c38
sha1 (docs/legal3) = 085d991238d61bd0cfa2946c183be8e32cccf6c9
sha1 (docs/legal4) = f3085eae7e2c8d008816564fdf28027d10e1d983
```

▼ Calcul du code MAC d'un fichier

Un code d'authentification des messages, ou MAC, calcule la synthèse pour le fichier et utilise une clé secrète pour protéger davantage cette synthèse. Un code MAC n'altère pas le fichier d'origine.

1. **Répertoriez les mécanismes disponibles.**

```
% mac -l
Algorithm       Keysize:  Min   Max
---------------------------------
des_mac                   64    64
sha1_hmac                  8   512
md5_hmac                   8   512
sha224_hmac                8   512
sha256_hmac                8   512
sha384_hmac                8  1024
sha512_hmac                8  1024
```

Remarque - Chaque algorithme pris en charge est un alias de la version la plus couramment utilisée et la moins limitée d'un type d'algorithme particulier. La sortie ci-dessus indique les noms d'algorithme disponibles et la taille de clé de chaque algorithme. Dans la mesure du possible, utilisez un algorithme correspondant à l'algorithme approuvé par FIPS avec une longueur de clé approuvée par FIPS, tel que répertorié dans la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

2. **Générez une clé symétrique de la longueur appropriée.**

   Vous pouvez indiquer une phrase de passe à partir de laquelle une clé sera générée ou vous pouvez fournir une clé.

   - Si vous fournissez une phrase de passe, vous devez la stocker ou la mémoriser. Si vous la stockez en ligne, le fichier de la phrase de passe ne doit être lisible que par vous.
   - Si vous fournissez une clé, elle doit avoir la taille correcte pour le mécanisme. Vous pouvez utiliser la commande `pktool`. Pour plus d'informations sur cette procédure et des exemples, reportez-vous à la section "Génération d'une clé symétrique à l'aide de la commande `pktool`" à la page 22.

3. **Créez un code MAC pour un fichier.**

Fournissez une clé et utilisez un algorithme de clé symétrique avec la commande `mac`.

```
% mac [-v] -a algorithm [-k keyfile | -K key-label [-T token]] input-file
```

`-v`	Affiche la sortie au format suivant : *algorithm* (*input-file*) = *mac*
`-a` *algorithm*	Algorithme à utiliser pour calculer le code MAC. Saisissez l'algorithme lorsqu'il s'affiche dans la sortie de la commande `mac -l`.
`-k` *keyfile*	Fichier contenant une clé de longueur spécifiée par algorithme.
`-K` *key-label*	Etiquette d'une clé dans le keystore PKCS #11.
`-T` *token*	Nom du jeton. Par défaut, le jeton est `Sun Software PKCS#11 softtoken`. Il est utilisé uniquement lorsque l'option *-Kkey-label* est utilisée.
input-file	Fichier d'entrée pour le MAC.

Exemple 3-5 Calcul d'un MAC avec SHA1_HMAC et une phrase de passe

Dans l'exemple suivant, la pièce jointe d'e-mail est authentifiée avec le mécanisme SHA1_HMAC et une clé dérivée d'une phrase de passe. La liste MAC est enregistrée dans un fichier. Si la phrase de passe est stockée dans un fichier, celui-ci doit être lisible uniquement par l'utilisateur.

```
% mac -v -a sha1_hmac email.attach
Enter passphrase:      Type passphrase
sha1_hmac (email.attach) = 2b31536d3b3c0c6b25d653418db8e765e17fe07b
% echo "sha1_hmac (email.attach) = 2b31536d3b3c0c6b25d653418db8e765e17fe07b" \
>> ~/sha1hmac.daily.05.12
```

Exemple 3-6 Calcul d'un MAC avec SHA1_HMAC et un fichier de clés

Dans l'exemple suivant, le manifeste de répertoire est authentifié avec le mécanisme SHA1_HMAC et une clé secrète. Les résultats sont placés dans un fichier.

```
% mac -v -a sha1_hmac \
-k $HOME/keyf/05.07.mack64 docs/* > $HOME/mac.docs.legal.05.07
% more ~/mac.docs.legal.05.07
sha1_hmac (docs/legal1) = 9b31536d3b3c0c6b25d653418db8e765e17fe07a
sha1_hmac (docs/legal2) = 865af61a3002f8a457462a428cdb1a88c1b51ff5
sha1_hmac (docs/legal3) = 076c944cb2528536c9aebd3b9fbe367e07b61dc7
sha1_hmac (docs/legal4) = 7aede27602ef6e4454748cbd3821e0152e45beb4
```

Exemple 3-7 Calcul d'un MAC avec SHA1_HMAC et une étiquette de clés

Dans l'exemple suivant, le manifeste de répertoire est authentifié avec le mécanisme SHA1_HMAC et une clé secrète. Les résultats sont placés dans le keystore PKCS #11 de l'utilisateur. L'utilisateur a initialement créé le keystore et le mot de passe pour le keystore à l'aide de la commande `pktool setpin`.

```
% mac -a sha1_hmac -K legaldocs0507 docs/*
Enter pin for Sun Software PKCS#11 softtoken:     Type password
```

Pour récupérer le MAC à partir du keystore, l'utilisateur se sert des informations détaillées et fournit l'étiquette clé et le nom du répertoire qui a été authentifié.

```
% mac -v -a sha1_hmac -K legaldocs0507  docs/*
Enter pin for Sun Software PKCS#11 softtoken:     Type password
sha1_hmac (docs/legal1) = 9b31536d3b3c0c6b25d653418db8e765e17fe07a
sha1_hmac (docs/legal2) = 865af61a3002f8a457462a428cdb1a88c1b51ff5
sha1_hmac (docs/legal3) = 076c944cb2528536c9aebd3b9fbe367e07b61dc7
sha1_hmac (docs/legal4) = 7aede27602ef6e4454748cbd3821e0152e45beb4
```

▼ Chiffrement et déchiffrement d'un fichier

Lorsque vous chiffrez un fichier, le fichier d'origine n'est ni supprimé, ni modifié. Le fichier de sortie est chiffré.

Pour trouver des solutions aux erreurs courantes relatives à la commande `encrypt`, reportez-vous à la section après les exemples.

Remarque - En cas de chiffrement et de déchiffrement de fichiers, essayez d'utiliser des algorithmes approuvés par FIPS avec des longueurs de clé approuvées aussi souvent que possible. Reportez-vous à la liste de la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ". Exécutez la commande `encrypt -l` pour afficher les algorithmes disponibles et leurs longueurs de clé.

---

1. **Créez une clé symétrique de la longueur appropriée.**

   Vous pouvez indiquer une phrase de passe à partir de laquelle une clé sera générée ou vous pouvez fournir une clé.

   - Si vous fournissez une phrase de passe, vous devez stocker ou mémoriser la phrase de passe. Si vous la stockez en ligne, le fichier de la phrase de passe ne doit être lisible que par vous.
   - Si vous fournissez une clé, elle doit avoir la taille correcte pour le mécanisme. Vous pouvez utiliser la commande `pktool`. Pour plus d'informations sur cette procédure et des exemples, reportez-vous à la section "Génération d'une clé symétrique à l'aide de la commande `pktool`" à la page 22.

2. Chiffrez un fichier.

Fournissez une clé et utilisez un algorithme de clé symétrique avec la commande `encrypt`.

```
% encrypt -a algorithm [-v] \
[-k keyfile | -K key-label [-T token]] [-i input-file] [-o output-file]
```

-a *algorithm*	Algorithme à utiliser pour chiffrer le fichier. Saisissez l'algorithme lorsqu'il s'affiche dans la sortie de la commande `encrypt -l`. Dans la mesure du possible, sélectionnez un algorithme approuvé par FIPS dans la liste de la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".
-k *keyfile*	Fichier contenant une clé de longueur spécifiée par algorithme. La longueur de la clé pour chaque algorithme est répertoriée, en bits, dans la sortie de la commande `encrypt -l`.
-K *key-label*	Etiquette d'une clé dans le keystore PKCS #11.
-T *token*	Nom du jeton. Par défaut, le jeton est `Sun Software PKCS#11 softtoken`. Il est utilisé uniquement lorsque l'option -K*key-label* est utilisée.
-i *input-file*	Fichier d'entrée que vous voulez chiffrer. Ce fichier n'est pas modifié par la commande.
-o *output-file*	Fichier de sortie correspondant à la forme chiffrée du fichier d'entrée.

Exemple 3-8 Création d'une clé AES pour le chiffrement de vos fichiers

Dans l'exemple suivant, un utilisateur crée et stocke une clé AES dans keystore PKCS #11 existant pour l'utilisation lors du chiffrement et du déchiffrement. L'utilisateur peut vérifier que la clé existe et l'utiliser, mais il ne peut pas l'afficher.

```
% pktool genkey label=MyAESkeynumber1 keytype=aes keylen=256
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password

% pktool list objtype=key
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
No.      Key Type      Key Len.      Key Label
-------------------------------------------------
Symmetric keys:
1        AES           256           MyAESkeynumber1
```

Pour utiliser la clé pour chiffrer un fichier, l'utilisateur le récupère la clé par son étiquette.

```
% encrypt -a aes -K MyAESkeynumber1 -i encryptthisfile -o encryptedthisfile
```

Pour déchiffrer le `encryptedthisfile`, l'utilisateur récupère la clé par son étiquette.

```
% decrypt -a aes -K MyAESkeynumber1 -i encryptedthisfile -o sameasencryptthisfile
```

Exemple 3-9 Chiffrement et déchiffrement avec AES et une phrase de passe

Dans l'exemple suivant, un fichier est chiffré avec l'algorithme AES. La clé est générée à partir de la phrase de passe. Si la phrase de passe est stockée dans un fichier, celui-ci doit être lisible uniquement par l'utilisateur.

```
% encrypt -a aes -i ticket.to.ride -o ~/enc/e.ticket.to.ride
Enter passphrase:         Type passphrase
Re-enter passphrase:      Type passphrase again
```

Le fichier d'entrée, `ticket.to.ride`, existe toujours sous sa forme d'origine.

Pour déchiffrer le fichier de sortie, l'utilisateur utilise la même phrase de passe et le même mécanisme de chiffrement que ceux utilisés pour le chiffrement du fichier.

```
% decrypt -a aes -i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
Enter passphrase:      Type passphrase
```

Exemple 3-10 Chiffrement et déchiffrement avec AES et un fichier de clés

Dans l'exemple suivant, un fichier est chiffré avec l'algorithme AES. Les mécanismes AES utilisent une clé de 128 bits, ou 16 octets.

```
% encrypt -a aes -k ~/keyf/05.07.aes16 \
-i ticket.to.ride -o ~/enc/e.ticket.to.ride
```

Le fichier d'entrée, `ticket.to.ride`, existe toujours sous sa forme d'origine.

Pour déchiffrer le fichier de sortie, l'utilisateur utilise la même clé et le même mécanisme de chiffrement que ceux utilisés pour le chiffrement.

```
% decrypt -a aes -k ~/keyf/05.07.aes16  \
-i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
```

Erreurs fréquentes

Les messages suivants indiquent que la clé que vous avez fournie à la commande `encrypt` n'est pas autorisée par l'algorithme que vous utilisez.

- `encrypt: unable to create key for crypto operation: CKR_ATTRIBUTE_VALUE_INVALID`
- `encrypt: failed to initialize crypto operation: CKR_KEY_SIZE_RANGE`

Si vous transmettez une clé ne répondant pas aux exigences de l'algorithme, vous devez fournir une meilleure clé à l'aide de l'une des méthodes suivantes :

- Utiliser une phrase de passe. La structure fournit ensuite une clé qui remplit les conditions requises.

- Transmettre une taille de clé acceptée par l'algorithme. Par exemple, l'algorithme DES requiert une clé de 64 bits. L'algorithme 3DES requiert une clé de 192 bits.

Administration de la structure cryptographique

Cette section décrit la procédure d'administration des fournisseurs de logiciels et des fournisseurs de matériel dans la structure cryptographique. L'utilisation de ces fournisseurs peut être empêchée lorsque cela est souhaitable. Par exemple, vous pouvez désactiver l'implémentation d'un algorithme d'un seul fournisseur de logiciels. Ensuite, vous pouvez forcer le système à utiliser l'algorithme d'un autre fournisseur de logiciels.

Remarque - Un aspect important de l'administration de la structure cryptographique est la planification et l'implémentation de votre stratégie en matière de norme FIPS 140 : la norme de sécurité informatique du gouvernement américain destinée aux modules cryptographiques.

Si vous êtes tenu d'utiliser uniquement la cryptographie validée par la norme FIPS 140-2, vous devez exécuter Oracle Solaris 11.1 version SRU5.5 ou Oracle Solaris 11.1 version SRU3. Dans ces deux versions, Oracle a finalisé une validation de la norme FIPS 140-2 dans la structure cryptographique Solaris. Oracle Solaris 11.2 s'appuie sur cette validation et inclut des améliorations logicielles en matière de performances, de fonctions et de fiabilité. Dans la mesure du possible, vous devez configurer Solaris 11.2 en mode FIPS 140-2 pour profiter de ces améliorations.

Consultez la section " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 " et planifiez une stratégie FIPS globale pour vos systèmes.

La liste des tâches suivante présente les procédures permettant d'administrer les fournisseurs de logiciels et matériels dans la structure cryptographique.

TABLEAU 3-2 Liste des tâches Administration de la structure cryptographique

Tâche	Description	Voir
Planifiez la stratégie FIPS de vos systèmes.	Mettez au point votre plan d'activation des fournisseurs et consommateurs approuvés pour FIPS et mettez-le en oeuvre.	" Using a FIPS 140 Enabled System in Oracle Solaris 11.2 "
Etablissement de la liste des fournisseurs dans la structure cryptographique	Répertorie les algorithmes, les bibliothèques et les périphériques matériels disponibles pour l'utilisation dans la structure cryptographique	"Affichage de la liste des fournisseurs disponibles" à la page 35
Activez le mode FIPS 140.	Exécute la structure cryptographique conformément à la norme du gouvernement américain relatives aux modules cryptographiques.	"Création d'un environnement d'initialisation compatible FIPS 140" à la page 43
Ajout d'un fournisseur de logiciels.	Ajoute une bibliothèque PKCS #11 ou un module de noyau à la structure	"Ajout d'un fournisseur de logiciels" à la page 41

Tâche	Description	Voir
	cryptographique. Le fournisseur doit être signé.	
Interdiction d'utilisation d'un mécanisme au niveau de l'utilisateur.	Empêche l'utilisation d'un mécanisme logiciel. Le mécanisme peut être activé à nouveau.	“Interdiction de l'utilisation d'un mécanisme au niveau de l'utilisateur” à la page 45
Désactivation temporaire des mécanismes d'un module de noyau.	Empêche temporairement l'utilisation d'un mécanisme. Généralement utilisée à des fins de test.	“Interdiction de l'utilisation d'un mécanisme de logiciel noyau” à la page 46
Désinstallation d'une bibliothèque.	Empêche l'utilisation d'un fournisseur de logiciels utilisateur.	Exemple 3-17, “Suppression définitive d'une bibliothèque au niveau de l'utilisateur”
Désinstallation d'un fournisseur de noyau.	Empêche l'utilisation d'un fournisseur de logiciels noyau.	Exemple 3-19, “Suppression temporaire de la disponibilité d'un fournisseur de logiciels noyau”
Désactivation de mécanismes d'un fournisseur de matériel.	Permet de s'assurer que les mécanismes sélectionnés sur un accélérateur matériel ne sont pas utilisés.	“Désactivation des mécanismes et fonctions d'un fournisseur de matériel” à la page 49
Redémarrage ou actualisation des services cryptographiques.	Permet de s'assurer que les services cryptographiques sont disponibles.	“Actualisation ou redémarrage de tous les services cryptographiques” à la page 52

Affichage de la liste des fournisseurs disponibles

Les fournisseurs de matériel sont localisés et chargés automatiquement. Pour plus d'informations, reportez-vous à la page de manuel `driver.conf`(4).

Lorsque du matériel doit être connecté à la structure cryptographique, le matériel s'enregistre sur la SPI dans le noyau. La structure vérifie que le pilote matériel est signé. Plus précisément, la structure vérifie que le fichier d'objet du pilote est signé au moyen d'un certificat émis par Oracle.

Par exemple, la carte Sun Crypto Accelerator 6000 (`mca`), le pilote `ncp` pour l'accélérateur cryptographique sur les processeurs UltraSPARC T1 et T2 (`ncp`), le pilote n2cp pour les processeurs UltraSPARC T2 (`n2cp`) et le pilote `/dev/crypto` pour les systèmes de série T connectent les mécanismes matériels à la structure.

Pour plus d'informations sur l'obtention de votre fournisseur signé, reportez-vous aux informations relatives à la commande `elfsign` dans “Commandes au niveau de l'utilisateur dans la structure cryptographique” à la page 12.

Pour répertorier les fournisseurs disponibles, vous utilisez les commandes `cryptoadm list` avec différentes options, selon les informations spécifiques que vous souhaitez obtenir.

- Création d'une liste de tous les fournisseurs sur le système.

Le contenu et le format de la liste de fournisseurs varient selon les versions d'Oracle Solaris et les plates-formes. Exécutez la commande `cryptoadm list` sur votre système pour afficher les fournisseurs que votre système prend en charge. Seuls les mécanismes au niveau de l'utilisateur sont disponibles pour l'utilisation directe par les utilisateurs standard.

```
% cryptoadm list
User-level providers:       /* for applications */
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
Provider: /usr/lib/security/$ISA/pkcs11_tpm.so

Kernel software providers:       /* for IPsec, kssl, Kerberos */
des
aes
arcfour
blowfish
camellia
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0      /* for hardware */
ncp/0
n2cp/0
```

- Création de la liste des fournisseurs et de leurs mécanismes dans la structure cryptographique.

  Vous pouvez afficher la force et les modes, tels qu'`ECB` et `CBC`, des mécanismes disponibles. Cependant, certains de ces mécanismes peuvent ne pas être disponibles pour l'utilisation. Reportez-vous à l'élément suivant pour obtenir des instructions sur l'affichage de la liste des mécanismes pouvant être utilisés.

  La sortie suivante est tronquée à des fins d'affichage.

```
% cryptoadm list -m [provider=provider]
User-level providers:
=====================

Provider: /usr/lib/security/$ISA/pkcs11_kernel.so

Mechanisms:
CKM_DSA
CKM_RSA_X_509
CKM_RSA_PKCS
```

```
...
CKM_SHA256_HMAC_GENERAL
CKM_SSL3_MD5_MAC

Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
Mechanisms:
CKM_DES_CBC
CKM_DES_CBC_PAD
CKM_DES_ECB
CKM_DES_KEY_GEN
CKM_DES_MAC_GENERAL
...
CKM_ECDSA_SHA1
CKM_ECDH1_DERIVE

Provider: /usr/lib/security/$ISA/pkcs11_tpm.so
/usr/lib/security/$ISA/pkcs11_tpm.so: no slots presented.

Kernel providers:
=========================
des: CKM_DES_ECB,CKM_DES_CBC,CKM_DES3_ECB,CKM_DES3_CBC
aes: CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM, \
     CKM_AES_GCM,CKM_AES_GMAC,
CKM_AES_CFB128,CKM_AES_XTS,CKM_AES_XCBC_MAC
arcfour: CKM_RC4
blowfish: CKM_BLOWFISH_ECB,CKM_BLOWFISH_CBC
ecc: CKM_EC_KEY_PAIR_GEN,CKM_ECDH1_DERIVE,CKM_ECDSA, \
     CKM_ECDSA_SHA1
sha1: CKM_SHA_1,CKM_SHA_1_HMAC,CKM_SHA_1_HMAC_GENERAL
sha2: CKM_SHA224,CKM_SHA224_HMAC,...CKM_SHA512_256_HMAC_GENERAL

md4: CKM_MD4
md5: CKM_MD5,CKM_MD5_HMAC,CKM_MD5_HMAC_GENERAL
rsa: CKM_RSA_PKCS,CKM_RSA_X_509,CKM_MD5_RSA_PKCS, \
     CKM_SHA1_RSA_PKCS,CKM_SHA224_RSA_PKCS,
CKM_SHA256_RSA_PKCS,CKM_SHA384_RSA_PKCS,CKM_SHA512_RSA_PKCS
swrand: No mechanisms presented.
n2rng/0: No mechanisms presented.
ncp/0: CKM_DSA,CKM_RSA_X_509,CKM_RSA_PKCS,CKM_RSA_PKCS_KEY_PAIR_GEN,
CKM_DH_PKCS_KEY_PAIR_GEN,CKM_DH_PKCS_DERIVE,CKM_EC_KEY_PAIR_GEN,
CKM_ECDH1_DERIVE,CKM_ECDSA
n2cp/0: CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES3_CBC, \
     ...CKM_SSL3_SHA1_MAC
```

- Création de la liste des mécanismes cryptographiques disponibles.

  La stratégie détermine quels mécanismes sont disponibles pour l'utilisation. L'administrateur définit la stratégie. Un administrateur peut choisir de désactiver des mécanismes à partir

d'un fournisseur particulier. L'option -p affiche la liste des mécanismes autorisés par la stratégie définie par l'administrateur.

```
% cryptoadm list -p [provider=provider]
User-level providers:
=====================
/usr/lib/security/$ISA/pkcs11_kernel.so: \
     all mechanisms are enabled.random is enabled.
/usr/lib/security/$ISA/pkcs11_softtoken.so: \
     all mechanisms are enabled, random is enabled.
/usr/lib/security/$ISA/pkcs11_tpm.so: all mechanisms are enabled.

Kernel providers:
=========================
des: all mechanisms are enabled.
aes: all mechanisms are enabled.
arcfour: all mechanisms are enabled.
blowfish: all mechanisms are enabled.
ecc: all mechanisms are enabled.
sha1: all mechanisms are enabled.
sha2: all mechanisms are enabled.
md4: all mechanisms are enabled.
md5: all mechanisms are enabled.
rsa: all mechanisms are enabled.
swrand: random is enabled.
n2rng/0: all mechanisms are enabled. random is enabled.
ncp/0: all mechanisms are enabled.
n2cp/0: all mechanisms are enabled.
```

Les exemples suivants montrent des utilisations spécifiques supplémentaires de la commande `cryptoadm list`.

EXEMPLE 3-11 Affichage de la liste des informations cryptographiques d'un fournisseur spécifique

La spécification du fournisseur dans la commande `cryptoadm` *options* limite la sortie uniquement aux informations qui s'appliquent au fournisseur.

```
# cryptoadm enable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5, CKM_MD5_HMAC,...
random is enabled.
```

Dans la sortie suivante, seuls les mécanismes sont activés. Le générateur aléatoire reste désactivé.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
```

```
# cryptoadm enable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

La sortie suivante montre que toutes les fonctions et tous les mécanismes de la carte sont activés.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms ar enabled, except CKM_DES_ECB,CKM_DES3_ECB.
random is disabled.

# cryptoadm enable provider=dca/0 all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
```

EXEMPLE 3-12 Mécanismes cryptographique au niveau de la recherche de l'utilisateur uniquement

Dans l'exemple suivant, tous les mécanismes proposés par la bibliothèque au niveau de l'utilisateur, `pkcs11_softtoken`, sont répertoriés.

```
% cryptoadm list -m provider=/usr/lib/security/\
   $ISA/pkcs11_softtoken.so
Mechanisms:
CKM_DES_CBC
CKM_DES_CBC_PAD
CKM_DES_ECB
CKM_DES_KEY_GEN
CKM_DES_MAC_GENERAL
CKM_DES_MAC
…
CKM_ECDSA
CKM_ECDSA_SHA1
CKM_ECDH1_DERIVE
```

EXEMPLE 3-13 Détermination des fonctions exécutées par les différents mécanismes cryptographiques

Les mécanismes exécutent des fonctions cryptographiques spécifiques, telles que la signature ou la génération de clé. Les options `-v -m` affichent tous les mécanismes et leurs fonctions.

Dans cet exemple, l'administrateur souhaite déterminer les fonctions pour lesquelles le mécanisme `CKM_ECDSA*` peut être utilisé.

```
% cryptoadm list -vm
User-level providers:
=====================
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
Number of slots: 3
Slot #2
Description: ncp/0 Crypto Accel Asym 1.0
...
CKM_ECDSA                   163  571  X  .  .  .  X  .  X  .  .  .  .  .  .  .
```

```
...

Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
...
CKM_ECDSA       112 571  .  .  .  .  X  .  X  .  .  .  .  .  .  .  .
CKM_ECDSA_SHA1  112 571  .  .  .  .  X  .  X  .  .  .  .  .  .  .  .
...
Kernel providers:
=================
...
ecc: CKM_EC_KEY_PAIR_GEN,CKM_ECDH1_DERIVE,CKM_ECDSA,CKM_ECDSA_SHA1
...
```

La liste indique que ces mécanismes sont disponibles à partir des fournisseurs au niveau de l'utilisateur suivants :

- `CKM_ECDSA` et `CKM_ECDSA_SHA1` : en tant que mise en oeuvre logicielle dans la bibliothèque `/usr/lib/security/$ISA/pkcs11_softtoken.so`
- `CKM_ECDSA` : accéléré par `ncp/0 Crypto Accel Asym 1.0` dans la bibliothèque `/usr/lib/security/$ISA/pkcs11_kernel.so`

Chaque élément dans une entrée représente un élément d'information sur le mécanisme. Pour ces mécanismes ECC, la liste indique les éléments suivants :

- Longueur minimale : 112 octets
- Longueur maximale : 571 octets
- Matériel : est ou n'est pas disponible sur le matériel
- Chiffrer : n'est pas utilisé pour chiffrer les données.
- Déchiffrer : n'est pas utilisé pour déchiffrer les données.
- Résumé : n'est pas utilisé pour créer des résumés de message.
- Signer : est utilisé pour signer les données.
- Signer + récupérer : n'est pas utilisé pour signer les données, lorsque les données peuvent être récupérées à partir de la signature.
- Vérifier : est utilisé pour vérifier les données signées.
- Vérifier + récupérer : n'est pas utilisé pour vérifier les données qui peuvent être récupérées à partir de la signature.
- Génération de clé : n'est pas utilisé pour générer une clé privée.
- Génération de paire : n'est pas utilisé pour générer une paire de clés.
- Encapsuler : n'est pas utilisé pour encapsuler c'est-à-dire, chiffrer, une clé existante.
- Désencapsuler : n'est pas utilisé pour désencapsuler une clé encapsulée.
- Dériver : n'est pas utilisé pour dériver une nouvelle clé à partir d'une clé de base.
- Capacités EC : capacités EC (courbe elliptique) absentes qui ne sont pas couvertes par les éléments précédents

Ajout d'un fournisseur de logiciels

La procédure suivante explique comment ajouter des fournisseurs au système. Vous devez vous connecter en tant qu'administrateur disposant du profil de droits Crypto Management (gestion de la cryptographie). Pour plus d'informations, reportez-vous à la section “ A l’aide de vos droits administratifs attribués ” du manuel “ Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ”.

▼ Ajout d'un fournisseur de logiciels

1. **Répertoriez les fournisseurs de logiciels disponibles sur le système.**

```
% cryptoadm list
User-level providers:
Provider: /usr/lib/security/$ISA/pkcs11_kernel.so
Provider: /usr/lib/security/$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_tpm.so: all mechanisms are enabled.

Kernel software providers:
des
aes
arcfour
blowfish
camellia
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

2. **Ajoutez le fournisseur à partir d'un référentiel.**
   Le logiciel du fournisseur existant a reçu un certificat émis par Oracle.

3. **Actualisez les fournisseurs.**
   Vous devez actualiser les fournisseurs si vous avez ajouté un fournisseur de logiciels ou si vous avez ajouté un matériel et spécifié une stratégie pour ce matériel.

```
# svcadm refresh svc:/system/cryptosvc
```

4. **Localisez le nouveau fournisseur dans la liste.**
   Dans ce cas, un nouveau fournisseur de logiciels noyau a été installé.

```
# cryptoadm list
…
```

```
Kernel software providers:
des
aes
arcfour
blowfish
camellia
ecc
sha1
sha2
md4
md5
rsa
swrand
sha3       <-- added provider
…
```

Exemple 3-14 Ajout d'un fournisseur de logiciels au niveau de l'utilisateur

Dans l'exemple suivant, une bibliothèque PKCS 11 signée est installée.

```
# pkgadd -d /cdrom/cdrom0/PKCSNew
     Answer the prompts
# svcadm refresh system/cryptosvc
# cryptoadm list
user-level providers:
=========================
/usr/lib/security/$ISA/pkcs11_kernel.so
/usr/lib/security/$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_tpm.so
/opt/lib/$ISA/libpkcs11.so.1      <-- added provider
```

Les développeurs qui testent une bibliothèque avec la structure cryptographique peuvent installer la bibliothèque manuellement.

```
# cryptoadm install provider=/opt/lib/\$ISA/libpkcs11.so.1
```

Création d'un environnement d'initialisation compatible FIPS 140

Par défaut, le mode FIPS 140 est désactivé dans Oracle Solaris. Au cours de cette procédure, vous créez un environnement d'initialisation (BE) pour le mode FIPS-140, puis activez FIPS 140 et initialisez dans le nouvel environnement. En vous offrant un environnement d'initialisation de sauvegarde, cette méthode vous permet d'effectuer rapidement une récupération après une panique du système pouvant résulter de tests de compatibilité FIPS 140.

Pour une présentation du mode FIPS, reportez-vous à la section " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ". Reportez-vous également à la page de manuel `cryptoadm`(1M) et à la rubrique "Structure cryptographique et FIPS 140" à la page 13.

▼ Création d'un environnement d'initialisation compatible FIPS 140

Avant de commencer

Vous devez prendre le rôle `root`. Pour plus d'informations, reportez-vous à la section " A l'aide de vos droits administratifs attribués " du manuel " Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ".

1. Déterminez si le système est en mode FIPS 140.

```
% cryptoadm list fips-140
User-level providers:
=====================
/usr/lib/security/$ISA/pkcs11_softtoken: FIPS-140 mode is disabled.

Kernel software providers:
==========================
des: FIPS-140 mode is disabled.
aes: FIPS-140 mode is disabled.
ecc: FIPS-140 mode is disabled.
sha1: FIPS-140 mode is disabled.
sha2: FIPS-140 mode is disabled.
rsa: FIPS-140 mode is disabled.
swrand: FIPS-140 mode is disabled.

Kernel hardware providers:
==========================:
```

2. Créez un environnement d'initialisation pour votre version FIPS 140 de la structure cryptographique.

Avant d'activer le mode FIPS 140, vous devez créer, activer et initialiser un nouvel environnement d'initialisation à l'aide de la commande `beadm`. Un système compatible FIPS-140 exécute des tests de conformité qui peuvent entraîner une panique en cas d'échec. Par conséquent, il est important de disposer d'un environnement d'initialisation que vous pouvez initialiser pour rétablir votre système pendant que vous résolvez les problèmes relatifs à la limite FIPS 140.

a. Créez un environnement d'initialisation en fonction de votre environnement d'initialisation actuel.

Dans cet exemple, vous créez un environnement d'initialisation nommé `S11.1-FIPS`.

```
# beadm create S11.1-FIPS-140
```

b. Activez cet environnement d'initialisation.

```
# beadm activate S11.1-FIPS-140
```

c. Réinitialisez le système.

d. Activez le mode FIPS 140 dans le nouvel environnement d'initialisation.

```
# cryptoadm enable fips-140
```

Remarque - Cette sous-commande ne désactive pas les algorithmes non approuvés par FIPS 140 dans la bibliothèque `pkcs11_softtoken` de niveau utilisateur et les fournisseurs de logiciels noyau. Les utilisateurs de la structure doivent utiliser uniquement des algorithmes compatibles FIPS 140.

Pour plus d'informations sur les effets du mode FIPS 140, reportez-vous à la section " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ". Voir également la page de manuel `cryptoadm`(1M).

---

3. Si vous ne souhaitez pas que FIPS 140 soit activé, désactivez le mode FIPS 140.

Vous pouvez effectuer une réinitialisation dans l'environnement d'initialisation d'origine ou désactiver FIPS 140 dans l'environnement d'initialisation actuel.

- **Initialisez l'environnement d'initialisation d'origine.**

```
# beadm list
BE               Active Mountpoint Space   Policy Created
--               ------ ---------- -----   ------ -------
S11.1            -      -          48.22G   static 2012-10-10 10:10
S11.1-FIPS-140   NR     /          287.01M  static 2012-11-18 18:18
# beadm activate S11.1
# beadm list
BE               Active Mountpoint Space   Policy Created
--               ------ ---------- -----   ------ -------
S11.1            R      -          48.22G   static 2012-10-10 10:10
S11.1-FIPS-140   N      /          287.01M  static 2012-11-18 18:18
# reboot
```

- **Désactivez le mode FIPS 140 dans l'environnement d'initialisation actuel et effectuez une réinitialisation.**

```
# cryptoadm disable fips-140
```

Le mode FIPS 140 reste actif tant que le système n'a pas été réinitialisé.

```
# reboot
```

Prévention de l'utilisation des mécanismes

Si certains des mécanismes cryptographiques provenant d'un fournisseur de bibliothèques ne doivent pas être utilisés, vous pouvez supprimer les mécanismes sélectionnés. Vous pouvez

envisager d'empêcher l'utilisation de mécanismes si, par exemple, le même mécanisme dans une autre bibliothèque fonctionne mieux, ou si une vulnérabilité de sécurité est en cours d'étude.

Si la structure cryptographique fournit plusieurs modes d'un fournisseur tel qu'AES, vous pouvez supprimer un mécanisme lent ou endommagé. Vous pouvez également avoir recours à cette procédure pour supprimer un algorithme avec des failles de sécurité avérées.

Vous pouvez désactiver de façon sélective des mécanismes et la fonction de nombres aléatoires à partir d'un fournisseur de matériel. Pour les réactiver, reportez-vous à l'Exemple 3-22, “Activation des mécanismes et fonctions sur un fournisseur de matériel”. Le matériel de cet exemple, la carte Sun Crypto Accelerator 1000, fournit un générateur de nombres aléatoires.

▼ Interdiction de l'utilisation d'un mécanisme au niveau de l'utilisateur

Avant de commencer

Vous devez vous connecter en tant qu'administrateur disposant du profil de droits Crypto Management (gestion de la cryptographie). Pour plus d'informations, reportez-vous à la section “ A l’aide de vos droits administratifs attribués ” du manuel “ Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ”.

1. **Répertoriez les mécanismes offerts par un fournisseur de logiciels particulier au niveau de l'utilisateur.**

```
% cryptoadm list -m provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
CKM_AES_CBC,CKM_AES_CBC_PAD,CKM_AES_ECB,CKM_AES_KEY_GEN,
…
```

2. **Répertoriez les mécanismes disponibles pour l'utilisation.**

```
$ cryptoadm list -p
user-level providers:
=====================
…
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.
…
```

3. **Désactivez les mécanismes qui ne doivent pas être utilisés.**

```
$ cryptoadm disable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so \
> mechanism=CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB
```

4. **Répertoriez les mécanismes disponibles pour l'utilisation.**

```
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
```

```
except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
```

Exemple 3-15 Activation d'un mécanisme d'un fournisseur de logiciels au niveau de l'utilisateur

Dans l'exemple suivant, un mécanisme DES désactivé est de nouveau rendu disponible pour l'utilisation.

```
$ cryptoadm list -m provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
…
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
$ cryptoadm enable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so \
> mechanism=CKM_DES_ECB
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
```

Exemple 3-16 Activation de tous les mécanismes d'un fournisseur de logiciels au niveau de l'utilisateur

Dans l'exemple suivant, tous les mécanismes de la bibliothèque au niveau de l'utilisateur sont activés.

```
$ cryptoadm enable provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so all
$ cryptoadm list -p provider=/usr/lib/security/\$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.
```

Exemple 3-17 Suppression définitive d'une bibliothèque au niveau de l'utilisateur

Dans l'exemple suivant, une bibliothèque `libpkcs11.so.1` est supprimée du répertoire `/opt`

```
$ cryptoadm uninstall provider=/opt/lib/\$ISA/libpkcs11.so.1
$ cryptoadm list
user-level providers:
/usr/lib/security/$ISA/pkcs11_kernel.so
/usr/lib/security/$ISA/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_tpm.so

kernel providers:
…
```

▼ Interdiction de l'utilisation d'un mécanisme de logiciel noyau

Avant de commencer

Vous devez vous connecter en tant qu'administrateur disposant du profil de droits Crypto Management (gestion de la cryptographie). Pour plus d'informations, reportez-vous à la section

" A l'aide de vos droits administratifs attribués " du manuel " Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ".

1. **Répertoriez les mécanismes offerts par un fournisseur de logiciels noyau particulier.**

```
$ cryptoadm list -m provider=aes
aes: CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM,CKM_AES_GCM,
CKM_AES_GMAC,CKM_AES_CFB128,CKM_AES_XTS,CKM_AES_XCBC_MAC
```

2. **Répertoriez les mécanismes disponibles pour l'utilisation.**

```
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled.
```

3. **Désactivez le mécanisme qui ne doit pas être utilisé.**

```
$ cryptoadm disable provider=aes mechanism=CKM_AES_ECB
```

4. **Répertoriez les mécanismes disponibles pour l'utilisation.**

```
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled, except CKM_AES_ECB.
```

Exemple 3-18 Activation d'un mécanisme d'un fournisseur de logiciels noyau

Dans l'exemple suivant, un mécanisme AES désactivé est de nouveau rendu disponible pour l'utilisation.

```
cryptoadm list -m provider=aes
aes: CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM,
CKM_AES_GCM,CKM_AES_GMAC,CKM_AES_CFB128,CKM_AES_XTS,CKM_AES_XCBC_MAC
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled, except CKM_AES_ECB.
$ cryptoadm enable provider=aes mechanism=CKM_AES_ECB
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled.
```

Exemple 3-19 Suppression temporaire de la disponibilité d'un fournisseur de logiciels noyau

Dans l'exemple suivant, l'utilisation du fournisseur AES est temporairement rendue impossible. La sous-commande `unload` est utile pour empêcher le chargement automatique d'un fournisseur pendant que le fournisseur est en cours de désinstallation. Par exemple, la sous-commande `unload` peut être utilisée pour modifier un mécanisme de ce fournisseur.

```
$ cryptoadm unload provider=aes

$ cryptoadm list
…
Kernel software providers:
```

```
des
aes (inactive)
arcfour
blowfish
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

Le fournisseur AES reste indisponible jusqu'à l'actualisation de la structure cryptographique.

```
$ svcadm refresh system/cryptosvc

$ cryptoadm list
…
Kernel software providers:
des
aes
arcfour
blowfish
camellia
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

Si un consommateur de noyau utilise le fournisseur de logiciels noyau, le logiciel n'est pas déchargé. Un message d'erreur s'affiche et le fournisseur continue d'être disponible pour l'utilisation.

Exemple 3-20 Suppression définitive de la disponibilité du fournisseur de logiciels

Dans l'exemple suivant, l'utilisation du fournisseur AES est définitivement rendue impossible. Une fois supprimé, le fournisseur AES n'apparaît plus dans la liste des stratégies des fournisseurs de logiciels noyau.

```
$ cryptoadm uninstall provider=aes

$ cryptoadm list
…
Kernel software providers:
```

```
des
arcfour
blowfish
camellia
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

Si un consommateur de noyau utilise ce fournisseur de logiciels noyau, un message d'erreur s'affiche et le fournisseur continue d'être disponible pour l'utilisation.

Exemple 3-21 Réinstallation d'un fournisseur de logiciels noyau supprimé

Dans l'exemple suivant, le fournisseur de logiciels noyau AES est réinstallé. Pour réinstaller un fournisseur de noyau supprimé, vous devez énumérer les mécanismes à installer.

```
$ cryptoadm install provider=aes \
mechanism=CKM_AES_ECB,CKM_AES_CBC,CKM_AES_CTR,CKM_AES_CCM,
CKM_AES_GCM,CKM_AES_GMAC,CKM_AES_CFB128,CKM_AES_XTS,CKM_AES_XCBC_MAC

$ cryptoadm list
…
Kernel software providers:
des
aes
arcfour
blowfish
camellia
ecc
sha1
sha2
md4
md5
rsa
swrand
n2rng/0
ncp/0
n2cp/0
```

▼ Désactivation des mécanismes et fonctions d'un fournisseur de matériel

Avant de commencer

Vous devez vous connecter en tant qu'administrateur disposant du profil de droits Crypto Management (gestion de la cryptographie). Pour plus d'informations, reportez-vous à la section

" A l'aide de vos droits administratifs attribués " du manuel " Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ".

- **Choisissez les mécanismes ou la fonction à désactiver.**

  Répertoriez le fournisseur de matériel.

  ```
  # cryptoadm list
  ...
  Kernel hardware providers:
  dca/0
  ```

  - **Désactivez les mécanismes sélectionnés.**

    ```
    # cryptoadm list -m provider=dca/0
    dca/0: CKM_RSA_PKCS, CKM_RSA_X_509, CKM_DSA, CKM_DES_CBC, CKM_DES3_CBC
    random is enabled.
    # cryptoadm disable provider=dca/0 mechanism=CKM_DES_CBC,CKM_DES3_CBC
    # cryptoadm list -p provider=dca/0
    dca/0: all mechanisms are enabled except CKM_DES_CBC,CKM_DES3_CBC.
    random is enabled.
    ```

  - **Désactivez le générateur de nombres aléatoires.**

    ```
    # cryptoadm list -p provider=dca/0
    dca/0: all mechanisms are enabled. random is enabled.
    # cryptoadm disable provider=dca/0 random
    # cryptoadm list -p provider=dca/0
    dca/0: all mechanisms are enabled. random is disabled.
    ```

  - **Désactivez tous les mécanismes. Ne désactivez pas le générateur de nombres aléatoires.**

    ```
    # cryptoadm list -p provider=dca/0
    dca/0: all mechanisms are enabled. random is enabled.
    # cryptoadm disable provider=dca/0 mechanism=all
    # cryptoadm list -p provider=dca/0
    dca/0: all mechanisms are disabled. random is enabled.
    ```

  - **Désactivez toutes les fonctions et tous les mécanismes sur le matériel.**

    ```
    # cryptoadm list -p provider=dca/0
    dca/0: all mechanisms are enabled. random is enabled.
    # cryptoadm disable provider=dca/0 all
    # cryptoadm list -p provider=dca/0
    dca/0: all mechanisms are disabled. random is disabled.
    ```

Exemple 3-22 Activation des mécanismes et fonctions sur un fournisseur de matériel

Dans les exemples suivants, les mécanismes désactivés sur un élément matériel sont activés individuellement.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_ECB,CKM_DES3_ECB

.
random is enabled.
# cryptoadm enable provider=dca/0 mechanism=CKM_DES3_ECB
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_ECB.
random is enabled.
```

Dans l'exemple ci-dessous, seul le générateur aléatoire est activé.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,….
random is disabled.
# cryptoadm enable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,….
random is enabled.
```

Dans l'exemple ci-dessous, seuls les mécanismes sont activés. Le générateur aléatoire reste désactivé.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,….
random is disabled.
# cryptoadm enable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

Dans l'exemple suivant, toutes les fonctions et tous les mécanismes de la carte sont activés.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_DES_ECB,CKM_DES3_ECB.
random is disabled.
# cryptoadm enable provider=dca/0 all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
```

Actualisation ou redémarrage de tous les services cryptographiques

Par défaut, la structure cryptographique est activée. Lorsque le démon `kcfd` échoue pour une raison quelconque, l'utilitaire de gestion des services peut être utilisé pour redémarrer les services cryptographiques. Pour plus d'informations, reportez-vous aux pages de manuel `smf`(5) et `svcadm`(1M). Pour connaître l'effet du redémarrage des services cryptographiques sur les zones, reportez-vous à la section “Services cryptographiques et zones” à la page 13.

▼ Actualisation ou redémarrage de tous les services cryptographiques

Avant de commencer Vous devez vous connecter en tant qu'administrateur disposant du profil de droits Crypto Management (gestion de la cryptographie). Pour plus d'informations, reportez-vous à la section " A l'aide de vos droits administratifs attribués " du manuel " Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ".

1. Vérifiez l'état des services cryptographiques.

```
% svcs cryptosvc
STATE          STIME    FMRI
offline         Dec_09   svc:/system/cryptosvc:default
```

2. Activez les services cryptographiques.

```
# svcadm enable svc:/system/cryptosvc
```

Exemple 3-23 Actualisation des services cryptographiques

Dans l'exemple suivant, les services cryptographiques sont actualisés dans la zone globale. Par conséquent, la stratégie de cryptographie au niveau du noyau dans chaque zone non globale est également actualisée.

```
# svcadm refresh system/cryptosvc
```

♦ ♦ ♦ CHAPITRE 4

Structure de gestion des clés

La fonction KMF (Key Management Framework, structure de gestion clé) fournit des outils et interfaces de programmation pour gérer les objets de clé publique. Les objets de clé publique comprennent les certificats X.509 et les paires de clés publiques et privées. Les formats de stockage de ces objets peuvent varier. KMF offre également un outil de gestion des stratégies qui définissent l'utilisation de certificats X.509 par des applications. KMF prend en charge des plug-ins de fournisseurs tiers.

Ce chapitre comprend les sections suivantes :

Gestion des technologies à clé publique

KMF centralise la gestion des technologies à clé publique (PKI). Oracle Solaris possède plusieurs applications qui utilisent les technologies PKI. Chaque application fournit ses propres interfaces de programmation, mécanismes de stockage des clés et utilitaires d'administration. Si une application offre un mécanisme d'application de stratégie, ce mécanisme s'applique uniquement à l'application correspondante. Avec KMF, les applications utilisent un ensemble unifié d'outils d'administration, un ensemble d'interfaces de programmation et un mécanisme d'application de stratégie. Ces fonctions gèrent les besoins en PKI de toutes les applications adoptant ces interfaces.

KMF unifie la gestion des technologies à clé publique avec les interfaces suivantes :

- Commande `pktool` : gère les objets PKI, tels que les certificats, dans divers keystores.
- Commande `kmfcfg` : gère la base de données de stratégie PKI et les plug-ins tiers.

  Les décisions de stratégie PKI comprennent des opérations telles que la méthode de validation d'une opération. En outre, la stratégie PKI peut limiter l'étendue d'un certificat. Par exemple, la stratégie PKI peut affirmer qu'un certificat peut être utilisé uniquement à des fins spécifiques. Une telle stratégie peut empêcher qu'un certificat soit utilisé pour d'autres demandes.

- Bibliothèque KMF : contient des interfaces de programmation qui abstraient le mécanisme keystore sous-jacent.

  Les applications n'ont pas à choisir un mécanisme de keystore spécifique, ils peuvent migrer d'un seul mécanisme à un autre. Les fichiers keystore pris en charge sont PKCS #11, NSS et OpenSSL La bibliothèque comprend une structure modulable permettant l'ajout de nouveaux mécanismes keystore. Par conséquent, les applications qui utilisent les nouveaux mécanismes ne nécessitent que des modifications mineures pour utiliser un nouveau keystore.

Utilitaires de la structure de gestion des clés

KMF offre des méthodes pour la gestion du stockage des clés et fournit la stratégie globale concernant l'utilisation de ces clés. KMF gère la stratégie, les clés et les certificats pour les trois technologies à clé publique suivantes :

- Fournisseurs de jetons de PKCS #11, c'est-à-dire provenant de la structure cryptographique
- NSS, c'est-à-dire les services de sécurité réseau (Network Security Services)
- OpenSSL, un keystore basé les fichiers

L'outil `kmfcfg` peut créer, modifier ou supprimer des entrées de stratégie KMF. L'outil gère également les plug-ins de la structure. KMF gère les keystores par le biais de la commande `pktool`. Pour plus d'informations, reportez-vous aux pages de manuel `kmfcfg`(1) et `pktool`(1) et aux sections suivantes.

Gestion de la stratégie KMF

La stratégie KMF est stockée dans une base de données. Cette base de données de stratégie est accédée en interne par toutes les applications qui utilisent les interfaces de programmation KMF. La base de données peut limiter l'utilisation des clés et des certificats qui sont gérés par la bibliothèque KMF. Lorsqu'une application tente de vérifier un certificat, l'application vérifie la base de données de stratégies. La commande `kmfcfg` modifie la base de données de stratégies.

Gestion de plug-in KMF

La commande `kmfcfg` fournit les sous-commandes suivantes pour les plug-ins :

- `list plugin` : répertorie les plug-ins gérés par KMF.
- `install` *plugin* : installe le plug-in par le nom de chemin d'accès du module et crée un keystore pour le plug-in. Pour supprimer le plug-in de KMF, supprimez le keystore.

- `uninstall` *plugin* : supprime le plug-in de KMF en supprimant son keystore.
- `modify` *plugin* : active le plug-in à exécuter avec une option définie dans le code du plug-in, comme `debug`.

Pour plus d'informations, reportez-vous à la page de manuel `kmfcfg`(1). Pour connaître la procédure, reportez-vous à la section “Gestion des plug-ins tiers dans KMF” à la page 68.

Gestion de keystore KMF

KMF gères les keystores pour trois technologies à clé publique, jetons PKCS #11, NSS et OpenSSL. Pour l'ensemble de ces technologies, la commande `pktool` vous permet d'effectuer les opérations suivantes :

- Générez un certificat autosigné
- Génération d'une demande de certificat
- Génération d'une clé symétrique
- Génération d'une paire de clés publique/privée
- Génération d'une CSR (certificate signing request, demande de signature de certificat) PKCS #10 à envoyer à une autorité de certification externe (CA) pour signature
- Signature d'une CSR PKCS #10
- Importation d'objets dans le keystore
- Liste des objets dans le keystore
- Suppression d'objets du keystore
- Téléchargement d'une CRL

Pour les technologies PKCS #11 et NSS, la commande `pktool` vous permet également de définir un code confidentiel en générant une phrase de passe pour le keystore ou pour un objet dans le keystore.

Pour consulter des exemples d'utilisation de l'utilitaire `pktool`, reportez-vous à la page de manuel `pktool`(1) et au Tableau 4-1, “Liste des tâches Utilisation de la structure de gestion des clés”.

Utilisation de la structure de gestion des clés

Cette section décrit l'utilisation de la commande `pktool` pour gérer vos objets de clé publique, tels que des mots et phrases de passe, des fichiers, des keystores, des certificats et des CRL.

La structure de gestion des clés (KMF) vous permet de gérer de manière centralisée les technologies à clé publique.

TABLEAU 4-1 Liste des tâches Utilisation de la structure de gestion des clés

Tâche	Description	Voir
Création d'un certificat	Crée un certificat à utiliser par PKCS#11, NSS ou OpenSSL.	"Création d'un certificat à l'aide de la commande `pktool gencert`" à la page 56
Exportation d'un certificat	Crée un fichier avec le certificat et ses clés de prise en charge. Le fichier peut être protégé par un mot de passe.	"Exportation d'un certificat et de la clé privée au format PKCS #12" à la page 60
Importation d'un certificat	Importe un certificat à partir d'un autre système.	"Importation d'un certificat dans votre keystore" à la page 58
	Importe un certificat en format PKCS #12 à partir d'un autre système.	Exemple 4-2, "Importation d'un fichier PKCS #12 dans votre keystore"
Génération d'une phrase de passe	Génère une phrase de passe pour l'accès à un keystore PKCS #11 ou NSS.	"Génération d'une phrase de passe à l'aide de la commande `pktool setpin`" à la page 61
Génération d'une clé symétrique	Génère des clés symétriques à utiliser pour chiffrer les fichiers, pour créer le code MAC d'un fichier et pour les applications.	"Génération d'une clé symétrique à l'aide de la commande `pktool`" à la page 22
Génération d'une paire de clés	Génération d'une paire de clés publique/privée à utiliser avec des applications.	"Génération d'une paire de clés à l'aide de la commande `pktool genkeypair`" à la page 63
Génération d'une CSR PKCS #10	Génération d'une CSR (certificate signing request, demande de signature de certificat) PKCS #10 à faire signer par une autorité de certification externe (CA).	Page de manuel `pktool`(1)
Signature d'une CSR PKCS #10	Signe une CSR PKCS #10.	"Signature d'une demande de certificat à l'aide de la commande `pktool signcsr`" à la page 67
Ajout d'un plug-in à KMF	Installe, modifie et répertorie un plug-in. En outre, supprime le plug-in dans la KMF.	"Gestion des plug-ins tiers dans KMF" à la page 68

▼ Création d'un certificat à l'aide de la commande `pktool gencert`

Cette procédure crée un certificat autosigné et le stocke dans le keystore PKCS #11. Dans le cadre de cette opération, une paire de clés publique et privée RSA est également créée. La clé privée est stockée dans le keystore avec le certificat.

1. Générez un certificat autosigné

```
% pktool gencert [keystore=keystore] label=label-name \
subject=subject-DN serial=hex-serial-number keytype=rsa/dsa keylen=key-size
```

`keystore=`*keystore* Spécifie le keystore par type d'objet de clé publique. La valeur peut être `nss`, `pkcs11` ou `file`. Ce mot de passe est facultatif.

`label=`*label-name*	Spécifie un nom unique donné au certificat par l'émetteur.
`subject=`*subject-DN*	Spécifie le nom distinctif du certificat.
`serial=`*hex-serial-number*	Spécifie le numéro de série au format hexadécimal. L'émetteur du certificat choisit ce nombre, comme par exemple `0x0102030405`.
`keytype=`*key type*	Variable facultative qui spécifie le type de clé privée associée au certificat. Pour connaître les types de clé disponibles pour le keystore sélectionné, reportez-vous à la page de manuel `pktool`(1). Pour utiliser une clé approuvée par la norme FIPS 140, consultez les types de clé approuvés dans la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".
`keylen=`*key size*	Variable facultative qui spécifie la longueur de la clé privée associée au certificat. Pour utiliser une clé approuvée par la norme FIPS 140, consultez les longueurs de clé approuvées pour le type de clé sélectionné dans la section " FIPS 140 Algorithms in the Cryptographic Framework " du manuel " Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ".

2. **Vérifiez le contenu du keystore.**

```
% pktool list
Found number certificates.
1. (X.509 certificate)
Label:  label-name
ID: fingerprint that binds certificate to private key
Subject: subject-DN
Issuer:  distinguished-name
Serial:  hex-serial-number
n. ...
```

Cette commande répertorie tous les certificats dans le keystore. Dans l'exemple suivant, le keystore ne contient qu'un seul certificat.

Exemple 4-1 Création d'un certificat autosigné à l'aide de `pktool`

Dans l'exemple suivant, un utilisateur à `My Company` crée un certificat autosigné et le stocke dans un keystore pour les objets PKCS #11. Le keystore est initialement vide. Si le keystore n'a pas été initialisé, le numéro d'identification personnel du softtoken est `changeme`. Pour réinitialiser ce numéro, vous pouvez utiliser la commande `pktool setpin`. Notez qu'un type et une longueur de clé approuvés par FIPS, RSA 2048, est spécifié dans les options de commande.

```
% pktool gencert keystore=pkcs11 label="My Cert" \
subject="C=US, O=My Company, OU=Security Engineering Group, CN=MyCA" \
serial=0x000000001 keytype=rsa keylen=2048
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token

% pktool list
No.  Key Type  Key Len.  Key Label
-----------------------------------------------------
Asymmetric public keys:
1    RSA                My Cert
Certificates:
1    X.509 certificate
Label: My Cert
ID: d2:7e:20:04:a5:66:e6:31:90:d8:53:28:bc:ef:55:55:dc:a3:69:93
Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
...
...
Serial: 0x00000010
...
```

▼ Importation d'un certificat dans votre keystore

Cette procédure explique comment importer un fichier contenant des informations PKI codées avec PEM ou DER raw dans votre keystore. Pour connaître la procédure d'exportation, reportez-vous à l'Exemple 4-4, “Exportation d'un certificat et de la clé privée au format PKCS #12”.

1. **Importez le certificat.**

   ```
   % pktool import keystore=keystore infile=infile-name label=label-name
   ```

2. **Si vous importez des objets PKI privés, entrez les mots de passe lorsque vous y êtes invité.**

   a. **A l'invite, saisissez le mot de passe pour le fichier.**

      Si vous importez des informations PKI privées, tels qu'un fichier d'exportation au format PKCS #12, le fichier nécessite un mot de passe. Le créateur du fichier que vous importez vous fournit le mot de passe PKCS #12.

      ```
      Enter password to use for accessing the PKCS12 file:      Type PKCS #12 password
      ```

   b. **A l'invite, entrez le mot de passe pour le keystore.**

      ```
      Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
      ```

3. **Vérifiez le contenu du keystore.**

```
% pktool list
Found number certificates.
1. (X.509 certificate)
Label:  label-name
ID: fingerprint that binds certificate to private key
Subject: subject-DN
Issuer:  distinguished-name
Serial:  hex-serial-number

2. ...
```

Exemple 4-2 Importation d'un fichier PKCS #12 dans votre keystore

Dans l'exemple suivant, l'utilisateur importe un fichier PKCS #12 d'un tiers. La commande `pktool import` extrait la clé privée et le certificat à partir du fichier `gracedata.p12` et les stocke dans le keystore préféré de l'utilisateur.

```
% pktool import keystore=pkcs11 infile=gracedata.p12 label=GraceCert
Enter password to use for accessing the PKCS12 file:      Type PKCS #12 password
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Found 1 certificate(s) and 1 key(s) in gracedata.p12
% pktool list
No.  Key Type  Key Len.  Key Label
-------------------------------------------------
Asymmetric public keys:
1    RSA                GraceCert
Certificates:
1    X.509 certificate
Label: GraceCert
ID: 71:8f:11:f5:62:10:35:c2:5d:b4:31:38:96:04:80:25:2e:ad:71:b3
Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Serial: 0x00000010
```

Exemple 4-3 Importation d'un certificat X.509 dans votre keystore

Dans l'exemple suivant, l'utilisateur importe un certificat X.509 au format PEM dans le keystore préféré de l'utilisateur. Ce certificat public n'est pas protégé par un mot de passe. Le keystore public de l'utilisateur n'est pas protégé par un mot de passe non plus.

```
% pktool import keystore=pkcs11 infile=somecert.pem label="TheirCompany Root Cert"
% pktool list
No.  Key Type  Key Len.  Key Label
Certificates:
1    X.509 certificate
Label: TheirCompany Root Cert
ID: ec:a2:58:af:83:b9:30:9d:de:b2:06:62:46:a7:34:49:f1:39:00:0e
Subject: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
Issuer: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
Serial: 0x00000001
```

▼ Exportation d'un certificat et de la clé privée au format PKCS #12

Vous pouvez créer un fichier au format PKCS #12 afin d'exporter des clés privées et leur certificat X.509 associé vers d'autres systèmes. L'accès au fichier est protégé par un mot de passe.

1. Recherchez le certificat à exporter.

```
% pktool list
Found number certificates.
1. (X.509 certificate)
Label:  label-name
ID: fingerprint that binds certificate to private key
Subject: subject-DN
Issuer:  distinguished-name
Serial:  hex-serial-number

2. ...
```

2. Exportez les clés et le certificat.

Utilisez le keystore et l'étiquette de la commande `pktool list`. Donnez un nom au fichier d'exportation. Si le nom contient un espace, mettez le nom entre guillemets doubles.

```
% pktool export keystore=keystore outfile=outfile-name label=label-name
```

3. Protégez le fichier d'exportation par un mot de passe.

A l'invite, entrez le mot de passe courant du keystore. A ce stade, vous créez un mot de passe pour le fichier d'exportation. Le destinataire doit fournir ce mot de passe lors de l'importation du fichier.

```
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Enter password to use for accessing the PKCS12 file:     Create PKCS #12 password
```

Astuce - Envoyez le mot de passe séparément du fichier d'exportation. Les pratiques recommandées suggèrent que vous fournissiez le mot de passe hors bande, par exemple lors d'un appel téléphonique.

Exemple 4-4 Exportation d'un certificat et de la clé privée au format PKCS #12

Dans l'exemple suivant, un utilisateur exporte les clés privées avec leur certificat X.509 associé dans un fichier PKCS #12 standard. Ce fichier peut être importé dans d'autres keystores. Le mot de passe PKCS #11 protège le keystore source. Le mot de passe PKCS #12 est utilisé pour protéger des données privées dans le fichier PKCS #12. Ce mot de passe est requis pour importer le fichier.

```
% pktool list
No.  Key Type  Key Len.  Key Label
-------------------------------------------------
Asymmetric public keys:
1    RSA                My Cert
Certificates:
1    X.509 certificate
Label: My Cert
ID: d2:7e:20:04:a5:66:e6:31:90:d8:53:28:bc:ef:55:55:dc:a3:69:93
Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
Serial: 0x000001
```

```
% pktool export keystore=pkcs11 outfile=mydata.p12 label="My Cert"
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Enter password to use for accessing the PKCS12 file:      Create PKCS #12 password
```

L'utilisateur appelle ensuite le destinataire par téléphone pour lui fournir le mot de passe PKCS #12.

▼ Génération d'une phrase de passe à l'aide de la commande `pktool setpin`

Vous pouvez générer une phrase de passe pour un objet dans un keystore et pour le keystore lui-même. La phrase de passe est nécessaire pour accéder à l'objet ou au keystore. Pour consulter un exemple de création d'une phrase de passe pour un objet dans un keystore, reportez-vous à l'Exemple 4-4, “Exportation d'un certificat et de la clé privée au format PKCS #12”.

1. Générez une phrase de passe pour accéder à un keystore.

```
% pktool setpin keystore=nss|pkcs11 [dir=directory]
```

Le répertoire par défaut pour le stockage des clés est `/var/`*username*.

Le mot de passe initial pour un keystore PKCS #11 est `changeme`. Le mot de passe initial pour un keystore NSS est un mot de passe vide.

2. Répondez aux invites.

Lorsque vous êtes invité à saisir le mot de passe du jeton en cours, saisissez le code PIN du jeton pour un keystore PKCS #11, ou appuyez sur la touche Entrée pour un keystore NSS.

```
Enter current token passphrase:      Type PIN or press the Return key
Create new passphrase:               Type the passphrase that you want to use
Re-enter new passphrase:             Retype the passphrase
Passphrase changed.
```

Le keystore est maintenant protégé par une *passphrase*. Si vous perdez la phrase de passe, vous perdez l'accès aux objets dans le keystore.

3. **(Facultatif) Affichez la liste des jetons.**

```
# pktool tokens
```

La sortie varie selon que le metaslot est activé ou non. Pour plus d'informations sur le metaslot, reportez-vous à la section “Concepts propres à la structure cryptographique” à la page 9.

- Si le metaslot est activé, la commande `pktools token` génère une sortie similaire à la suivante :

```
ID Slot   Name                      Token Name                     Flags
--        ---------                 ----------                     -----
0         Sun Metaslot              Sun Metaslot
1         Sun Crypto Softtoken      Sun Software PKCS#11 softtoken  LIX
2         PKCS#11 Interface for TPM TPM                            LXS
```

- Si le metaslot est désactivé, la commande `pktools token` génère une sortie similaire à la suivante :

```
ID Slot   Name                      Token Name                     Flags
--        ---------                 ----------                     -----
1         Sun Crypto Softtoken      Sun Software PKCS#11 softtoken  LIX
2         PKCS#11 Interface for TPM TPM                            LXS
```

Dans les deux cas, les indicateurs peuvent être une combinaison quelconque des éléments suivants :

- `L` : connexion requise
- `I` : initialisé
- `X` : code confidentiel utilisateur expiré
- `S` – code confidentiel SO expiré

Exemple 4-5 Protection d'un keystore par une phrase de passe

L'exemple suivant montre comment définir la phrase de passe pour une base de données NSS. Comme aucune phrase de passe n'a été créée, l'utilisateur appuie sur la touche Entrée à la première invite.

```
% pktool setpin keystore=nss dir=/var/nss
Enter current token passphrase:      Press the Return key
Create new passphrase: has8n0NdaH
Re-enter new passphrase: has8n0NdaH
Passphrase changed.
```

Génération d'une paire de clés à l'aide de la commande `pktool genkeypair`

Certaines applications exigent une paire de clés publique/privée. Dans cette procédure, vous créez ces paires de clés et les stockez.

1. **(Facultatif) Si vous prévoyez d'utiliser un keystore, créez-le.**

   - **Pour créer et initialiser un keystore PKCS #11, reportez-vous à la section "Génération d'une phrase de passe à l'aide de la commande `pktool setpin`" à la page 61.**

   - **Pour créer et initialiser un keystore NSS, reportez-vous à l'Exemple 4-5, "Protection d'un keystore par une phrase de passe".**

2. **Créez la paire de clés.**

   Choisissez l'une des méthodes suivantes.

   - **Créez la paire de clés et stockez-la dans un fichier.**

     Les clés basées sur des fichiers sont créées pour les applications qui lisent les clés directement à partir de fichiers sur le disque. En règle générale, les applications qui utilisent directement les bibliothèques de chiffrement OpenSSL requièrent le stockage des clés et des certificats pour l'application dans des fichiers.

Remarque - Le keystore `file` ne prend pas en charge les clés et les certificats de courbes elliptiques (`ec`) .

```
% pktool genkeypair keystore=file outkey=key-filename \
[format=der|pem] [keytype=rsa|dsa] [keylen=key-size]
```

`keystore=file`

La valeur `file` spécifie le type de fichier dans l'emplacement de stockage de la clé.

`outkey=`*key-filename*

Spécifie le nom du fichier de stockage de la paire de clés.

`format=der|pem`

Spécifie le format de codage de la paire de clés. La sortie `der` est binaire et la sortie `pem` est ASCII.

`keytype=rsa|dsa`

Spécifie le type de paire de clés qui peut être stockée dans un keystore `file`. Pour obtenir des définitions, reportez-vous à DSA et RSA.

`keylen=`*key-size*

Spécifie la longueur de la clé en bits. Le nombre doit être divisible par 8. Pour déterminer les tailles de clés possibles, utilisez la commande `cryptoadm list -vm`.

- **Créez la paire de clés et stockez-la dans un keystore PKCS #11.**

  Vous devez effectuer l'Étape 1 avant d'utiliser cette méthode.

  Le keystore PKCS #11 permet de stocker des objets sur un périphérique matériel. Le périphérique peut être une carte Sun Crypto Accelerator 6000, un périphérique TPM (Trusted Platform Module, module de plate-forme de confiance) ou une carte à puce branchée à la structure cryptographique. PKCS #11 peut également être utilisé pour stocker des objets dans le `softtoken`, ou jeton logiciel, qui stocke les objets dans un sous-répertoire privé sur le disque. Pour plus d'informations, reportez-vous à la page de manuel `pkcs11_softtoken`(5).

  Vous pouvez récupérer la paire de clés dans le keystore par une étiquette que vous indiquez.

  ```
  % pktool genkeypair label=key-label \
  [token=token[:manuf[:serial]]] \
  [keytype=rsa|dsa|ec]  [curve=ECC-Curve-Name]]\
  [keylen=key-size] [listcurves]
  ```

  `label=`*key-label*

  Spécifie une étiquette pour la paire de clés. La paire de clés peut être récupérée à partir du keystore par son étiquette.

  `token=`*token*[:*manuf*[:*serial*]]

  Spécifie le nom du jeton. Par défaut, le nom du jeton est `Sun Software PKCS#11 softtoken`.

  `keytype=rsa|dsa|ec [curve=`*ECC-Curve-Name*`]`

  Spécifie le type de la paire de clés. Pour le type de courbe elliptique (`ec`), vous pouvez éventuellement spécifier le nom de la courbe. Les noms de courbes sont répertoriés en tant que sortie de l'option `listcurves`.

  `keylen=`*key-size*

  Spécifie la longueur de la clé en bits. Le nombre doit être divisible par 8.

`listcurves`

Répertorie les noms de courbes elliptiques qui peuvent être utilisés comme valeurs de l'option `curve=` pour un type de clé `ec`.

- **Générez la paire de clés et stockez-la dans un keystore NSS.**

  Le keystore NSS est utilisé par les serveurs qui utilisent NSS comme interface de chiffrement principale.

  Vous devez effectuer l'Étape 1 avant d'utiliser cette méthode.

```
% pktool keystore=nss genkeypair label=key-nickname \
[token=token[:manuf[:serial]]] \
[dir=directory-path] [prefix=database-prefix] \
[keytype=rsa|dsa|ec] [curve=ECC-Curve-Name]] \
[keylen=key-size] [listcurves]
```

`keystore=nss`

La valeur `nss` spécifie le type NSS de l'emplacement de stockage de la clé.

`label=`*nickname*

Spécifie une étiquette pour la paire de clés. La paire de clés peut être récupérée à partir du keystore par son étiquette.

`token=`*token*[:*manuf*[:*serial*]]

Spécifie le nom du jeton. Par défaut, le jeton est `Sun Software PKCS#11 softtoken`.

`dir=`*directory*

Spécifie le chemin d'accès au répertoire de la base de données NSS. Par défaut, *directory* est le répertoire courant.

`prefix=`*database-prefix*

Spécifie le préfixe de la base de données NSS. Par défaut, le champ de préfixe est vide.

`keytype=rsa|dsa|ec [curve=`*ECC-Curve-Name*`]`

Spécifie le type de la paire de clés. Pour le type de courbe elliptique, vous pouvez éventuellement spécifier le nom de la courbe. Les noms de courbes sont répertoriés en tant que sortie de l'option `listcurves`.

`keylen=`*key-size*

Spécifie la longueur de la clé en bits. Le nombre doit être divisible par 8.

`listcurves`

Répertorie les noms de courbes elliptiques qui peuvent être utilisés comme valeurs de l'option `curve=` pour un type de clé `ec`.

3. (Facultatif) Vérifiez que la clé existe.

Utilisez l'une des commandes suivantes, en fonction de l'emplacement où vous avez stocké la clé :

- **Vérifiez la clé dans le fichier** *key-filename***.**

```
% pktool list keystore=file objtype=key infile=key-filename
Found n keys.
Key #1 - keytype:location (keylen)
```

- **Vérifiez la clé dans le keystore PKCS #11.**

```
$ pktool list objtype=key
Enter PIN for keystore:
Found n keys.
Key #1 - keytype:location (keylen)
```

- **Vérifiez la clé dans le keystore NSS.**

```
% pktool list keystore=nss dir=directory objtype=key
```

Exemple 4-6 Création d'une paire de clés à l'aide de la commande `pktool`

Dans l'exemple ci-dessous, un utilisateur crée un keystore PKCS #11 pour la première fois. Après avoir déterminé les tailles de clé des paires de clés RSA, l'utilisateur génère une paire de clés pour une application. Enfin, l'utilisateur vérifie que la paire de clés figure dans le keystore. L'utilisateur constate que la seconde instance de la paire de clés RSA peut être stockée sur le matériel. Etant donné que l'utilisateur ne spécifie pas un argument `token`, la paire de clés est stockée sous forme d'un `Sun Software PKCS#11 softtoken`.

```
# pktool setpin
Create new passphrase:
Re-enter new passphrase:     Retype password
Passphrase changed.
% cryptoadm list -vm | grep PAIR
...
CKM_DSA_KEY_PAIR_GEN       512  3072 .  .  .  .  .  .  .  .  .  X  .  .  .  .
CKM_RSA_PKCS_KEY_PAIR_GEN  256  8192 .  .  .  .  .  .  .  .  .  X  .  .  .  .
...
CKM_RSA_PKCS_KEY_PAIR_GEN  256  2048 X  .  .  .  .  .  .  .  .  X  .  .  .  .
ecc: CKM_EC_KEY_PAIR_GEN,CKM_ECDH1_DERIVE,CKM_ECDSA,CKM_ECDSA_SHA1
% pktool genkeypair label=specialappkeypair keytype=rsa keylen=2048
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
```

```
% pktool list
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
No.      Key Type      Key Len.      Key Label
-------------------------------------------------
Asymmetric public keys:
1        RSA                         specialappkeypair
```

Exemple 4-7 Création d'une paire de clés qui utilise l'algorithme de courbe elliptique

Dans l'exemple suivant, un utilisateur ajoute une paire de clés de courbe elliptique (ec) dans le keystore, spécifie un nom de courbe et vérifie que la paire de clés figure dans le keystore.

```
% pktool genkeypair listcurves
secp112r1, secp112r2, secp128r1, secp128r2, secp160k1
.
.
.
c2pnb304w1, c2tnb359v1, c2pnb368w1, c2tnb431r1, prime192v2
prime192v3
% pktool genkeypair label=eckeypair keytype=ec curves=c2tnb431r1
% pktool list
Enter PIN for Sun Software PKCS#11 softtoken  :     Type password
No.  Key Type  Key Len.  Key Label
-------------------------------------------------
Asymmetric public keys:
1    ECDSA              eckeypair
```

▼ Signature d'une demande de certificat à l'aide de la commande `pktool signcsr`

Cette procédure est utilisée pour signer une CSR (demande de signature du certificat) PKCS #10. La CSR peut être au format PEM ou DER. Le processus de signature émet un certificat X.509 v3. Pour générer une demande de signature de certificat PKCS #10, reportez-vous à la page de manuel pktool(1).

Avant de commencer Cette procédure suppose que vous êtes une autorité de certification (CA), que vous avez reçu une CSR et qu'elle est stockée dans un fichier.

1. **Recueillez les informations suivantes pour les arguments requis de la commande `pktool signcsr` :**

`signkey` Si vous avez stocké la clé du signataire dans un keystore PKCS #11, `signkey` est l'étiquette qui permet de récupérer cette clé privée.

Si vous avez stocké la clé du signataire dans un keystore NSS, `signkey` est le nom du fichier qui contient cette clé privée.

`csr`	Spécifie le nom de fichier de la CSR.
`serial`	Spécifie le numéro de série du certificat signé.
`outcer`	Spécifie le nom de fichier du certificat signé.
`issuer`	Spécifie votre nom d'émetteur CA au format DN (nom distinctif).

Pour plus d'informations sur les arguments facultatifs de la sous-commande `signcsr`, reportez-vous à la page de manuel `pktool`(1).

2. **Signez la demande et émettez le certificat.**

   Par exemple, la commande suivante signe le certificat avec la clé du signataire figurant dans le référentiel PKCS #11 :

```
# pktool signcsr signkey=CASigningKey \
csr=fromExampleCoCSR \
serial=0x12345678 \
outcert=ExampleCoCert2010 \
issuer="O=Oracle Corporation, \
OU=Oracle Solaris Security Technology, L=Redwood City, ST=CA, C=US, \
CN=rootsign Oracle"
```

   La commande suivante signe le certificat avec la clé du signataire figurant dans un fichier :

```
# pktool signcsr signkey=CASigningKey \
csr=fromExampleCoCSR \
serial=0x12345678 \
outcert=ExampleCoCert2010 \
issuer="O=Oracle Corporation, \
OU=Oracle Solaris Security Technology, L=Redwood City, ST=CA, C=US, \
CN=rootsign Oracle"
```

3. **Envoyez le certificat au demandeur.**

   Vous pouvez utiliser un e-mail, un site web ou un autre mécanisme pour remettre le certificat au demandeur.

   Par exemple, vous pouvez utiliser votre messagerie pour envoyer le fichier `ExampleCoCert2010` au demandeur.

▼ Gestion des plug-ins tiers dans KMF

Vous identifiez votre plug-in en lui attribuant un nom de keystore. Lorsque vous ajoutez le plug-in à la KMF, le logiciel l'identifie par son nom de keystore. Le plug-in peut être défini de manière à accepter une option. Cette procédure inclut la suppression du plug-in de la KMF.

1. **Installez le plug-in.**

```
% /usr/bin/kmfcfg install keystore=keystore-name \
modulepath=path-to-plugin [option="option-string"]
```

où

keystore-name	Spécifie un nom unique pour le keystore que vous fournissez.
path-to-plugin	Spécifie le chemin d'accès complet à l'objet de bibliothèque partagée pour le plug-in KMF.
option-string	Spécifie un argument facultatif de l'objet de bibliothèque partagée.

2. **Répertoriez les plug-ins.**

```
% kmfcfg list plugin
keystore-name:path-to-plugin [(built-in)] | [;option=option-string]
```

3. **Pour supprimer le plug-in, désinstallez-le et vérifiez sa suppression.**

```
% kmfcfg uninstall keystore=keystore-name
% kmfcfg plugin list
```

Exemple 4-8 Appel d'un plug-in KMF avec une option

Dans l'exemple suivant, l'administrateur stocke un plug-in KMF dans un répertoire spécifique au site. Le plug-in est défini pour accepter une option `debug`. L'administrateur ajoute le plug-in et vérifie qu'il est installé.

```
# /usr/bin/kmfcfg install keystore=mykmfplug \
modulepath=/lib/security/site-modules/mykmfplug.so
# kmfcfg list plugin
KMF plugin information:
----------------------
pkcs11:kmf_pkcs11.so.1 (built-in)
file:kmf_openssl.so.1 (built-in)
nss:kmf_nss.so.1 (built-in)
mykmfplug:/lib/security/site-modules/mykmfplug.so
# kmfcfg modify plugin keystore=mykmfplug option="debug"
# kmfcfg list plugin
KMF plugin information:
----------------------
...
mykmfplug:/lib/security/site-modules/mykmfplug.so;option=debug
```

Le plug-in s'exécute à présent en mode de débogage.

Glossaire de la sécurité

AES Standard de chiffrement avancé (Advanced Encryption Standard). Technique de chiffrement de données symétrique par blocs de 128 bits. Le gouvernement des Etats-Unis a adopté la variante Rijndael de l'algorithme comme norme de chiffrement en octobre 2000. AES remplace le chiffrement principal d'utilisateur comme norme administrative.

algorithme Algorithme cryptographique. Il s'agit d'une procédure de calcul récursive établie qui chiffre ou hache une entrée.

algorithme cryptographique Voir algorithme.

allocation de périphériques Protection des périphériques au niveau de l'utilisateur. L'allocation de périphériques met en oeuvre l'utilisation exclusive d'un périphérique par un utilisateur à la fois. Les données des périphériques sont purgées avant toute réutilisation d'un périphérique. Des autorisations peuvent être utilisées pour limiter les utilisateurs autorisés à allouer un périphérique.

amorce Valeur numérique de départ pour la génération de nombres aléatoires. Lorsque cette valeur provient d'une source aléatoire, l'amorce est appelée *amorce aléatoire*.

application privilégiée Application pouvant remplacer les contrôles système. L'application vérifie les attributs de sécurité, tels que des UID spécifiques, des ID de groupe, des autorisations ou des privilèges.

attributs de sécurité Remplace la stratégie de sécurité qui permet à une commande d'administration de s'exécuter correctement lorsque celle-ci est exécutée par un utilisateur autre que le superutilisateur. Dans le modèle de superutilisateur, les programmes `setuid root` et `setgid` sont des attributs de sécurité. Lorsque ces attributs sont appliqués à une commande, la commande s'exécute correctement, quel que soit l'utilisateur qui l'exécute. Dans le modèle de privilège, les privilèges de noyau et les autres droits remplacent les programmes `setuid root` en tant qu'attributs de sécurité. Le modèle de privilège est compatible avec le modèle de superutilisateur dans la mesure où le modèle de privilège reconnaît également les programmes `setuid` et `setgid` comme des attributs de sécurité.

authentificateur Des authentificateurs sont transmis par des clients lors de la demande de tickets (à un KDC) et de services (à un serveur). Ils contiennent des informations générées par l'utilisation d'une clé de session connue uniquement du client et du serveur, dont l'origine récente peut être prouvée, indiquant ainsi que la transaction est sécurisée. Lorsqu'un authentificateur est utilisé avec un ticket, il peut permettre d'authentifier un principal d'utilisateur. Un authentificateur inclut le

nom du principal de l'utilisateur, l'adresse IP de l'hôte de l'utilisateur, ainsi qu'un horodatage. A la différence d'un ticket, un authentificateur ne peut servir qu'une seule fois, généralement lorsque l'accès à un service est demandé. Un authentificateur est chiffré à l'aide de la clé de session pour ce client et ce serveur.

authentification Processus de vérification de l'identité déclarée d'un principal.

autorisation 1. Dans Kerberos, processus consistant à déterminer si un principal peut utiliser un service et à définir les objets auxquels il peut accéder, ainsi que le type d'accès autorisé pour chaque objet.

2. Dans la gestion des droits des utilisateurs, droit pouvant être affecté à un rôle ou un utilisateur (ou intégré dans un profil de droits) en vue de l'exécution d'une classe d'opérations autrement interdites par la stratégie de sécurité. Les autorisations sont mises en oeuvre au niveau de l'application utilisateur, et non pas dans le noyau.

Blowfish Algorithme de chiffrement par bloc symétrique de longueur de clé variable (entre 32 et 448 bits). Son créateur, Bruce Schneier, affirme que Blowfish est optimisé pour les applications pour lesquelles la clé n'a pas besoin d'être régulièrement modifiée.

cache d'informations d'identification Espace de stockage (généralement un fichier) contenant des informations d'identification reçues de KDC.

certificat Un certificat de clé publique est un ensemble de données qui encode une valeur de clé publique. Il inclut certaines informations concernant la génération du certificat, comme son nom et l'auteur de sa signature, un hachage ou une somme de contrôle du certificat et une signature numérique du hachage. Ensemble, ces valeurs constituent le certificat. La signature numérique garantit que le certificat n'a pas été modifié.

Pour plus d'informations, voir clé.

champ d'application du service de noms Champ d'application dans lequel un rôle est autorisé à fonctionner, c'est-à-dire un hôte particulier ou tous les hôtes desservis par un service de noms spécifié, tel que NIS LDAP.

chiffrement par clé privée Dans le cas du chiffrement par clé privée, l'expéditeur et le destinataire utilisent la même clé de chiffrement. Voir également chiffrement par clé publique.

chiffrement par clé publique Modèle de chiffrement où chaque utilisateur dispose de deux clés, l'une publique et l'autre privée. Dans le cas du chiffrement par clé publique, l'expéditeur chiffre le message à l'aide de la clé publique du destinataire, et ce dernier se sert d'une clé privée pour le déchiffrer. Le service Kerberos est un système à clé privée. Voir également chiffrement par clé privée.

clé 1. En règle générale, l'un des deux principaux types de clés :

- *clé symétrique* : clé de chiffrement identique à la clé de déchiffrement. Les clés symétriques sont utilisées pour chiffrer des fichiers.

- *clé asymétrique* ou *clé publique* : clé utilisée dans les algorithmes à clé publique, tels que Diffie-Hellman ou RSA. Les clés publiques contiennent une clé privée, connue uniquement d'un utilisateur, une clé publique utilisée par le serveur ou des ressources générales, et une paire de clés publique-privée combinant les deux. Une clé privée est également qualifiée de clé *secrète*. La clé publique est également qualifiée de clé *partagée* ou *commune*.

2. Entrée (nom de principal) dans un fichier keytab. Voir également fichier keytab.

3. Dans Kerberos, clé de chiffrement dont il existe trois types :

- *Clé privée* : clé de chiffrement partagée par un principal et le KDC, et distribuée en dehors des limites du système. Voir également clé privée.
- *Clé de service* : clé remplissant la même fonction que la clé privée, mais utilisée par des serveurs et des services. Voir également clé de service.
- *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. Voir également clé de session.

clé de service Clé de chiffrement partagée par un principal de service et le KDC, et distribuée en dehors des limites du système. Voir également clé.

clé de session Clé générée par le service d'authentification ou le service d'octroi de ticket. Une clé de session est générée dans le but de sécuriser les transactions entre un client et un service. La durée de vie d'une clé de session est limitée à une seule session de connexion. Voir également clé.

clé privée Chaque principal d'utilisateur reçoit une clé qui n'est connue que du KDC et de l'utilisateur du principal. Pour les principaux d'utilisateur, la clé est basée sur le mot de passe de l'utilisateur. Voir également clé.

clé secrète Voir clé privée.

client Au sens strict, il s'agit d'un processus utilisant un service réseau pour le compte d'un utilisateur, par exemple, une application utilisant `rlogin`. Dans certains cas, un serveur peut être lui-même le client d'un autre serveur ou service.

Au sens large, il s'agit d'un hôte qui a) reçoit des informations d'identification Kerberos et b) utilise un service fourni par un serveur.

Dans la pratique, ce terme désigne un principal utilisant un service.

confidentialité Voir confidentialité.

confidentialité Un service de sécurité, dans lequel les données transmises sont chiffrées avant leur envoi. La confidentialité inclut également l'intégrité des données et l'authentification de l'utilisateur. Voir également authentification, intégrité, service.

conscient des privilèges Programmes, scripts et commandes qui activent et désactivent l'utilisation des privilèges dans leur code. Dans un environnement de production, les privilèges qui sont activés doivent être fournis au processus, par exemple, en demandant aux utilisateurs du programme d'utiliser un

profil de droits qui ajoute les privilèges au programme. Pour une description complète des privilèges, reportez-vous à la page de manuel `privileges`(5).

consommateur Dans la fonctionnalité Structure cryptographique d'Oracle Solaris, un consommateur est un utilisateur des services cryptographiques provenant de fournisseurs. Les consommateurs peuvent être des applications, des utilisateurs finaux ou des opérations de noyau. Kerberos, IKE et IPsec sont des exemples de consommateurs. Pour consulter des exemples de fournisseurs, reportez-vous à la section fournisseur.

DES Standard de chiffrement de données (Data Encryption Standard). Méthode de chiffrement à clé symétrique développée en 1975 et standardisée par l'ANSI en 1981 comme ANSI X.3.92. Le DES utilise une clé de 56 bits.

domaine 1. Réseau logique desservi par une seule base de données Kerberos et un ensemble de centre de distribution des clés (KDC).

2. Troisième partie d'un nom de principal. Pour le nom de principal `jdoe/admin@CORP.EXAMPLE.COM`, le domaine est `CORP.EXAMPLE.COM`. Voir également nom de principal.

droits Alternative au modèle tout ou rien des superutilisateurs. La gestion des droits des utilisateurs et la gestion des droits de processus permettent à une organisation de répartir les privilèges du superutilisateur et de les affecter à des utilisateurs ou rôles. Les droits dans Oracle Solaris sont implémentés en tant que privilèges de noyau, autorisations et la capacité à exécuter un processus sous la forme d'un UID ou GID spécifique. Les droits peuvent être collectés dans un profil de droits et un rôle.

DSA Algorithme de signature numérique (Digital Signature Algorithm). Algorithme de clé publique dont la longueur de clé varie de 512 à 4 096 bits. La norme du gouvernement américain, DSS, atteint 1 024 bits. L'algorithme DSA repose sur l'algorithme SHA1 en entrée.

écart d'horloge Ecart maximal toléré entre les horloges système internes de tous les hôtes participant au système d'authentification Kerberos. Si l'écart d'horloge est dépassé entre des hôtes participants, les demandes sont rejetées. L'écart d'horloge est spécifié dans le fichier `krb5.conf`.

ECDSA Algorithme de signature numérique de courbe elliptique. Algorithme de clé publique basé sur une courbe elliptique mathématique. Une clé ECDSA est de beaucoup plus petite taille qu'une clé publique DSA nécessaire pour générer une signature de la même longueur.

escalade des privilèges Obtention de l'accès à des ressources situées en dehors de la gamme de ressources que les droits qui vous ont été octroyés, y compris les droits qui remplacent les valeurs par défaut, autorisent. Il en résulte qu'un processus peut effectuer des opérations non autorisées.

événement d'audit asynchrone Les événements asynchrones constituent la minorité des événements système. Ces événements ne sont associés à aucun processus, de sorte qu'aucun processus ne peut être bloqué puis

réactivé ultérieurement. L'initialisation système initiale et les événements d'entrée et de sortie PROM sont des exemples d'événements asynchrones.

événement d'audit non allouable Evénement d'audit dont l'initiateur ne peut pas être déterminé, tel que l'événement AUE_BOOT.

événement d'audit synchrone Majorité des événements d'audit. Ces événements sont associés à un processus dans le système. Un événement non allouable associé à un processus est un événement synchrone, tel que l'échec d'une connexion.

fichier de ticket Voir cache d'informations d'identification.

fichier keytab Fichier de table de clés contenant une ou plusieurs clés (principaux). Un hôte ou un service utilise un fichier keytab à peu près de la même façon qu'un utilisateur se sert d'un mot de passe.

fichier stash Un fichier stash contient une copie chiffrée de la clé principale pour le KDC. Cette clé principale est utilisée lorsqu'un serveur est réinitialisé pour authentifier automatiquement le KDC avant qu'il ne démarre les processus `kadmind` et `krb5kdc`. Etant donné que le fichier stash inclut la clé principale, ce fichier et toutes ses sauvegardes doivent être sécurisés. Si le chiffrement est compromis, la clé peut être utilisée pour accéder à la base de données KDC ou la modifier.

fichiers d'audit Journaux d'audit binaires. Les fichiers d'audit sont stockés séparément dans un système de fichiers d'audit.

fournisseur Dans la fonctionnalité Structure cryptographique d'Oracle Solaris, service cryptographique fourni aux consommateurs. Les bibliothèques PKCS #11, les modules cryptographiques du noyau et les accélérateurs de matériel sont des exemples de fournisseurs. Les fournisseurs se connectent à la structure cryptographique et sont donc également appelés *plug-ins*. Pour consulter des exemples de consommateurs, voir consommateur.

fournisseur de logiciels Dans la fonctionnalité Structure cryptographique d'Oracle Solaris, module logiciel de noyau ou bibliothèque PKCS#11 fournissant des services cryptographiques. Voir également fournisseur.

fournisseur de matériel Dans la fonctionnalité Structure cryptographique d'Oracle Solaris, un pilote de périphérique et son accélérateur matériel. Les fournisseurs de matériel déchargent le système informatique d'opérations cryptographiques coûteuses, libérant ainsi les ressources de la CPU pour d'autres utilisations. Voir également fournisseur.

FQDN Nom de domaine complet. Par exemple, `central.example.com` (et pas simplement `denver`).

GSS-API Interface de programmation d'application générique de service de sécurité. Couche réseau assurant la prise en charge de différents services de sécurité modulaires, y compris le service Kerberos. GSS-API fournit des services d'authentification, d'intégrité et de confidentialité. Voir également authentification, intégrité, confidentialité.

hôte Système accessible par l'intermédiaire d'un réseau.

hôte principal Instance spécifique d'un principal de service dans lequel le principal (indiqué par le nom primaire `host`) est configuré de manière à fournir une gamme de services réseau, comme `ftp`, `rcp` ou `rlogin`. `host/central.example.com@EXAMPLE.COM` est un exemple d'hôte principal. Voir également principal de serveur.

image système unique Une image système unique est utilisée dans l'audit d'Oracle Solaris afin de décrire un groupe de systèmes audités utilisant le même service de noms. Ces systèmes envoient leurs enregistrements d'audit à un serveur d'audit central où les enregistrements peuvent être comparés comme s'ils provenaient d'un même système.

informations d'identification Package d'informations comprenant un ticket et une clé de session correspondante. Informations utilisées pour authentifier l'identité d'un principal. Voir également ticket, clé de session.

instance Deuxième partie d'un nom de principal, une instance qualifie le primaire du principal. Dans le cas d'un principal de service, l'instance est requise. L'instance est le nom de domaine complet de l'hôte, comme dans `host/central.example.com`. Pour les principaux d'utilisateur, une instance est facultative. Notez, cependant, que `jdoe` et `jdoe/admin` sont des principaux uniques. Voir également primaire, nom de principal, principal de service, principal d'utilisateur.

intégrité Service de sécurité qui assure, outre l'authentification de l'utilisateur, la validité des données transmises par le biais de sommes de contrôle cryptographiques. Voir également authentification, confidentialité.

jeu autorisé Jeu des privilèges disponibles pour l'utilisation par un processus.

jeu de base Jeu de privilèges affecté à un processus d'utilisateur lors de la connexion. Sur un système non modifié, le jeu héritable initial de chaque utilisateur correspond au jeu de base défini au moment de la connexion.

jeu de privilèges Ensemble de privilèges. Chaque processus comporte quatre jeux de privilèges qui déterminent s'il peut utiliser un privilège particulier. Voir jeu limite, jeu effectif, jeu autorisé et jeu hérité.

De même, le jeu de base de privilèges est l'ensemble des privilèges affectés aux processus d'un utilisateur au moment de la connexion.

jeu effectif Jeu de privilèges actuellement en vigueur sur un processus.

jeu hérité Jeu de privilèges dont un processus peut hériter en appelant `exec`.

jeu limite Limite extérieure des privilèges disponibles pour un processus et ses enfants.

KDC Centre de distribution de clés (Key Distribution Center). Machine disposant de trois composants Kerberos V5.

- Principal et base de données de clés

- Service d'authentification
- Service d'octroi de tickets

Chaque domaine dispose d'un KDC maître et doit avoir un ou plusieurs KDC esclaves.

KDC esclave Copie d'un KDC maître capable de réaliser la plupart des fonctions du maître. Chaque domaine dispose généralement de plusieurs KDC esclaves et d'un seul KDC maître. Voir également KDC, KDC maître.

KDC maître KDC maître dans chaque domaine, comprenant un serveur d'administration Kerberos, `kadmind` et un démon d'authentification et d'octroi de tickets, `krb5kdc`. Chaque domaine doit disposer d'au moins un KDC maître, et peut avoir plusieurs KDC de duplication ou esclaves fournissant des services d'authentification aux clients.

Kerberos Service d'authentification, protocole utilisé par ce service ou code servant à mettre en oeuvre ce service.

Mise en oeuvre Kerberos d'Oracle Solaris étroitement basée sur la mise en oeuvre Kerberos V5.

Bien que techniquement différents, "Kerberos" et "Kerberos V5" sont souvent utilisés de façon interchangeable dans la documentation Kerberos.

Dans la mythologie grecque, Kerberos (en français Cerbère) était un chien féroce tricéphale qui gardait la porte des Enfers.

keystore Un keystore contient des mots de passe, des phrases de passe, des certificats et d'autres objets d'authentification pour l'extraction par des applications. Il peut être spécifique à une technologie, ou à un emplacement utilisé par plusieurs applications.

kvno Numéro de version de la clé. Numéro de série faisant le suivi d'une clé spécifique selon l'ordre dans lequel elle a été générée. Plus le numéro kvno est élevé, plus la clé est récente.

liste de contrôle d'accès (ACL) Une liste de contrôle d'accès (ACL) offre une sécurité des fichiers plus précise que la protection de fichier UNIX conventionnelle. Par exemple, une ACL vous permet d'autoriser l'accès en lecture de groupe à un fichier, tout en autorisant un seul membre de ce groupe à écrire dans le fichier.

MAC 1. Voir MAC.

2. Egalement appelé étiquetage. Dans la terminologie de sécurité gouvernementale, MAC signifie Mandatory Access Control (contrôle d'accès obligatoire). Les étiquettes telles que Top Secret et Confidential sont des exemples de MAC. MAC diffère de DAC (Discretionary Access Control, contrôle d'accès discrétionnaire). Les autorisations UNIX constituent un exemple de DAC.

3. Dans le matériel, il s'agit de l'adresse système unique sur un réseau local (LAN). Si le système est sur un réseau Ethernet, le MAC est l'adresse Ethernet.

MAC MAC garantit l'intégrité des données et authentifie leur origine. MAC ne protège aucunement contre l'écoute frauduleuse des informations échangées.

MD5 Fonction de hachage cryptographique répétitive utilisée pour authentifier les messages, y compris les signatures numériques. Elle a été développée en 1991 par Rivest. Son utilisation a été désapprouvée.

mécanisme 1. Package logiciel spécifiant des techniques cryptographiques pour assurer l'authentification ou la confidentialité des données. Exemples : Kerberos V5, clé publique Diffie-Hellman

2. Dans la fonctionnalité Structure cryptographique d'Oracle Solaris, mise en oeuvre d'un algorithme destiné à un usage particulier. Par exemple, un mécanisme DES appliqué à l'authentification, tel que CKM_DES_MAC, est un mécanisme distinct d'un mécanisme DES appliqué au chiffrement, CKM_DES_CBC_PAD.

mécanisme de sécurité Voir mécanisme.

minimisation Installation du système d'exploitation minimal nécessaire pour l'exécution du serveur. Tout logiciel n'étant pas directement lié au fonctionnement du serveur n'est pas installé, ou est supprimé après l'installation.

modèle de privilège Modèle de sécurité plus strict que le modèle superutilisateur sur un système informatique. Dans le modèle de privilège, les processus nécessitent des privilèges pour s'exécuter. L'administration du système peut être divisée en quatre parties discrètes basées sur les privilèges dont les administrateurs disposent dans leurs processus. Des privilèges peuvent être affectés au processus de connexion d'un administrateur, ou des privilèges peuvent être affectés de manière à ne s'appliquer qu'à certaines commandes.

modèle de superutilisateur Modèle de sécurité UNIX standard sur un système informatique. Dans le modèle de superutilisateur, un administrateur dispose d'un contrôle de type tout ou rien sur le système. En règle générale, pour l'administration de la machine, un utilisateur se connecte en tant que superutilisateur (`root`) et peut effectuer toutes les activités d'administration.

moindre privilège Modèle de sécurité qui octroie à un processus spécifié uniquement un sous-ensemble de pouvoirs de superutilisateur. Le modèle de moindre privilège octroie aux utilisateurs standard les privilèges suffisants pour qu'ils puissent effectuer des tâches d'administration personnelles, telles que le montage de systèmes de fichiers et la modification de l'appartenance des fichiers. Par ailleurs, les processus s'exécutent uniquement avec les privilèges dont ils ont besoin pour terminer la tâche, et non avec tous les pouvoirs du superutilisateur, c'est-à-dire, tous les privilèges. Les dommages causés par les erreurs de programmation, comme les débordements de la mémoire tampon, peuvent être limités à un utilisateur non root, qui n'a pas accès à des fonctions essentielles comme la lecture ou l'écriture de fichiers système protégés ou l'arrêt de la machine.

moteur d'analyse Application tierce, résidant sur un hôte externe, qui examine un fichier à la recherche de virus connus.

nom de principal
1. Le nom d'un principal, au format *primary/instance@REALM*. Voir également instance, primaire, domaine.

2. (RPCSEC_GSS API) Voir principal de client, principal de serveur.

NTP
Network Time Protocol. Logiciel de l'Université de l'Etat du Delaware qui vous permet de gérer avec précision la synchronisation de l'heure ou de l'horloge réseau, ou les deux, dans un environnement réseau. Vous pouvez utiliser le protocole NTP pour préserver l'écart d'horloge dans un environnement Kerberos. Voir également écart d'horloge.

objet public
Fichier appartenant à l'utilisateur `root` et lisible par tout le monde, tel que n'importe quel fichier du répertoire `/etc`.

PAM
Module d'authentification enfichable (Pluggable Authentication Module). Structure permettant d'utiliser plusieurs mécanismes d'authentification sans recompilation des services recourant à ces mécanismes. PAM permet l'initialisation de la session SEAM au moment de son ouverture

phrase de passe
Phrase utilisée pour vérifier qu'une clé privée a été créée par l'utilisateur de la phrase de passe. Une bonne phrase de passe contient 10 à 30 caractères alphanumériques et évite les noms et proses simples. Vous êtes invité à saisir la phrase de passe pour authentifier l'utilisation de la clé privée pour chiffrer et déchiffrer les communications.

piste d'audit
Collection de tous les fichiers d'audit provenant de tous les hôtes.

primaire
Première partie du nom d'un nom de principal. Voir également instance, nom de principal, domaine.

principal
1. Client/utilisateur dont le nom est unique ou instance de serveur/service participant à une communication en réseau. Les transactions Kerberos impliquent des interactions entre des principaux (principaux de service et d'utilisateur) ou entre des principaux et des KDC. En d'autres termes, un principal est une entité unique à laquelle Kerberos peut attribuer des tickets. Voir également nom de principal, principal de service, principal d'utilisateur.

2. (RPCSEC_GSS API) Voir principal de client, principal de serveur.

Principal admin
Principal d'utilisateur dont le nom est au format *nom_utilisateur*`/admin` (comme dans `jdoe/admin`). Un principal admin peut disposer de davantage de privilèges (de modification de stratégies par exemple) qu'un principal d'utilisateur standard. Voir également nom de principal et principal d'utilisateur.

principal d'utilisateur
Principal attribué à un utilisateur particulier. Le nom primaire d'un principal d'utilisateur est un nom d'utilisateur et son instance facultative est un nom utilisé pour décrire l'utilisation prévue des informations d'identification correspondantes (`jdoe` ou `jdoe/admin` par exemple). Egalement appelé instance d'utilisateur. Voir également principal de service.

principal de client
(RPCSEC_GSS API) Client (utilisateur ou application) utilisant des services réseau sécurisés RPCSEC_GSS. Les noms de principaux client sont stockés sous forme de structures `rpc_gss_principal_t`.

principal de serveur (RPCSEC_GSS API) Principal fournissant un service. Le principal de serveur est stocké sous forme d'une chaîne de caractères ASCII dont le format est *service*@*hôte*. Voir également principal de client.

principal de service Principal assurant l'authentification Kerberos pour un ou plusieurs services. Pour les principaux de service, le nom primaire est un nom de service, tel que `ftp`, et son instance est le nom d'hôte complet du système fournissant le service. Voir également hôte principal, principal d'utilisateur.

principe du moindre privilège Reportez-vous à moindre privilège.

privilège 1. En règle générale, aptitude ou capacité à effectuer une opération sur un système informatique qui se situe au-delà des pouvoirs d'un utilisateur standard. Les privilèges de superutilisateur correspondent à l'ensemble des droits octroyés au superutilisateur. Un utilisateur privilégié ou une application privilégiée est un utilisateur ou une application qui a reçu des droits supplémentaires.

2. Droit discret dans le cadre d'un processus dans un système Oracle Solaris. Les privilèges offrent un contrôle des processus plus détaillé que `root`. Les privilèges sont définis et appliqués dans le noyau. Les privilèges sont également appelés *privilèges de processus* ou *privilèges de noyau*. Pour une description complète des privilèges, reportez-vous à la page de manuel `privileges`(5).

profil de droits Egalement appelé profil. Collection de remplacements de sécurité pouvant être affectés à un rôle ou à un utilisateur. Un profil de droits peut inclure des autorisations, des privilèges, des commandes avec des attributs de sécurité et d'autres profils de droits appelés profils supplémentaires.

profil de droits authentifié profil de droits qui exige que l'utilisateur ou le rôle affecté saisisse un mot de passe avant d'exécuter une opération à partir du profil. Ce comportement est similaire au comportement `sudo`. La durée pendant laquelle le mot de passe est valide est configurable.

protocole Diffie-Hellman Egalement appelé cryptographie par clé publique. Protocole d'accord de clés cryptographiques asymétriques mis au point par Diffie et Hellman en 1976. Ce protocole permet à deux utilisateurs d'échanger une clé secrète via un moyen non sécurisé sans secrets préalables. Diffie-Hellman est utilisé par Kerberos.

QOP Qualité de la protection. Paramètre servant à sélectionner les algorithmes cryptographiques utilisés en association avec le service d'intégrité ou de confidentialité.

RBAC Contrôle d'accès basé sur les rôles, la fonction de gestion des droits des utilisateurs d'Oracle Solaris. Reportez-vous à droits.

réauthentification Exigence de fournir un mot de passe pour effectuer une opération informatique. En règle générale, les opérations `sudo` nécessitent une réauthentification. Les profils de droits authentifiés peuvent contenir des commandes qui nécessitent une réauthentification. Reportez-vous à profil de droits authentifié.

relation Variable ou relation de configuration définie dans les fichiers `kdc.conf` ou `krb5.conf`.

rôle Identité spéciale destinée à l'exécution d'applications privilégiées et ne pouvant être prise que par des utilisateurs assignés.

RSA Méthode permettant d'obtenir des signatures numériques et des systèmes de cryptographie par clé publique. Cette méthode datant de 1978 a été décrite par trois développeurs (Rivest, Shamir et Adleman).

SEAM Nom du produit pour la version initiale de Kerberos sur les systèmes Solaris. Ce produit est basé sur la technologie Kerberos V5 développée au Massachusetts Institute of Technology. SEAM est maintenant appelé "service Kerberos". Il continue d'être légèrement différent de la version MIT.

Secure Shell Un protocole particulier pour une connexion à distance sécurisée et d'autres services de réseau sécurisé via un réseau non sécurisé.

sécurisation Modification de la configuration par défaut du système d'exploitation pour supprimer les failles de sécurité inhérentes à l'hôte.

séparation des tâches Composante de la notion de moindre privilège. La séparation des tâches permet d'empêcher un utilisateur d'exécuter ou d'approuver les opérations terminant une transaction. Par exemple, dans RBAC, vous pouvez séparer la création d'un utilisateur de connexion de l'affectation des remplacements de sécurité. Un rôle crée l'utilisateur. Un autre rôle peut affecter les attributs de sécurité, tels que des profils de droits, des rôles et des privilèges aux utilisateurs existants.

serveur Principal fournissant une ressource aux clients réseau. Si vous vous connectez par `ssh` au système `central.example.com` par exemple, ce système est le serveur fournissant le service `ssh`. Voir également principal de service.

serveur d'application Voir serveur d'application réseau.

serveur d'application réseau Serveur fournissant une application réseau, telle que `ftp`. Un domaine peut contenir plusieurs serveurs d'application réseau.

service 1. Ressource fournie aux clients du réseau, souvent par plusieurs serveurs. Si vous vous connectez par `rlogin` à la machine `central.example.com` par exemple, cette machine est le serveur fournissant le service `rlogin`.

2. Service de sécurité (d'intégrité ou de confidentialité) fournissant un niveau de protection supérieur à l'authentification. Voir également intégrité et confidentialité.

service de sécurité Voir service.

SHA1 Algorithme de hachage sécurisé (Secure Hashing Algorithm). L'algorithme s'applique à toute longueur d'entrée inférieure à 2^{64} afin d'obtenir une synthèse des messages. L'algorithme SHA1 sert d'entrée à DSA.

shell de profil Dans la gestion des droits, shell permettant à un rôle (ou un utilisateur) d'exécuter à partir de la ligne de commande toutes les applications privilégiées affectées aux profils de droits du rôle. Les versions du shell de profil correspondent aux shells disponibles sur le système, tels que la version `pfbash` de `bash`.

stratégie En règle générale, plan ou ensemble d'actions qui influence ou détermine les décisions et actions. Pour les systèmes informatiques, la stratégie fait généralement référence à la stratégie de sécurité. La stratégie de sécurité de votre site constitue un ensemble de règles qui définissent la sensibilité des informations traitées et les mesures prises pour protéger les informations contre tout accès non autorisé. Par exemple, la stratégie de sécurité peut exiger que les systèmes soient audités, que les périphériques soient alloués pour être utilisés et que les mots de passe soient modifiés toutes les six semaines.

Pour l'implémentation d'une stratégie dans des zones spécifiques du SE Oracle Solaris, voir stratégie d'audit, stratégie dans la structure cryptographique, stratégie de périphériques, stratégie Kerberos, stratégie de mot de passe et stratégie de droits.

stratégie d'audit Paramètres globaux et par utilisateur qui déterminent les événements d'audit enregistrés. Les paramètres globaux s'appliquant au service d'audit déterminent généralement les informations facultatives à inclure dans la piste d'audit. Deux paramètres, `cnt` et `ahlt`, affectent le fonctionnement du système lorsque la file d'attente de l'audit est pleine. Par exemple, la stratégie d'audit peut exiger qu'un numéro de séquence fasse partie de chaque enregistrement d'audit.

stratégie dans la structure cryptographique Dans la fonctionnalité Structure cryptographique d'Oracle Solaris, la stratégie correspond à la désactivation de mécanismes de chiffrement existants. Les mécanismes ne peuvent ensuite plus être utilisés. La stratégie de la structure cryptographique peut empêcher l'utilisation d'un mécanisme particulier tel que `CKM_DES_CBC` provenant d'un fournisseur tel que DES.

stratégie de droits Stratégie de sécurité associée à une commande. Actuellement, `solaris` est la stratégie valide pour Oracle Solaris. La stratégie `solaris` reconnaît les privilèges et la stratégie de privilèges étendue, les autorisations et les attributs de sécurité `setuid`.

stratégie de mot de passe Algorithmes de chiffrement pouvant être utilisés pour générer des mots de passe. Peut également faire référence à des questions plus générales concernant les mots de passe, telles que la fréquence à laquelle le mot de passe doit être modifié, le nombre de tentatives autorisées de saisie d'un mot de passe et d'autres considérations relatives à la sécurité. La stratégie de sécurité requiert des mots de passe. La stratégie de mot de passe peut requérir des mots de passe chiffrés avec l'algorithme AES, et imposer d'autres exigences relatives à la force des mots de passe.

stratégie de périphériques Protection d'un périphérique au niveau du noyau. La stratégie de périphériques repose sur deux jeux de privilèges pour un périphérique. Un jeu de privilèges contrôle l'accès en lecture au périphérique. Le deuxième jeu de privilèges contrôle l'accès en écriture sur le périphérique. Voir également stratégie.

stratégie de sécurité Voir stratégie.

stratégie Kerberos Ensemble de règles régissant l'utilisation des mots de passe dans le service Kerberos. Les stratégies peuvent réguler les accès des principaux ou des paramètres de ticket, tels que la durée de vie.

stratégie pour technologies à clé publique Dans la structure de gestion des clés (KMF), la stratégie correspond à la gestion de l'utilisation des certificats. La base de données de stratégies KMF peut définir des contraintes s'appliquant à l'utilisation des clés et des certificats gérés par la bibliothèque KMF.

stratégie RBAC Reportez-vous à stratégie de droits.

stratégies de réseau Paramètres configurés par les utilitaires réseau pour protéger le trafic réseau. Pour plus d'informations sur la sécurité réseau, reportez-vous à la section " Sécurisation du réseau dans Oracle Solaris 11.2 ".

synthèse Voir synthèse de message.

synthèse de message Valeur de hachage calculée à partir d'un message. La valeur de hachage identifie le message de manière presque unique. Une synthèse permet de vérifier l'intégrité d'un fichier.

TGS Service d'octroi de tickets (Ticket-Granting Service) Partie du KDC responsable de l'émission des tickets.

TGT Ticket d'octroi de tickets (Ticket-Granting Ticket) Ticket émis par le KDC, permettant au client de demander des tickets pour d'autres services.

ticket Paquet d'informations servant à transmettre en toute sécurité l'identité d'un utilisateur à un serveur ou un service. Un ticket n'est valable que pour un client et un service particulier sur un serveur spécifique. Il contient le nom de principal du service, le nom de principal de l'utilisateur, l'adresse IP de l'hôte de l'utilisateur, un horodatage et une valeur définissant la durée de vie du ticket. La création d'un ticket s'effectue à l'aide d'une clé de session aléatoire utilisée par le client et le service. Une fois le ticket créé, il peut être réutilisé jusqu'à son expiration. Un ticket sert uniquement à authentifier un client lorsqu'il est présenté avec un nouvel authentificateur. Voir également authentificateur, informations d'identification, service, clé de session.

ticket initial Ticket émis de manière directe, et pas à partir d'un ticket d'octroi de tickets. Certains services, tels que les applications modifiant les mots de passe, peuvent nécessiter des tickets marqués comme étant `initiaux` afin d'assurer que le client peut démontrer qu'il connait sa clé secrète. Cette garantie est importante car un ticket initial indique que le client s'est récemment authentifié et ne dépend pas d'un ticket d'octroi de tickets qui peut exister depuis un certain temps.

ticket non valide Ticket postdaté n'étant pas encore utilisable. Un ticket non valide est rejeté par un serveur d'application jusqu'à ce qu'il soit validé. Pour être validé, un ticket non valide doit être présenté au KDC par le client dans une demande de TGS, avec l'indicateur `VALIDATE`, après l'heure de début. Voir également ticket postdaté.

ticket postdaté Un ticket postdaté ne devient valide qu'un certain temps après sa création. Par exemple, un tel ticket peut être utile avec les tâches exécutées par lots la nuit car le ticket, s'il est volé, ne peut pas être utilisé tant que l'exécution de ces tâches n'a pas eu lieu. Lorsqu'un ticket postdaté est émis, il est émis en tant que `non valide` et le reste jusqu'à ce que a) son heure de début soit dépassée et b) le client demande la validation par le KDC. Un ticket postdaté est normalement valide jusqu'à l'heure d'expiration du ticket d'octroi de tickets. Toutefois, si le ticket postdaté est marqué comme `renouvelable`, sa durée de vie est normalement égale à la durée de vie entière du ticket d'octroi de tickets. Voir également ticket non valide, ticket renouvelable.

ticket renouvelable Etant donné que tout ticket dont la durée de vie est très longue peut présenter un risque pour la sécurité, les tickets peuvent être conçus pour être `renouvelables`. Un ticket renouvelable possède deux moments d'expiration : a) l'heure à laquelle l'instance courante du ticket expire et b) la durée de vie maximale de tout ticket. Si un client souhaite continuer à utiliser un ticket, il peut le renouveler avant sa première expiration. Par exemple, un ticket peut être valide pendant une heure, et tous les tickets ont une durée de vie maximale de dix heures. Si le client détenant le ticket souhaite le conserver plus d'une heure, il doit le renouveler. Lorsqu'un ticket atteint sa durée de vie maximale, celui-ci expire automatiquement et ne peut pas être renouvelé.

ticket transmissible Ticket pouvant être utilisé par un client pour demander un ticket sur un hôte distant sans devoir se soumettre au processus complet d'authentification sur l'hôte concerné. Par exemple, si l'utilisateur `david` obtient un ticket transmissible sur la machine de l'utilisateur `jennifer`, `david` peut se connecter à sa propre machine sans avoir à demander un nouveau ticket (et donc à s'authentifier de nouveau). Voir également ticket utilisable avec proxy.

ticket utilisable avec proxy Ticket pouvant être utilisé par un service pour le compte d'un client afin d'effectuer une opération pour ce dernier. On dit alors que le service agit en tant que proxy du client. Grâce à ce ticket, le service peut prendre l'identité du client. Le service peut utiliser un tel ticket afin d'obtenir un ticket de service d'un autre service, mais non un ticket d'octroi de tickets. La différence entre un ticket utilisable avec proxy et un ticket transmissible réside dans le fait que le premier n'est valide que pour une seule opération. Voir également ticket transmissible.

utilisateur privilégié Utilisateur disposant de droits dépassant les droits d'utilisateur standard sur un système informatique. Reportez-vous également aux utilisateurs de confiance.

utilisateurs de confiance Utilisateurs qui selon votre volonté peuvent effectuer des tâches d'administration à un certain niveau de confiance. En règle générale, les administrateurs créent des connexions pour les utilisateurs de confiance et affectent des droits d'administration qui correspondent au niveau de confiance des utilisateurs et à leur capacité. Ces utilisateurs aident ensuite à configurer et à entretenir le système. Egalement appelés *utilisateurs privilégiés*.

variante Historiquement, la *variante de sécurité* et la *variante d'authentification* désignaient la même chose, lorsqu'une variante indiquait un type d'authentification (AUTH_UNIX, AUTH_DES, AUTH_KERB). RPCSEC_GSS est également une variante de sécurité, même s'il permet d'assurer des services d'intégrité et de confidentialité, en plus de l'authentification.

variante de sécurité Voir variante.

VPN Réseau privé virtuel assurant une communication sécurisée en utilisant les mécanismes de chiffrement et de mise en tunnel pour connecter les utilisateurs via un réseau public.

Index

D

E

F

G

H

I

J

K

R

S