Groupe de travail Réseau
Request for Comments : 4108
Catégorie : Sur la voie de la normalisation

R. Housley
Vigil Security
août 2005
Traduction Claude Brière de L'Isle

Utilisation de la syntaxe de message cryptographique (CMS) pour protéger les paquetages de microcode

Statut de ce mémoire

Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé

Le présent document décrit l'utilisation de la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*) pour protéger les paquetages de microcode *(firmware package)*, qui fournissent le code d'objet pour un ou plusieurs composants de module de matériel. La CMS est spécifiée dans la RFC 3852. Une signature numérique est utilisée pour protéger le paquetage de microcode contre la modification non détectée et fournir l'authentification de l'origine des données. Le chiffrement est utilisé facultativement pour protéger le paquetage de microcode de la divulgation, et la compression est utilisée facultativement pour réduire la taille du paquetage de microcode protégé. Un récépissé de chargement de paquetage de microcode peut être facultativement généré pour notifier la réussite du chargement d'un paquetage de microcode. De façon similaire, un rapport d'erreur de chargement de paquetage de microcode peut être facultativement généré pour porter un échec de chargement du paquetage de microcode.

Table des Matières

1. Introduction

Le présent document décrit l'utilisation de la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*) [RFC3852] pour protéger les paquetages de microcode. Le présent document décrit aussi l'utilisation de la CMS pour les récépissés et les rapports d'erreurs pour le chargement de paquetage de microcode. La CMS est une syntaxe d'encapsulation de protection qui utilise l'ASN.1 [X.208-88], [X.209-88]. Le paquetage de microcode protégé peut être associé à tout module de matériel ; cependant, la présente spécification a été écrite en considérant les exigences de modules de matériels cryptographiques, car ces modules ont de fortes exigences de sécurité.

Le paquetage de microcode contient du code d'objet pour un ou plusieurs composants programmables qui constituent le module de matériel. Le paquetage de microcode, qui est traité comme un objet binaire opaque, est signé numériquement. Le chiffrement et la compression facultatifs sont aussi pris en charge. Lorsque les trois sont utilisés, le paquetage de microcode est compressé, puis chiffré, et ensuite signé. La compression réduit simplement la taille du paquetage de microcode, permettant un traitement et une transmission plus efficaces. Le chiffrement protège le paquetage de microcode de la divulgation, ce qui permet la transmission de paquetages de microcode sensibles sur des liaisons non sûres. L'algorithme de chiffrement et le mode employés peuvent aussi assurer l'intégrité, protégeant le paquetage de microcode contre la modification non détectée. Le chiffrement protège les algorithmes propriétaires, les algorithmes classifiés, les secrets d'affaire, et les techniques de mise en œuvre. La signature numérique protège le paquetage de microcode contre la modification non détectée et assure l'authentification de l'origine des données. La signature numérique permet au module de matériel de confirmer que le paquetage de microcode vient d'une source acceptable.

Si le chiffrement est utilisé, la clé de déchiffrement de microcode doit être rendue disponible au module de matériel via un chemin sûr. La clé peut être livrée via un support physique ou via un chemin électronique indépendant. Un mécanisme facultatif pour distribuer la clé de déchiffrement de microcode est spécifié au paragraphe 2.3.1, mais tout mécanisme sûr de distribution de clé est acceptable. La clé publique de vérification de signature doit être rendue disponible au module de matériel d'une manière qui préserve son intégrité et confirme sa source. La CMS prend en charge le transfert des certificats, et cette facilité peut être utilisée pour transférer un certificat qui contient la clé publique de vérification de signature (un certificat de signature de microcode). Cependant, l'utilisation de cette facilité introduit une possibilité de tromperie. Finalement, une clé publique d'ancre de confiance peut être rendue disponible pour le module de matériel. Le paragraphe 1.2 établit l'exigence que le module de matériel mémorise une ou plusieurs ancres de confiance.

Les modules de matériel peuvent n'être pas capables d'accéder aux répertoires de certificats ou à des serveurs de découverte de chemin délégué (DPD, *delegated path discovery*) [RFC3379] pour acquérir les certificats nécessaires pour achever un chemin de certification. Donc, il est de la responsabilité du signataire du paquetage de microcode d'inclure des certificats suffisants pour permettre à chaque module de valider le certificat de signataire de microcode (voir le paragraphe 2.1.2). De même, les modules de matériel peuvent n'être pas capables d'accéder à un répertoire de listes de révocation de certificat (CRL, *certificate revocation list*) à un répondeur OCSP [RFC2560], ou à un serveur de validation de chemin délégué (DPV, *delegated path validation*) [RFC3379] pour acquérir les informations d'état de révocation. Donc, si la signature de paquetage de microcode ne peut pas être validée seulement avec la clé publique de l'ancre de confiance et si le module de matériel n'est pas capable d'effectuer la validation complète du chemin de certification, il est alors de la responsabilité de l'entité qui charge un paquetage dans un module de matériel de valider le chemin de certification du signataire de microcode avant de charger le paquetage dans un module de matériel. Les moyens par lesquels cette vérification externe de l'état de révocation de certificat est effectuée sortent du domaine d'application de la présente spécification.

Les modules de matériel vont seulement accepter des paquetages de microcode avec une signature numérique valide. La signature est soit validée directement en utilisant la clé publique d'ancre de confiance, soit en utilisant un chemin de certification du signataire du microcode qui est validé pour la clé publique d'ancre de confiance. Donc, les ancres de confiance définissent l'ensemble des entités qui peuvent créer des paquetages de microcode pour le module de matériel.

La disposition d'un paquetage de microcode précédemment chargé après la réussite de la validation d'un autre paquetage de microcode sort du domaine d'application de la présente spécification. La quantité de mémoire disponible chez le module de matériel va déterminer la gamme des solutions de remplacement.

Dans certains cas, les modules de matériel peuvent générer des récépissés pour reconnaître le chargement d'un paquetage de microcode particulier. De tels récépissés peuvent être utilisés pour déterminer quels modules de matériel ont besoin de recevoir une mise à jour du paquetage de microcode chaque fois qu'est découverte une faute dans un paquetage de microcode antérieur. Les modules de matériel peuvent aussi générer des rapports d'erreur pour indiquer l'échec du chargement d'un paquetage de microcode. Pour mettre en œuvre la génération de récépissé ou de rapport d'erreur, le module de matériel est obligé d'avoir un numéro de série unique permanent. Les récépissés et les rapports d'erreur peuvent être signés ou non signés. Pour générer des récépissés ou des rapports d'erreurs signés numériquement, un module de matériel DOIT produire sa propre clé de signature privée et un certificat qui contient la clé publique de validation de signature correspondante. Afin d'économiser la mémoire du module de matériel, le module de matériel peut mémoriser un désignateur de certificat au lieu du certificat lui-même. La clé de signature privée exige une mémorisation sûre.

1.1 Terminologie

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans la [RFC2119] et indiquent les niveaux d'exigence pour les mises en œuvre conformes.

1.2 Éléments d'architecture

L'architecture inclut le module de matériel, le paquetage de microcode, et un chargeur d'amorçage. Le chargeur d'amorçage DOIT avoir accès à une ou plusieurs clés publiques de confiance, appelées ancres de confiance, pour valider la signature sur le paquetage de microcode. Si un récépissé ou rapport d'erreur sur le chargement d'un paquetage de microcode signé est créé au nom du module de matériel, le chargeur d'amorçage DOIT alors avoir accès à une clé de signature privée pour générer la signature et l'identifiant de signataire pour le certificat de validation de signature correspondant ou son désignataire. Un certificat de validation de signature PEUT être inclus pour aider à la validation de signature. Pour mettre en œuvre cette capacité facultative, le module de matériel DOIT avoir un numéro de série unique et une clé de signature privée ; le module de matériel PEUT aussi inclure un certificat qui contient la clé publique de validation de signature correspondante. Ces éléments DOIVENT être installés dans le module de matériel avant qu'il soit déployé. La clé privée et le certificat peuvent être générés et installés au titre du processus de fabrication du module de matériel. La Figure 1 illustre ces éléments d'architecture.

Les identifiants d'objet ASN.1 sont le moyen préféré de désignation des éléments d'architecture. Les détails de la gestion des ancres de confiance sortent du domaine d'application de la présente spécification. Cependant, une ou plusieurs ancres de confiance DOIVENT être installées dans le module de matériel en utilisant un processus sûr avant qu'il soit déployé. Ces ancres de confiance fournissent un moyen de contrôler les sources acceptables de paquetages de microcode. Le vendeur de module de matériel peut inclure des dispositions pour une gestion à distance sûre des ancres de confiance. Une approche est d'inclure les ancres de confiance dans les paquetages de microcode eux-mêmes. Cette approche est analogue à la capacité facultative décrite plus loin pour la mise à jour du chargeur d'amorçage. Dans un module de matériel cryptographique, le paquetage de microcode peut mettre en œuvre de nombreux algorithmes de chiffrement différents. Lorsque le paquetage de microcode est chiffré, la clé de déchiffrement de microcode et le paquetage de microcode DOIVENT tous deux être fournis au module de matériel. La clé de déchiffrement de microcode est nécessaire pour utiliser le paquetage de microcode associé. Généralement, des mécanismes de distribution séparés seront employés pour la clé de déchiffrement de microcode et le paquetage de microcode. Un mécanisme facultatif pour la distribution sûre de la clé de déchiffrement de microcode avec le paquetage de microcode est spécifié au paragraphe 2.3.1.

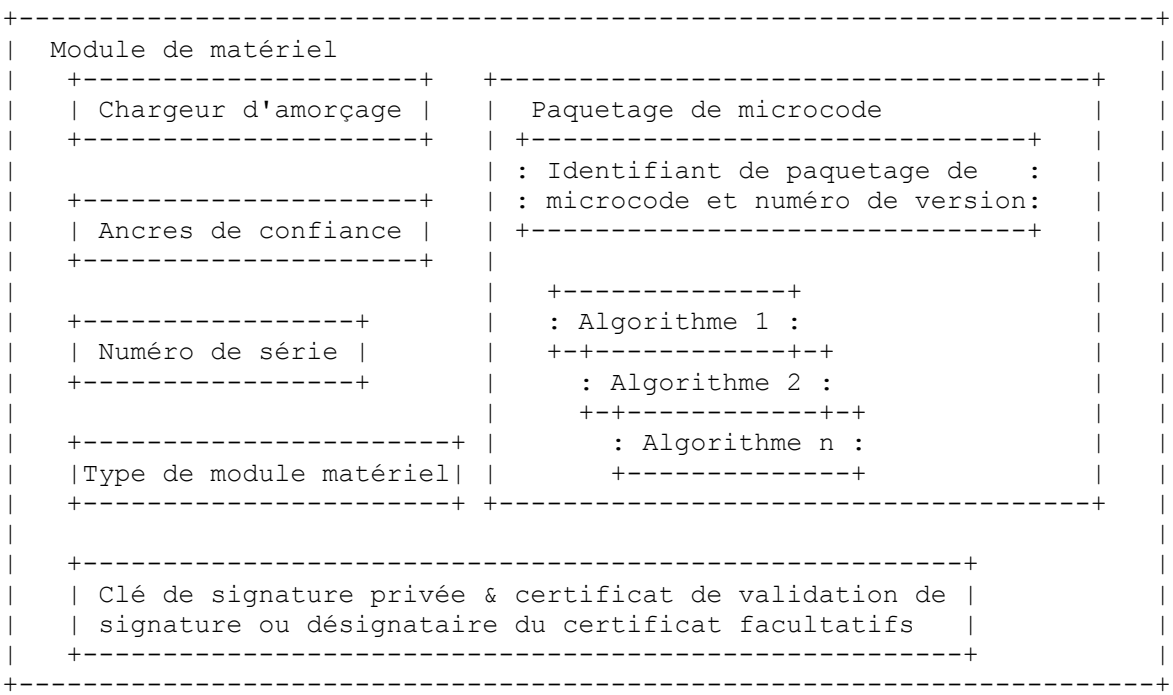

Figure 1. Éléments d'architecture

1.2.1 Exigences de module de matériel

De nombreux fabricants différents développent des modules de matériel, et chaque fabricant identifie normalement ses modules par type (famille) de produit et niveau de révision. Un identifiant d'objet univoque DOIT désigner chaque type et révision de module de matériel.

Chaque module de matériel au sein d'une famille de modules de matériel DEVRAIT avoir un numéro de série unique et permanent. Cependant, si la capacité facultative de génération de récépissé ou de rapport d'erreur est mise en œuvre, le module de matériel DOIT alors avoir un numéro de série unique et permanent. Si la capacité facultative de signature de récépissé ou de rapport d'erreur est mise en œuvre, le module de matériel DOIT alors avoir une clé de signature privée et un certificat contenant la clé publique de validation de signature correspondante ou son désignataire. Si un numéro de série est présent, le chargeur d'amorçage l'utilise pour les décisions d'autorisation (voir au paragraphe 2.2.8) pour la génération des récépissés (voir la Section 3) et la génération de rapport d'erreur (voir la Section 4).

Lorsque un module de matériel inclut plus d'un composant programmable par microcode, le chargeur d'amorçage distribue les composants du paquetage aux composants appropriés au sein du module de matériel après la validation du paquetage de microcode. Le chargeur d'amorçage est exposé plus en détails au paragraphe 1.2.3.

1.2.2. Exigences de paquetage de microcode

Deux approches de désignation des paquetages de microcode sont prises en charge : l'héritage et la préférence. Les noms de paquetage de microcode sont placés dans un attribut de CMS signé, et non dans le paquetage de microcode lui-même.

L'héritage de noms de paquetage de microcode est une simple chaîne d'octets, et aucune structure n'est obligée. Cette forme de nom de paquetage de microcode est prise en charge afin de faciliter les systèmes existants de gestion de configuration. On suppose que le signataire du microcode et le chargeur d'amorçage vont comprendre toute la structure interne de la chaîne d'octets. En particulier, étant donnés deux noms hérités de paquetage de microcode, on suppose que le signataire du microcode et le chargeur d'amorçage seront capables de déterminer lequel représente la version la plus récente du paquetage de microcode. Cette capacité est nécessaire pour mettre en œuvre la caractéristique de version périmée. Si un paquetage de microcode avec une faille désastreuse est publiée, les versions suivantes du paquetage de microcode PEUVENT désigner un nom hérité de paquetage de microcode périmé afin d'empêcher des replis ultérieurs sur la version périmée ou sur des versions antérieures à la version périmée.

Les noms de paquetage de microcode préférés sont une combinaison de l'identifiant d'objet du paquetage de microcode et d'un numéro de version. Un identifiant d'objet univoque DOIT identifier la collection de caractéristiques qui caractérise le paquetage de microcode. Par exemple, des paquetages de microcode pour un modem câble et une carte d'interface de réseau LAN sans fil portent des identifiants d'objet distincts. De même, des paquetages de microcode qui mettent en œuvre des suites d'algorithmes de chiffrement et des modes de fonctionnement distincts, ou qui émulent des appareils de chiffrement différents (non programmables) portent des identifiants d'objet distincts. Le numéro de version DOIT identifier une construction ou livraison particulière du paquetage de microcode. Le numéro de version DOIT être un entier non négatif à accroissement monotone. Généralement, une version antérieure est remplacée par une version ultérieure. Si un paquetage de microcode avec une faille désastreuse est publié, les versions suivantes du paquetage de microcode PEUVENT désigner un numéro de version périmée pour empêcher un repli ultérieur à la version périmée ou à des versions antérieures à la version périmée.

Les paquetages de microcode sont développés pour fonctionner sur un ou plusieurs types de module de matériel. La signature numérique du paquetage de microcode DOIT lier la liste des identifiants d'objet de module de matériel pris en charge au paquetage de microcode.

Dans de nombreux cas, la signature du paquetage de microcode sera validée directement avec la clé publique d'ancre de confiance, évitant d'avoir besoin de construire des chemins de certification. Autrement, l'ancre de confiance peut déléguer la signature du paquetage de microcode à une autre clé publique à travers un chemin de certification. Dans ce dernier cas, le paquetage de microcode DEVRAIT contenir les certificats nécessaires pour construire le chemin de certification qui commence par un certificat produit par les ancres de confiance et se termine par un certificat produit au signataire du paquetage de microcode.

Le paquetage de microcode PEUT contenir une liste d'identifiants de communauté. Ces identifiants désignent les modules de matériel qui sont autorisés à charger le paquetage de microcode. Si le paquetage de microcode contient une liste d'identifiants de communauté, alors le chargeur d'amorçage DOIT rejeter le paquetage de microcode si le module de matériel n'est pas un membre d'une des communautés identifiées.

Lorsque un module de matériel inclut plusieurs composants programmables, le paquetage de microcode DEVRAIT contenir du code exécutable pour tous les composants. L'étiquetage interne au sein du paquetage de microcode DOIT dire

au chargeur d'amorçage quelle portion du paquetage de microcode global est destinée à chaque composant ; cependant, cet étiquetage est supposé être spécifique de chaque module de matériel. Comme la présente spécification traite le paquetage de microcode comme un objet binaire opaque, le format du paquetage de microcode sort du domaine d'application de la présente spécification.

1.2.3 Exigences pour le chargeur d'amorçage

Le chargeur d'amorçage DOIT avoir accès à une interface physique et à tout pilote ou logiciel de protocole en rapport nécessaire pour obtenir un paquetage de microcode. La même interface DEVRAIT être utilisée pour délivrer les récépissés et les rapports d'erreur. Les détails sur l'interface physique ainsi que sur le pilote ou logiciel de protocole sortent du domaine d'application de la présente spécification.

Le chargeur d'amorçage peut être une partie permanente du module de matériel, ou il peut être remplacé par le chargement d'un paquetage de microcode. Dans la Figure 1, le chargeur d'amorçage est mis en œuvre comme une logique séparée au sein du module de matériel. Tous les modules de matériel ne vont pas inclure la capacité de remplacer ou mettre à jour le chargeur d'amorçage, et la présente spécification ne rend pas obligatoire une telle prise en charge.

Si le chargeur d'amorçage peut être chargé par un paquetage de microcode, un chargeur d'amorçage initial DOIT être installé dans une mémoire non volatile avant le déploiement. Tous les chargeurs d'amorçage, y compris un chargeur d'amorçage initial si il en est employé un, DOIVENT satisfaire aux exigences de cette section. Cependant, le paquetage de microcode qui contient le chargeur d'amorçage PEUT aussi contenir d'autres programmes.

Le chargeur d'amorçage exige l'accès aux programmes de chiffrement. Ces programmes peuvent être spécifiquement mis en œuvre pour le chargeur d'amorçage, ou ils peuvent être partagés avec d'autres caractéristiques de module de matériel. Le chargeur d'amorçage DOIT avoir accès à une fonction de hachage unidirectionnel et à des programmes de vérification de signature numérique pour valider la signature numérique sur le paquetage de microcode et pour valider le chemin de certification pour le certificat de signature de microcode.

Si les paquetages de microcode sont chiffrés, le chargeur d'amorçage DOIT avoir accès à un programme de déchiffrement. L'accès à la fonction de chiffrement correspondante n'est pas exigé, car les modules de matériel n'ont pas besoin d'être capables de générer des paquetages de microcode. Parce que certaines mises en œuvre d'algorithme de chiffrement symétrique (comme [AES]) emploient des logiques séparées pour le chiffrement et le déchiffrement, certaines économies de module de matériel peuvent en résulter.

Si les paquetages de microcode sont compressés, le chargeur d'amorçage DOIT aussi avoir accès à une fonction de décompression. Cette fonction peut être mise en œuvre spécifiquement pour le chargeur d'amorçage, ou elle peut être partagée avec d'autres caractéristiques du module de matériel. L'accès à la fonction de compression correspondante n'est pas exigé, car les modules de matériel n'ont pas besoin d'être capables de générer des paquetages de microcode.

Si la capacité facultative de génération de récépissé ou de rapport d'erreur est prise en charge, le chargeur d'amorçage DOIT avoir accès au numéro de série du module de matériel et à l'identifiant d'objet pour le type de module de matériel. Si la capacité facultative de génération de récépissé ou de rapport d'erreur signé est prise en charge, le chargeur d'amorçage DOIT aussi avoir accès à une fonction de hachage unidirectionnel et à des sous programmes de signature numérique, à la clé de signature privée du module de matériel, et au certificat de validation de signature correspondant ou son désignataire.

Le chargeur d'amorçage requiert l'accès à une ou plusieurs clés publiques de confiance, appelées des ancres de confiance, pour valider la signature numérique du paquetage de microcode. Une ou plusieurs ancres de confiance DOIVENT être installées dans une mémoire non volatile avant le déploiement. Le chargeur d'amorçage DOIT rejeter un paquetage de microcode si il ne peut pas valider la signature, ce qui PEUT exiger la construction d'un chemin de certification valide du certificat de signature du microcode à une des ancres de confiance [RFC3280]. Cependant, dans de nombreux cas, la signature du paquetage de microcode sera validée directement avec la clé publique d'ancre de confiance, évitant d'avoir besoin de construire les chemins de certification.

Le chargeur d'amorçage DOIT rejeter un paquetage de microcode si la liste des identifiants de type de module de matériel pris en charge au sein du paquetage de microcode ne comporte pas l'identifiant d'objet du module de matériel.

Le chargeur d'amorçage DOIT rejeter un paquetage de microcode si il inclut une liste d'identifiants de communauté et si le module de matériel n'est pas membre d'une des communautés de la liste. Les moyens pour déterminer l'appartenance à une communauté sortent du domaine d'application de la présente spécification.

Le chargeur d'amorçage DOIT rejeter un paquetage de microcode si il ne peut pas réussir à déchiffrer le paquetage de microcode en utilisant la clé de déchiffrement de microcode disponible au module de matériel. Le paquetage de microcode

contient un identifiant de la clé de déchiffrement de microcode nécessaire pour le déchiffrement.

Lorsque une version antérieure d'un paquetage de microcode en remplace une plus récente, le chargeur d'amorçage DEVRAIT générer un avertissement. La manière dont un tel avertissement est généré dépend beaucoup du module de matériel et de l'environnement de son utilisation. Si un paquetage de microcode avec une faille désastreuse est publié et que des versions ultérieures du paquetage de microcode désignent une version périmée, le chargeur d'amorçage DEVRAIT empêcher le chargement de la version périmée et des versions antérieures à la version périmée.

1.2.3.1 Traitement de version périmée héritée

En cas de publication d'un paquetage de microcode avec une faille désastreuse, les versions suivantes du paquetage de microcode qui emploient la forme de nom hérité du paquetage de microcode PEUVENT inclure un nom hérité de paquetage de microcode périmé pour empêcher des replis ultérieurs à la version périmée ou à des versions antérieures à la version périmée. Comme décrit à la Section "Considérations sur la sécurité" du présent document, l'inclusion d'un nom de paquetage de microcode hérité périmé dans un paquetage de microcode ne peut pas complètement empêcher l'utilisation ultérieure du paquetage de microcode périmé. Cependant, de nombreux modules de matériel sont supposés avoir très peu de paquetages de microcode écrits pour eux, permettant que la caractéristique de version périmée de paquetage de microcode fournisse des protections importantes.

Une mémorisation non volatile des numéros de version périmée est nécessaire. Le nombre de noms hérités de paquetage de microcode périmé qui peuvent être mémorisés dépend de la quantité de mémoire qui est disponible. Lorsque un paquetage de microcode est chargé et qu'il contient un nom hérité de paquetage de microcode périmé, il DEVRAIT alors être ajouté à une liste conservée dans une mémorisation non volatile. Lorsque des paquetages de microcode sont chargés ultérieurement, le nom hérité de paquetage de microcode du nouveau paquetage est comparé à la liste dans la mémoire non volatile. Si le nom hérité de paquetage de microcode représente la même version qu'une version plus ancienne d'un membre de la liste, le nouveau paquetage de microcode DEVRAIT alors être rejeté.

La quantité de mémoire non volatile qu'il est nécessaire de dédier à la sauvegarde des noms hérités de paquetage de microcode et des noms hérités de paquetages de microcode périmés dépend du nombre de paquetages de microcode qui vont vraisemblablement être développés pour le module de matériel.

1.2.3.2 Traitement de la version périmée préférée

Si un paquetage de microcode est publié avec une faille désastreuse, les versions suivantes du paquetage de microcode qui emploient la forme de nom préférée du paquetage de microcode PEUVENT inclure un numéro de version périmée pour empêcher le repli ultérieur à la version périmée ou à des versions antérieures à la version périmée. Comme décrit dans la Section "Considérations sur la sécurité" du présent document, l'inclusion d'un numéro de version périmée dans un paquetage de microcode ne peut pas empêcher complètement l'utilisation ultérieure du paquetage de microcode périmé. Cependant, de nombreux modules de matériel sont supposés avoir très peu de paquetages de microcode écrits pour eux, ce qui permet que la caractéristique de version périmée de paquetage de microcode fournisse une protection importante.

Une mémorisation non volatile est nécessaire pour les numéros de version périmée. Le numéro de version périmée qui peut être mémorisé dépend de la quantité de mémoire disponible. Lorsque un paquetage de microcode chargé contient un numéro de version périmée, l'identifiant d'objet du paquetage de microcode et le numéro de version périmée DEVRAIENT alors être ajoutés à une liste conservée dans une mémoire non volatile. Lorsque sont chargés ultérieurement des paquetages de microcode, l'identifiant d'objet et le numéro de version du nouveau paquetage sont comparés à la liste de la mémoire non volatile. Si l'identifiant d'objet correspond et si le numéro de version est inférieur ou égal au numéro de version périmée, le nouveau paquetage de microcode DEVRAIT alors être rejeté.

La quantité de mémoire non volatile qu'il est nécessaire de dédier à la sauvegarde des identifiants et numéros de version périmée de paquetage de microcode dépend du nombre de paquetages de microcode qui seront vraisemblablement développés pour le module de matériel.

1.2.4 Ancres de confiance

Une ancre de confiance DOIT consister en un algorithme de signature de clé publique et une clé publique associée, qui PEUT facultativement inclure des paramètres. Une ancre de confiance DOIT aussi inclure un identifiant de clé publique. Une ancre de confiance PEUT aussi inclure un nom distinctif X.500.

La clé publique d'ancre de confiance est utilisée en conjonction avec l'algorithme de validation de signature de deux façons différentes. D'abord, la clé publique d'ancre de confiance est utilisée directement pour valider la signature de paquetage de microcode. Ensuite, la clé publique d'ancre de confiance est utilisée pour valider un chemin de certification X.509, et ensuite la clé publique sujette dans le certificat final du chemin de certification est utilisée pour valider la signature de

paquetage de microcode.

La clé publique désigne l'ancre de confiance, et chaque clé publique a un identifiant de clé publique. L'identifiant de clé publique identifie l'ancre de confiance comme signataire quand elle est utilisée directement pour valider les signatures de paquetage de microcode. Cet identifiant de clé peut être mémorisé avec l'ancre de confiance, ou il peut être calculé à partir de la clé publique chaque fois que nécessaire.

Le nom distinctif X.500 de confiance facultatif DOIT être présent afin que la clé publique d'ancre de confiance soit utilisée pour valider un chemin de certification X.509. Sans le nom distinctif X.500, la construction du chemin de certification ne peut pas utiliser l'ancre de confiance.

1.2.5 Exigences des algorithmes de chiffrement et de compression

Un paquetage de microcode pour un module de matériel cryptographique comporte des mises en œuvre d'algorithme cryptographique. De plus, un paquetage de microcode pour un module de matériel non cryptographique va probablement inclure des mises en œuvre d'algorithme cryptographique pour prendre en charge le chargeur d'amorçage dans la validation des paquetages de microcode.

Un identifiant d'objet d'algorithme unique DOIT être alloué pour chaque algorithme cryptographique et mode mis en œuvre par un paquetage de microcode. Un identifiant d'objet d'algorithme unique DOIT aussi être alloué pour chaque algorithme de compression mis en œuvre par un paquetage de microcode. Les identifiants d'objet d'algorithme peuvent être utilisés pour déterminer si un paquetage de microcode particulier satisfait les besoins d'une application particulière. Pour faciliter le développement d'applications à capacités d'algorithmes, l'interface de module cryptographique DEVRAIT permettre aux applications d'interroger le module cryptographique sur les identifiants d'objet associés à chaque algorithme cryptographique contenu dans le paquetage de microcode qui se charge. Les applications DEVRAIENT aussi être capables d'interroger le module cryptographique pour déterminer les attributs associés à chaque algorithme. De tels attributs peuvent inclure le type d'algorithme (chiffrement symétrique, chiffrement asymétrique, accord de clé, fonction de hachage unidirectionnel, signature numérique, et ainsi de suite) la taille du bloc d'algorithme ou la taille de module, et les paramètres pour les algorithmes asymétriques. La présente spécification n'établit pas les conventions pour la restitution des identifiants ou attributs d'algorithmes.

1.3 Architecture de sécurité de module de matériel

Le chargeur d'amorçage PEUT être mémorisé en permanence sur une mémoire en lecture seule ou chargé séparément dans une mémoire non volatile comme exposé plus haut.

Dans la plupart des conceptions de module de matériel, l'environnement d'exécution du paquetage de microcode offre un seul espace d'adresses. Si c'est le cas, le paquetage de microcode DEVRAIT contenir une charge complète de paquetage de microcode pour le module de matériel. Dans cette situation, le paquetage de microcode ne contient pas un ensemble de fonctions partiel ou incrémentaire. Une charge complète de paquetage de microcode va minimiser la complexité et éviter de potentiels problèmes de sécurité. Du point de vue de la complexité, le chargement incrémentaire des paquetages rend nécessaire que chaque paquetage identifie tous les autres paquetages qui sont requis (ses dépendances) et le chargeur d'amorçage a besoin de vérifier que toutes les dépendances soient satisfaites avant de tenter d'exécuter le paquetage de microcode. Lorsque un module de matériel se fonde sur un processeur général ou un processeur de signal numérique, il est dangereux de permettre que des paquetages arbitraires soient chargés simultanément sauf si il y a une surveillance des références pour s'assurer que des portions indépendantes du code ne peuvent pas interférer avec d'autres. Aussi, il est difficile d'évaluer des combinaisons arbitraires de modules logiciels [SECREQMTS]. Pour ces raisons, une charge complète de paquetage de microcode est RECOMMANDÉE ; cependant, la présente spécification permet au signataire du microcode d'identifier les dépendances entre des paquetages de microcode afin de traiter toutes les situations.

Les paquetages de microcode PEUVENT avoir des dépendances à des programmes fournis par d'autres paquetages de microcode. Pour minimiser la complexité de l'évaluation de la sécurité d'un module de matériel qui emploie une telle conception, le paquetage de microcode DOIT identifier les identifiants de paquetage (et les numéros de version minimums quand la forme de nom préféré de paquetage de microcode est utilisée) des paquetages dont il dépend. Le chargeur d'amorçage DOIT rejeter une charge de paquetage de microcode si elle contient une dépendance à un paquetage de microcode qui n'est pas disponible.

Le chargement d'un paquetage de microcode peut impacter la résolution satisfaisante des dépendances à d'autres paquetages de microcode qui font déjà partie de la configuration du module de matériel. Pour cette raison, le chargeur d'amorçage DOIT rejeter le chargement d'un paquetage de microcode si les dépendances d'un paquetage de microcode dans les configurations résultantes ne sont pas satisfaites.

1.4 Codage ASN.1

La CMS utilise la notation de syntaxe abstraite numéro un (ASN.1, *Abstract Syntax Notation One*) [X.208-88], [X.209-88]. L'ASN.1 est une notation formelle utilisée pour décrire les protocoles de données, sans considération du langage de programmation utilisé par la mise en œuvre. Les règles de codage décrivent comment les valeurs définies en ASN.1 seront représentées pour la transmission. Les règles de codage de base (BER, *Basic Encoding Rules*) sont l'ensemble de règles le plus largement employé, mais elles offrent plus qu'un moyen de représenter les structures de données. Par exemple, des codage de longueur définie et des codages de longueur indéfinie sont acceptés. Cette souplesse n'est pas désirable lorsque des signatures numériques sont utilisées. Par suite, les règles de codage distinctif (DER, *Distinguished Encoding Rules*) [X.509-88] ont été inventées. DER est un sous ensemble des BER qui assure une seule façon de représenter une certaine valeur. Par exemple, DER emploie toujours un codage de longueur définie.

Dans la présente spécification, les structures signées numériquement DOIVENT être codées avec les DER. D'autres structures n'exigent pas les DER, mais l'utilisation de codages de longueur définie est fortement RECOMMANDÉE. En utilisant toujours un codage de longueur définie, le chargeur d'amorçage aura moins d'options à mettre en œuvre. Dans des situations où on a une très forte assurance que seul un codage de longueur définie sera utilisé, la prise en charge de codages de longueur indéfinie PEUT être omise.

1.5 Chargement protégé de paquetage de microcode

Le présent document ne tente pas de spécifier une interface physique, un logiciel de pilote qui s'y rapporte, ou un protocole nécessaire pour charger les paquetages de microcode. De nombreux mécanismes de livraison différents sont envisagés, incluant des appareils à mémoire portable, le transfert de fichiers, et les pages de la Toile. La Section 2 de la présente spécification définit le format qui DOIT être présenté au module de matériel sans considération de l'interface utilisée. Cette spécification définit aussi le format de la réponse qui PEUT être générée par le module de matériel. La Section 3 définit le format qui PEUT être retourné par le module de matériel lorsque un paquetage de microcode se charge avec succès. La Section 4 définit le format qui PEUT être retourné par le module de matériel lorsque un paquetage de microcode ne réussit pas à se charger. Les récépissés de chargement de paquetage de microcode et les rapports de chargement de paquetage de microcode peuvent être signés ou non signés.

2. Protection de paquetage de microcode

La syntaxe de message cryptographique (CMS) est utilisée pour protéger un paquetage de microcode, qui est traité comme un objet binaire opaque. Une signature numérique est utilisée pour protéger le paquetage de microcode contre la modification non détectée et pour assurer l'authentification de l'origine des données. Le chiffrement est facultativement utilisé pour protéger le paquetage de microcode de la divulgation, et la compression est facultativement utilisée pour réduire la taille du paquetage de microcode protégé. Le type de données ContentInfo CMS DOIT toujours être présent, et il DOIT encapsuler le type de données SignedData CMS. Si le paquetage de microcode est chiffré, le type de données SignedData CMS DOIT alors encapsuler le type de données EncryptedData CMS. Si le paquetage de microcode est compressé, le type de données SignedData CMS (lorsque le chiffrement n'est pas utilisé) ou le type de données EncryptedData CMS (lorsque le chiffrement est utilisé) DOIT encapsuler le type de contenu CompressedData CMS. Finalement, (1) le type de données SignedData CMS (lorsque ni le chiffrement ni la compression ne sont utilisés) (2) le type de données EncryptedData CMS (lorsque le chiffrement est utilisé, mais pas la compression) ou (3) le type de contenu CompressedData CMS (lorsque la compression est utilisée) DOIT encapsuler le simple paquetage de microcode en utilisant le type de contenu FirmwarePkgData défini dans la présente spécification (au paragraphe 2.1.5).

La protection du paquetage de microcode est résumée comme suit (voir la syntaxe complète dans la [RFC3852]) :

```
ContentInfo {
  contentType       id-signedData,                  -- (1.2.840.113549.1.7.2)
  content           SignedData
}

SignedData {
  version           CMSVersion,                     -- toujours réglé à 3.
  digestAlgorithms    DigestAlgorithmIdentifiers,   -- seulement un.
  encapContentInfo    EncapsulatedContentInfo,
  certificates      CertificateSet,                 -- chemin de certification du signataire.
  crls              CertificateRevocationLists,     -- facultatif.
  signerInfos       ENSEMBLE DE SignerInfo          -- seulement un.
}
```

```
SignerInfo {
  version            CMSVersion,                    -- toujours réglé à 3.
  sid                SignerIdentifier,
  digestAlgorithm    DigestAlgorithmIdentifier,
  signedAttrs        SignedAttributes,              -- exigé.
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature          SignatureValue,
  unsignedAttrs      UnsignedAttributes             -- facultatif
}

EncapsulatedContentInfo {
  eContentType     id-encryptedData,          -- (1.2.840.113549.1.7.6)
                -- OU --
                   id-ct-compressedData,      -- (1.2.840.113549.1.9.16.1.9)
                -- OU --
                   id-ct-firmwarePackage,     -- (1.2.840.113549.1.9.16.1.16)
  eContent         CHAINE D'OCTETS            -- contient EncryptedData OU CompressedData OU FirmwarePkgData
}

EncryptedData {
  version            CMSVersion,              -- toujours réglé à 0.
  encryptedContentInfo EncryptedContentInfo,
  unprotectedAttrs   UnprotectedAttributes    -- omis
}

EncryptedContentInfo {
  contentType      id-ct-compressedData,      -- (1.2.840.113549.1.9.16.1.9)
                -- OU --
                   id-ct-firmwarePackage,     -- (1.2.840.113549.1.9.16.1.16)
  contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
  encryptedContent CHAINE D'OCTETS            -- contient CompressedData OU FirmwarePkgData
}

CompressedData {
  version            CMSVersion,              -- toujours réglé à 0.
  compressionAlgorithm CompressionAlgorithmIdentifier,
  encapContentInfo   EncapsulatedContentInfo
}

EncapsulatedContentInfo {
  eContentType       id-ct-firmwarePackage,   -- (1.2.840.113549.1.9.16.1.16)
  eContent           CHAINE D'OCTETS          -- contient FirmwarePkgData
}

FirmwarePkgData      CHAINE D'OCTETS          -- contient le paquetage de microcode
```

2.1 Profil de type de contenu de CMS de protection de paquetage de microcode

Cette section spécifie les conventions d'utilisation des types de contenu CMS ContentInfo, SignedData, EncryptedData, et CompressedData. Elle définit aussi le type de contenu FirmwarePkgData.

2.1.1 ContentInfo

La CMS exige que l'encapsulation la plus externe soit ContentInfo [RFC3852]. Les champs de ContentInfo sont utilisés comme suit :

contentType indique le type du contenu associé, et dans ce cas, le type encapsulé est toujours SignedData. L'identifiant d'objet id-signedData (1.2.840.113549.1.7.2) DOIT être présent dans ce champ.

content contient le contenu associé, et dans ce cas, le champ "content" DOIT contenir SignedData.

2.1.2 SignedData

Le type de contenu SignedData [RFC3852] contient le paquetage de microcode signé (qui peut être compressé, chiffré, ou compressé et ensuite chiffré avant la signature) les certificats nécessaires pour valider la signature, et une valeur de signature numérique. Les champs de SignedData sont utilisés comme suit :

version est le numéro de version de la syntaxe, et dans ce cas, il DOIT être réglé à 3.

digestAlgorithms est une collection d'identifiants d'algorithmes de résumé de message, et dans ce cas, il DOIT contenir un seul identifiant d'algorithme de résumé de message. L'algorithme de résumé de message employé par le signataire du paquetage de microcode DOIT être présent.

encapContentInfo contient le contenu signé, consistant en un identifiant de type de contenu et le contenu lui-même. L'utilisation du type EncapsulatedContentInfo est discuté au paragraphe 2.1.2.2.

certificates est une collection facultative de certificats. Si l'ancre de confiance a signé directement le paquetage de microcode, certificates DEVRAIT être omis. Si elle ne l'a pas fait, certificates DEVRAIT inclure le certificat X.509 du signataire du paquetage de microcode. L'ensemble de certificats DEVRAIT être suffisant pour que le chargeur d'amorçage construise un chemin de certification allant de l'ancre de confiance au certificat du signataire du microcode. Les certificats PKCS#6 étendus [PKCS#6] et les certificats d'attribut (de version 1 ou de version 2) [X.509-97], [X.509-00], [RFC3281] NE DOIVENT PAS être inclus dans l'ensemble des certificats.

crls est une collection facultative de listes de révocation de certificats (CRL), et dans ce cas, les CRL NE DEVRAIENT PAS être incluses par le signataire du paquetage de microcode. Il est prévu que les paquetages de microcode puissent être générés, signés, et rendus disponibles dans des répertoires pour être téléchargés dans les modules de matériel. Dans ce contexte, il serait difficile au signataire du paquetage de microcode d'inclure des CRL à jour dans le paquetage de microcode. Cependant, comme les CRL ne sont pas couvertes par la signature, des CRL à jour PEUVENT être insérées par un tiers avant que le paquetage de microcode soit livré au module de matériel.

signerInfos est une collection d'informations par signataire, et dans ce cas, la collection DOIT contenir exactement une SignerInfo. L'utilisation du type SignerInfo est discutée au paragraphe 2.1.2.1.

2.1.2.1 SignerInfo

Le signataire du paquetage de microcode est représenté dans le type SignerInfo. Les champs de SignerInfo sont utilisés comme suit :

version est le numéro de version de la syntaxe, et il DOIT être 3.

sid identifie la clé publique du signataire. La CMS prend en charge deux solutions : issuerAndSerialNumber *(producteur et numéro de série)* et subjectKeyIdentifier *(identifiant de clé sujette)*. Cependant, le chargeur d'amorçage DOIT prendre en charge la solution subjectKeyIdentifier, qui identifie directement la clé publique du signataire. Quand cette clé publique est contenue dans un certificat, cet identifiant DEVRAIT apparaître dans l'extension X.509 subjectKeyIdentifier.

digestAlgorithm identifie l'algorithme de résumé de message, et tous les paramètres associés, utilisé par le signataire du paquetage de microcode. Il DOIT contenir l'algorithme de résumé de message employé par le signataire du paquetage de microcode. (Noter que cet identifiant d'algorithme de résumé de message DOIT être le même que celui porté dans la valeur de digestAlgorithms dans SignedData.)

signedAttrs est une collection facultative d'attributs qui sont signés avec le contenu. signedAttrs est facultatif dans la CMS, mais dans la présente spécification, signedAttrs est EXIGÉ pour le paquetage de microcode ; cependant, les mises en œuvre DOIVENT ignorer les attributs signés non reconnus. L'ensemble (ENSEMBLE DE) des attributs DOIT être codé en DER [X.509-88]. Le paragraphe 2.2 de ce document fait la liste des attributs qui DOIVENT être inclus dans la collection ; d'autres attributs PEUVENT aussi être inclus.

signatureAlgorithm identifie l'algorithme de signature, et tous les paramètres associés, utilisé par le signataire du paquetage de microcode pour générer la signature numérique.

signature est la valeur de la signature numérique.

unsignedAttrs est un ensemble facultatif d'attributs qui ne sont pas signés. Comme décrit au paragraphe 2.3, cet ensemble ne peut contenir qu'une seule instance de l'attribut wrapped-firmware-decryption-key *(clé de déchiffrement de microcode enveloppé)* et aucun autre.

2.1.2.2 EncapsulatedContentInfo

Le type de contenu EncapsulatedContentInfo *(informations de contenu encapsulées)* encapsule le paquetage de microcode, qui peut être compressé, chiffré, ou compressé et ensuite chiffré avant signature. Le paquetage de microcode, dans tous ces formats, est porté au sein du type EncapsulatedContentInfo. Les champs de EncapsulatedContentInfo sont utilisés comme suit :

eContentType *(type de contenu encapsulé)* est un identifiant d'objet qui spécifie de façon univoque le type de contenu, et dans ce cas, la valeur DOIT être id-encryptedData *(identifiant de données chiffrées)* (1.2.840.113549.1.7.6), id-ct-compressedData *(identifiant de contenu de données compressées)* (1.2.840.113549.1.9.16.1.9), ou id-ct-firmwarePackage *(identifiant de contenu de paquetage de microcode)* (1.2.840.113549.1.9.16.1.16). Lorsque eContentType contient id-encryptedData, le paquetage de microcode a été chiffré avant d'être signé, et peut aussi avoir été compressé avant le chiffrement. Lorsque il contient id-ct-compressedData, le paquetage de microcode a été compressé avant d'être signé, mais n'a pas été chiffré. Lorsque il contient id-ct-firmwarePackage, le paquetage de microcode n'a pas été compressé ni chiffré avant d'être signé.

eContent *(contenu encapsulé)*contient le paquetage de microcode signé, qui peut aussi être chiffré, compressé, ou compressé et ensuite chiffré, avant d'être signé. Le contenu est codé comme une chaîne d'octets. La chaîne d'octets eContent n'a pas besoin d'être codée en DER.

2.1.3 EncryptedData

Le type de contenu EncryptedData [RFC3852] contient le paquetage de microcode chiffré (qui peut être compressé avant d'être chiffré). Cependant, si le paquetage de microcode n'a pas été chiffré, le type de contenu EncryptedData n'est pas présent. Les champs de EncryptedData sont utilisés comme suit :

version est le numéro de version de la syntaxe, et dans ce cas, version DOIT être 0.

encryptedContentInfo sont les informations de contenu chiffré. L'utilisation du type EncryptedContentInfo est discutée au paragraphe 2.1.3.1.

unprotectedAttrs est une collection facultative d'attributs non chiffrés, et dans ce cas, unprotectedAttrs NE DOIT PAS être présent.

2.1.3.1 EncryptedContentInfo

Le paquetage de microcode chiffré, qui peut être compressé avant le chiffrement, est encapsulé dans le type EncryptedContentInfo. Les champs de EncryptedContentInfo sont utilisés comme suit :

contentType indique le type de contenu, et dans ce cas, il DOIT contenir soit id-ct-compressedData (1.2.840.113549.1.9.16.1.9) soit id-ct-firmwarePackage (1.2.840.113549.1.9.16.1.16). Lorsque il contient id-ct-compressedData, le paquetage de microcode a été compressé avant le chiffrement. Lorsque il contient id-ct-firmwarePackage, le paquetage de microcode n'a pas été compressé avant le chiffrement.

contentEncryptionAlgorithm identifie l'algorithme de chiffrement de microcode, et tous paramètres associés, utilisé pour chiffrer le paquetage de microcode.

encryptedContent est le résultat du chiffrement du paquetage de microcode. Le champ est facultatif ; cependant, dans ce cas, il DOIT être présent.

2.1.4 CompressedData

Le type de contenu CompressedData [RFC3274] contient le paquetage de microcode compressé. Si le paquetage de microcode n'a pas été compressé, le type de contenu CompressedData n'est alors pas présent. Les champs de CompressedData sont utilisés comme suit :

version est le numéro de version de la syntaxe; dans ce cas, il DOIT être 0.

compressionAlgorithm identifie l'algorithme de compression, et tous paramètres associés, utilisé pour compresser le paquetage de microcode.

encapContentInfo est le contenu compressé, consistant en un identifiant de type de contenu et le contenu lui-même. L'utilisation du type EncapsulatedContentInfo est discutée au paragraphe 2.1.4.1.

2.1.4.1 EncapsulatedContentInfo

Le type de contenu CompressedData encapsule le paquetage de microcode compressé, et il est porté au sein du type EncapsulatedContentInfo. Les champs de EncapsulatedContentInfo sont utilisés comme suit :

eContentType est un identifiant d'objet qui spécifie de façon univoque le type de contenu, et dans ce cas, ce DOIT être la valeur de id-ct-firmwarePackage (1.2.840.113549.1.9.16.1.16).

eContent est le paquetage de microcode compressé, codé comme chaîne d'octets. La chaîne d'octets eContent n'a pas besoin d'être codée en DER.

2.1.5 FirmwarePkgData

Le type de contenu FirmwarePkgData *(données du paquetage de microcode)* contient le paquetage de microcode. C'est une encapsulation directe dans une chaîne d'octets, et il n'a pas besoin d'être codé en DER.

Le type de contenu FirmwarePkgData est identifié par l'identifiant d'objet id-ct-firmwarePackage :

IDENTIFIANT D'OBJET id-ct-firmwarePackage ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
smime(16) ct(1) 16 }

Le type de contenu FirmwarePkgData est une simple chaîne d'octets :

FirmwarePkgData ::= CHAINE D'OCTETS

2.2 Attributs signés

Le signataire du paquetage de microcode DOIT signer numériquement une collection d'attributs avec le paquetage de microcode. Chaque attribut de la collection DOIT être codé en DER [X.509-88]. La syntaxe des attributs est définie dans la [RFC3852], mais est répétée ici pour en faciliter l'accès :

```
Attribute ::= SEQUENCE {
    attrType IDENTIFIANT D'OBJET,
    attrValues ENSEMBLE DE AttributeValue }

AttributeValue ::= TOUTE
```

Chacun des attributs utilisés avec ce profil a une seule valeur d'attribut, bien que la syntaxe soit définie comme un ENSEMBLE DE AttributeValue. Il DOIT y avoir exactement une instance de AttributeValue présente.

La syntaxe de SignedAttributes au sein de signerInfo est définie comme un ENSEMBLE DE Attribute. SignedAttributes DOIT inclure seulement une instance de tout attribut particulier.

Le signataire du paquetage de microcode DOIT inclure les quatre attributs suivants : content-type *(type de contenu)*, message-digest *(résumé de message)*, firmware-package-identifier *(identifiant de paquetage de microcode)*, et target-hardware-module-identifiers *(identifiants de modules de matériel cibles)*.

Si le paquetage de microcode est chiffré, le signataire du paquetage de microcode DOIT alors aussi inclure l'attribut decrypt-key-identifier *(identifiant de clé de déchiffrement)*.

Si le paquetage de microcode met en œuvre des algorithmes de chiffrement, le signataire du paquetage de microcode PEUT alors aussi inclure l'attribut implemented-crypto-algorithms *(algorithmes de chiffrement mis en œuvre)*. De même, si le paquetage de microcode met en œuvre des algorithmes de compression, le signataire du paquetage de microcode PEUT alors aussi inclure l'attribut implemented-compress-algorithms *(algorithme de compression mis en œuvre)*.

Si le paquetage de microcode est destiné à n'être utilisé que par des communautés spécifiques, le signataire du paquetage de microcode DOIT alors aussi inclure l'attribut community-identifiers *(identifiants de communauté)*.

Si le paquetage de microcode dépend de la présence d'un ou plusieurs autres paquetages de microcode pour fonctionner correctement, le signataire du paquetage de microcode DEVRAIT alors aussi inclure l'attribut firmware-package-info *(informations de paquetage de microcode)*. Par exemple, le champ de dépendances d'attribut firmware-package-info peut

indiquer que le paquetage de microcode contient une dépendance à un chargeur d'amorçage ou noyau de séparation particulier.

Le signataire du paquetage de microcode DEVRAIT aussi inclure les trois attributs suivants : firmware-package-message-digest *(résumé de message de paquetage de microcode)*, signing-time *(heure de signature)*, et content-hints *(indications sur le contenu)*. De plus, si le signataire du paquetage de microcode a un certificat (ce qui signifie que le signataire du paquetage de microcode n'est pas toujours configuré comme une ancre de confiance) le signataire du paquetage de microcode DEVRAIT aussi inclure l'attribut signing-certificate.

Le signataire du paquetage de microcode PEUT inclure tout autre attribut qu'il estime approprié.

2.2.1 Type de contenu

Le signataire du paquetage de microcode DOIT inclure un attribut content-type avec la valeur de id-encryptedData (1.2.840.113549.1.7.6), id-ct-compressedData (1.2.840.113549.1.9.16.1.9), ou id-ct-firmwarePackage (1.2.840.113549.1.9.16.1.16). Lorsque il contient id-encryptedData, le paquetage de microcode a été chiffré avant d'être signé. Lorsque il contient id-ct-compressedData, le paquetage de microcode a été compressé avant d'être signé, mais n'a pas été chiffré. Lorsque il contient id-ct-firmwarePackage, le paquetage de microcode n'a été ni compressé ni chiffré avant la signature. Le paragraphe 11.1 de la [RFC3852] définit l'attribut content-type.

2.2.2 Résumé de message

Le signataire du paquetage de microcode DOIT inclure un attribut message-digest, ayant sa valeur de résumé de message calculée sur la chaîne d'octets eContent de encapContentInfo, comme défini au paragraphe 2.1.2.2. Cette chaîne d'octets contient le paquetage de microcode, et elle PEUT être compressée, chiffrée, ou à la fois compressée et chiffrée. Le paragraphe 11.2 de la [RFC3852] définit l'attribut message-digest.

2.2.3 Identifiant de paquetage de microcode

L'attribut firmware-package-identifier désigne le paquetage de microcode protégé. Deux approches de désignation des paquetages de microcode sont prises en charge : héritage et préférée. Le signataire du paquetage de microcode DOIT inclure un attribut firmware-package-identifier en utilisant une de ces formes de nom.

Un nom hérité de paquetage de microcode est une chaîne d'octets, et aucune structure n'est supposée au sein de la chaîne d'octets.

Un nom préféré de paquetage de microcode est une combinaison d'un identifiant d'objet et d'un numéro de version. L'identifiant d'objet désigne une collection de fonctions mises en œuvre par le paquetage de microcode, et le numéro de version est un entier non négatif qui identifie une construction ou livraison particulière du paquetage de microcode.

Si un paquetage de microcode est livré avec une faille désastreuse, le paquetage de microcode qui corrige la faille distribuée précédemment PEUT désigner une version périmée de paquetage de microcode pour empêcher le chargement de la version contenant la faille. Le chargeur d'amorçage du module de matériel DEVRAIT empêcher les replis ultérieurs sur la version périmée ou sur des versions antérieures à la version périmée. Lorsque la forme de nom héritée de paquetage de microcode est utilisée, la version périmée est indiquée par un nom hérité périmé de paquetage de microcode, qui est une chaîne d'octets. On suppose que le signataire du paquetage de microcode et le chargeur d'amorçage peuvent déterminer si un certain nom hérité de paquetage de microcode représente une version plus récente que celle périmée. Lorsque la forme de nom préféré de paquetage de microcode est utilisée, la version périmée est indiquée par un numéro de version périmée, qui est un entier.

L'identifiant d'objet suivant identifie l'attribut firmware-package-identifier :

```
IDENTIFIANT D'OBJET id-aa-firmwarePackageID  ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                                    smime(16) aa(2) 35 }
```

Les valeurs de l'attribut firmware-package-identifier ont un type ASN.1 de FirmwarePackageIdentifier :

```
FirmwarePackageIdentifier ::= SEQUENCE {
    name PreferredOrLegacyPackageIdentifier,
    stale PreferredOrLegacyStalePackageIdentifier FACULTATIF }

PreferredOrLegacyPackageIdentifier ::= CHOIX {
    preferred PreferredPackageIdentifier,
```

```
    legacy CHAINE D'OCTETS }

PreferredPackageIdentifier ::= SEQUENCE {
    fwPkgID IDENTIFIANT D'OBJET,
    verNum ENTIER (0..MAX) }

PreferredOrLegacyStalePackageIdentifier ::= CHOIX {
    preferredStaleVerNum ENTIER (0..MAX),
    legacyStaleVersion CHAINE D'OCTETS }
```

2.2.4 Identifiants de module de matériel cible

L'attribut target-hardware-module-identifiers désigne les types de modules de matériel que le paquetage de microcode prend en charge. Un identifiant d'objet unique désigne chaque type et révision de modèle de matériel pris en charge.

Le chargeur d'amorçage DOIT rejeter le paquetage de microcode si son propre identifiant de type de module de matériel ne figure pas sur la liste de l'attribut target-hardware-module-identifiers.

L'identifiant d'objet suivant identifie l'attribut target-hardware-module-identifiers:

```
IDENTIFIANT D'OBJET id-aa-targetHardwareIDs ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                                  smime(16) aa(2) 36 }
```

Les valeurs de l'attribut target-hardware-module-identifiers ont le type ASN.1 TargetHardwareIdentifiers :

```
TargetHardwareIdentifiers ::= SEQUENCE DE IDENTIFIANT D'OBJET
```

2.2.5 Identifiant de clé de déchiffrement

L'attribut decrypt-key-identifier désigne la clé symétrique nécessaire pour déchiffrer le paquetage de microcode encapsulé. Le type de données de CMS EncryptedData est utilisé lorsque le paquetage de microcode est chiffré. L'attribut signé decrypt-key-identifier est porté dans le type de contenu SignedData qui encapsule le type de contenu EncryptedData, désignant la clé symétrique nécessaire pour déchiffrer le paquetage de microcode. Aucune structure particulière n'est imposée à l'identifiant de clé. Les moyens par lesquels la clé de déchiffrement de microcode est distribuée de façon sûre à tous les modules qui sont autorisés à utiliser le paquetage de microcode associé sort du domaine d'application de la présente spécification ; cependant, un mécanisme facultatif pour distribuer en toute sécurité la clé de déchiffrement de microcode avec le paquetage de microcode est spécifié au paragraphe 2.3.1.

L'identifiant d'objet suivant identifie l'attribut decrypt-key-identifier :

```
IDENTIFIANT D'OBJET id-aa-decryptKeyID ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                             smime(16) aa(2) 37 }
```

Les valeurs de l'attribut decrypt-key-identifier ont le type ASN.1 DecryptKeyIdentifier :

```
DecryptKeyIdentifier ::= CHAINE D'OCTETS
```

2.2.6 Algorithmes de chiffrement mis en œuvre

L'attribut implemented-crypto-algorithms PEUT être présent dans le SignedAttributes, et il désigne les algorithmes de chiffrement qui sont mis en en œuvre par le paquetage de microcode et disponibles aux applications. Seuls les algorithmes qui sont disponibles à l'interface du module cryptographique sont énumérés. Tout algorithme de chiffrement qui est utilisé en interne et n'est pas accessible via l'interface du module cryptographique NE DOIT PAS figurer sur la liste. Par exemple, si le paquetage de microcode met en œuvre l'algorithme de déchiffrement pour de futures installations de paquetages de microcode et si cet algorithme n'est pas rendu disponible pour d'autres usages, l'algorithme de déchiffrement de microcode ne va pas figurer sur la liste.

La portion identifiant d'objet de AlgorithmIdentifier identifie un algorithme et son mode d'utilisation. Aucun paramètre d'algorithme n'est inclus. Les algorithmes de chiffrement incluent des algorithmes de chiffrement du trafic, des algorithmes de chiffrement de clé, des algorithmes de transport de clé, des algorithmes d'accord de clé, des algorithmes de hachage unidirectionnel, et des algorithmes de signature numérique. Les algorithmes de chiffrement n'incluent pas les algorithmes de compression.

L'identifiant d'objet suivant identifie l'attribut implemented-crypto-algorithms :

IDENTIFIANT D'OBJET id-aa-implCryptoAlgs ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) aa(2) 38 }

Les valeurs de l'attribut implemented-crypto-algorithms ont le type ASN.1 ImplementedCryptoAlgorithms :

ImplementedCryptoAlgorithms ::= SEQUENCE DE IDENTIFIANT D'OBJET

2.2.7 Algorithme de compression mis en œuvre

L'attribut implemented-compress-algorithms PEUT être présent dans le SignedAttributes, et il désigne les algorithmes de compression qui sont mis en œuvre par le paquetage de microcode et disponibles aux applications. Seuls les algorithmes qui sont disponibles à l'interface du module de matériel sont désignés. Tout algorithme de compression qui est utilisé en interne et n'est pas accessible via l'interface du module de matériel NE DOIT PAS être mentionné. Par exemple, si le paquetage de microcode met en œuvre un algorithme de décompression pour de futures installations de paquetage de microcode et si cet algorithme n'est pas disponible pour d'autres utilisations, cet algorithme de décompression de microcode ne sera pas mentionné.

La portion identifiant d'objet de AlgorithmIdentifier identifie un algorithme de compression. Aucun paramètre d'algorithme n'est inclus.

L'identifiant d'objet suivant identifie l'attribut implemented-compress-algorithms :

IDENTIFIANT D'OBJET id-aa-implCompressAlgs ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) aa(2) 43 }

Les valeurs de l'attribut implemented-compress-algorithms ont le type ASN.1 ImplementedCompressAlgorithms :

ImplementedCompressAlgorithms ::= SEQUENCE DE IDENTIFIANT D'OBJET

2.2.8 Identifiant de communauté

Si il est présent dans SignedAttributes, l'attribut community-identifiers désigne les communautés qui ont la permission d'exécuter le paquetage de microcode. Le chargeur d'amorçage DOIT rejeter le paquetage de microcode si le module de matériel n'est pas membre d'une des communautés identifiées. Les moyens pour devenir membre d'une communauté sortent du domaine d'application de la présente spécification.

Les attributs community-identifiers désignent les communautés autorisées par une liste d'identifiants d'objet de communauté, par une liste de modules de matériel spécifiques, ou par une combinaison des deux listes. Un module de matériel spécifique est spécifié par la combinaison de l'identifiant de module de matériel (comme défini paragraphe 2.2.4) et un numéro de série. Pour faciliter une représentation compacte des numéros de série, un bloc contigu peut être spécifié par le plus faible et le plus fort numéros de série autorisés. Autrement, tous les numéros de série associés à une famille de module de matériel peuvent être spécifiés avec la valeur NULL.

Si le chargeur d'amorçage n'a pas de mécanisme pour obtenir une liste d'identifiants d'objet qui identifie les communautés auxquelles appartient le module de matériel, le chargeur d'amorçage DOIT alors se comporter comme si la liste était vide. De même, si le chargeur d'amorçage n'a pas accès au numéro de série du module de matériel, il DOIT alors se comporter comme si le module de matériel n'était pas inclus sur la liste des modules de matériel autorisés.

L'identifiant d'objet suivant identifie l'attribut community-identifiers :

IDENTIFIANT D'OBJET id-aa-communityIdentifiers ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) aa(2) 40 }

Les valeurs de l'attribut community-identifiers ont le type ASN.1 CommunityIdentifiers :

CommunityIdentifiers ::= SEQUENCE DE CommunityIdentifier

CommunityIdentifier ::= CHOIX {
  communityOID IDENTIFIANT D'OBJET,

```
   hwModuleList HardwareModules }

HardwareModules ::= SEQUENCE {
   hwType IDENTIFIANT D'OBJET,
   hwSerialEntries SEQUENCE DE HardwareSerialEntry }

HardwareSerialEntry ::= CHOIX {
   tout NULL,
   une seule CHAINE D'OCTETS,
   bloc SEQUENCE {
     CHAINE D'OCTETS inférieure,
     CHAINE D'OCTETS supérieure } }
```

2.2.9 Informations de paquetage de microcode

Si un module de matériel prend en charge plus d'un type de paquetage de microcode, le signataire du paquetage de microcode DEVRAIT alors inclure l'attribut firmware-package-info avec un champ fwPkgType rempli pour identifier le type de paquetage de microcode. Cette valeur peut aider le chargeur d'amorçage au placement correct du paquetage de microcode au sein du module de matériel. Le type du paquetage de microcode est un ENTIER, et la signification de la valeur d'entier est spécifique à chaque module de matériel. Par exemple, un module de matériel pourrait allouer des valeurs d'entier différentes pour un chargeur d'amorçage, un noyau de séparation, et une application.

Certaines architectures de module de matériel permettent qu'un paquetage de microcode utilise des sous programmes fournis par un autre. Si le paquetage de microcode contient une dépendance à un autre, le signataire du paquetage de microcode DEVRAIT alors aussi inclure l'attribut firmware-package-info avec un champ de dépendances rempli. Si le paquetage de microcode ne dépend d'aucun autre paquetage de microcode, le signataire du paquetage de microcode NE DOIT alors PAS inclure l'attribut firmware-package-info avec un champ de dépendances rempli.

Les dépendances de paquetage de microcode sont identifiées par l'identifiant de paquetage de microcode ou par les informations contenues dans le paquetage de microcode lui-même, et dans l'un et l'autre cas, le chargeur d'amorçage s'assure que les dépendances sont satisfaites. Le chargeur d'amorçage DOIT rejeter un chargement de paquetage de microcode si il identifie une dépendance à un paquetage de microcode qui n'est pas encore chargé. Aussi, le chargeur d'amorçage DOIT rejeter un chargement de paquetage de microcode si l'action résulterait en une configuration où les dépendances d'un paquetage de microcode déjà chargé ne seraient plus satisfaites. Comme décrit au paragraphe 2.2.3, deux approches pour désigner les paquetages de microcode sont prises en charge : l'héritage et la préférence. Lorsque la forme de nom hérité de paquetage de microcode est utilisée, la dépendance est indiquée par un nom hérité de paquetage de microcode. On suppose que le signataire du paquetage de microcode et le chargeur d'amorçage peuvent déterminer si un certain nom hérité de paquetage de microcode représente la version désignée d'une version plus récente acceptable. Lorsque la forme préférée de nom de paquetage de microcode est utilisée, un identifiant d'objet et un entier sont fournis. L'identifiant d'objet DOIT correspondre exactement à la portion identifiant d'objet d'un nom préféré de paquetage de microcode associé au paquetage de microcode qui est déjà chargé, et l'entier DOIT être inférieur ou égal à la portion entière du nom préféré de paquetage de microcode associé au même paquetage de microcode. C'est-à-dire que la dépendance spécifie la valeur minimum de la version qui est acceptable.

L'identifiant d'objet suivant identifie l'attribut firmware-package-info :

```
IDENTIFIANT D'OBJET id-aa-firmwarePackageInfo ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                                                    pkcs9(9) smime(16) aa(2) 42 }
```

Les valeurs de l'attribut firmware-package-info ont le type ASN.1 FirmwarePackageInfo :

```
FirmwarePackageInfo ::= SEQUENCE {
   fwPkgType ENTIER FACULTATIF,
   dependencies SEQUENCE DE
   PreferredOrLegacyPackageIdentifier FACULTATIF }
```

2.2.10 Résumé de message de paquetage de microcode

Le signataire du paquetage de microcode DEVRAIT inclure un attribut firmware-package-message-digest, qui fournit l'algorithme de résumé de message et la valeur de résumé de message calculés sur le paquetage de microcode. Le résumé de message est calculé sur le paquetage de microcode avant toute compression, tout chiffrement, ou traitement de signature. Le chargeur d'amorçage PEUT utiliser ce résumé de message pour confirmer que le paquetage de microcode prévu a été

récupéré après que toutes les couches d'encapsulation ont été retirées.

L'identifiant d'objet suivant identifie l'attribut firmware-package-message-digest :

IDENTIFIANT D'OBJET id-aa-fwPkgMessageDigest ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) aa(2) 41 }

Les valeurs de l'attribut firmware-package-message-digest ont le type ASN.1 FirmwarePackageMessageDigest :

```
FirmwarePackageMessageDigest ::= SEQUENCE {
   algorithm AlgorithmIdentifier,
   msgDigest CHAINE D'OCTETS }
```

2.2.11 Heure de signature

Le signataire du paquetage de microcode DEVRAIT inclure un attribut signing-time, qui spécifie l'heure à laquelle la signature a été appliquée au paquetage de microcode. Le paragraphe 11.3 de la [RFC3852] définit l'attribut signing-time.

2.2.12 Indications de contenu

Le signataire du paquetage de microcode DEVRAIT inclure un attribut content-hints, incluant un bref texte de description du paquetage de microcode. Le texte est codé en UTF-8, qui prend en charge la plupart des systèmes d'écriture du monde [RFC3629]. Le paragraphe 2.9 de la [RFC2634] définit l'attribut content-hints.

Lorsque plusieurs couches d'encapsulation sont employées, l'attribut content-hints est inclus dans le SignedData le plus externe pour fournir des informations sur le contenu le plus interne. Dans ce cas, l'attribut content-hints donne un bref texte de description du paquetage de microcode, qui peut aider une personne à choisir le paquetage de microcode correct lorsque plus d'un est disponible.

Lorsque la forme de nom préférée de paquetage de microcode est utilisée, l'attribut content-hints peut fournir un lien avec un nom hérité de paquetage de microcode. Ceci est particulièrement utile quand un système existant de gestion de configuration est utilisé, mais les caractéristiques associées au nom préféré de paquetage de microcode sont réputées utiles. Un nom de paquetage de microcode associé à un tel système de gestion de configuration peut ressembler à quelque chose comme "R1234.C0(AJ11).D62.A02.11(b)". Inclure ces noms de paquetage de microcode dans le texte de description peut être utile aux développeurs en fournissant un lien clair entre les deux formes de nom.

L'attribut content-hints contient deux champs, et dans ce cas, les deux champs DOIVENT être présents. Les champs de ContentHints sont utilisés comme suit :

contentDescription donne un bref texte de description du paquetage de microcode.

contentType donne le type de contenu du type de contenu le plus interne, et dans ce cas, il DOIT être id-ct-firmwarePackage (1.2.840.113549.1.9.16.1.16).

2.2.13 Certificat de signature

Lorsque la clé publique du signataire du microcode est contenue dans un certificat, le signataire du paquetage de microcode DEVRAIT inclure un attribut signing-certificate pour identifier le certificat qui a été employé. Cependant, si la signature du paquetage de microcode n'a pas de certificat (ce qui signifie que la signature ne sera validée qu'avec la clé publique d'ancre de confiance) le signataire du paquetage de microcode est alors incapable d'inclure un attribut signing-certificate. Le paragraphe 5.4 de la [RFC2634] définit cet attribut.

L'attribut signing-certificate contient deux champs : certs et policies. Le champ certs DOIT être présent, et le champ policies PEUT être présent. Les champs de SigningCertificate sont utilisés comme suit :

certs contient une séquence d'identifiants de certificats. Dans ce cas, la séquence des identifiants de certificat contient une seule entrée. Le champ certs DOIT contenir seulement l'identifiant de certificat du certificat qui contient la clé publique utilisée pour vérifier la signature du paquetage de microcode. Le champ certs utilise la syntaxe ESSCertID spécifiée au paragraphe 5.4 de la [RFC2634], et il est composé du hachage SHA-1 [SHA1] du certificat entier codé en DER ASN.1 et, facultativement, du producteur du certificat et du numéro de série du certificat. La valeur du hachage SHA-1 DOIT être présente. Le producteur du certificat et le numéro de série du certificat DEVRAIENT être présents.

policies est facultatif ; lorsque il est présent, il contient une séquence d'informations de politique. Le champ policies, lorsque il est présent, DOIT contenir une seule entrée, et cette entrée DOIT correspondre à une des politiques de certificat dans l'extension "politiques de certificat" du certificat qui contient la clé publique utilisée pour vérifier la signature du paquetage de microcode. Le champ policies utilise la syntaxe de PolicyInformation spécifiée au paragraphe 4.2.1.5 de la [RFC3280], et il se compose de l'identifiant d'objet de politique de certificat et, facultativement, des qualificatifs de politique de certificat. L'identifiant d'objet de politique de certificat DOIT être présent. Les qualificatifs de politique de certificat NE DEVRAIENT PAS être présents.

2.3 Attributs non signés

La CMS permet d'inclure un ENSEMBLE d'attributs non signés ; cependant, dans la présente spécification, l'ensemble DOIT être absent ou inclure une seule instance de l'attribut wrapped-firmware-decryption-key. Parce que la signature numérique ne couvre pas cet attribut, il peut être altéré en tout point du chemin de livraison du signataire du paquetage de microcode au module de matériel. Cette propriété peut être employée pour distribuer la clé de déchiffrement de microcode avec un paquetage de microcode chiffré et signé, permettant que la clé de déchiffrement de microcode soit enveloppée avec une clé de chiffrement de clé différente pour chaque liaison de la chaîne de distribution.

La syntaxe des attributs est définie dans la [RFC3852], et elle est répétée au début du paragraphe 2.2 du présent document. Chacun des attributs utilisés avec ce profil a une seule valeur d'attribut, même si la syntaxe est définie comme ENSEMBLE DE AttributeValue. Il DOIT y avoir exactement une instance de AttributeValue présente.

La syntaxe de UnsignedAttributes au sein de signerInfo est définie comme un ENSEMBLE DE Attribute. UnsignedAttributes DOIT inclure seulement une instance de tout attribut particulier.

2.3.1 Cle de déchiffrement de microcode enveloppée

Le signataire du paquetage de microcode, ou toute autre partie de la chaîne de distribution, PEUT inclure un attribut wrapped-firmware-decryption-key.

L'identifiant d'objet suivant identifie l'attribut wrapped-firmware-decryption-key :

IDENTIFIANT D'OBJET id-aa-wrappedFirmwareKey ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
pkcs9(9) smime(16) aa(2) 39 }

Les valeurs de l'attribut wrapped-firmware-decryption-key ont le type ASN.1 EnvelopedData. La Section 6 de la [RFC3852] définit le type de contenu EnvelopedData, qui est utilisé pour construire la valeur de l'attribut. EnvelopedData permet de protéger la clé de déchiffrement de microcode en utilisant des techniques symétriques ou asymétriques. EnvelopedData ne comporte aucun contenu chiffré ; on emploie plutôt la caractéristique de EnvelopedData d'avoir le contenu chiffré dans un autre endroit. Le contenu chiffré se trouve dans le champ eContent de la structure EncryptedData. La clé de déchiffrement de microcode est contenue dans le champ recipientInfos. La Section 6 de la [RFC3852] appelle cette clé une clé de chiffrement de contenu *(content-encryption key)*.

La syntaxe de EnvelopedData prend en charge de nombreux algorithmes de gestion de clés différents. Quatre techniques générales sont prises en charge : le transport de clé, l'accord de clé, les clés de chiffrement de clé symétriques, et les mots de passe.

Le type de contenu EnvelopedData est profilé pour l'attribut wrapped-firmware-decryption-key. Les champs de EnvelopedData sont pleinement décrits à la Section 6 de la [RFC3852]. Des règles supplémentaires s'appliquent lorsque EnvelopedData est utilisé comme attribut wrapped-firmware-decryption-key.

Dans la structure EnvelopedData, ce qui suit s'applique :
- L'ensemble des certificats inclus dans OriginatorInfo NE DOIT PAS inclure de certificat avec un type de extendedCertificate, v1AttrCert, ou v2AttrCert [X.509-97], [X.509-00], [RFC3281]. Le champ facultatif crls PEUT être présent.
- Le champ facultatif unprotectedAttrs NE DOIT PAS être présent.

Dans la structure EncryptedContentInfo, ce qui suit s'applique :
- contentType DOIT correspondre à l'identifiant d'objet de type de contenu porté dans le champ contentType au sein de la structure EncryptedContentInfo de EncryptedData comme décrit au paragraphe 2.1.3.1.
- contentEncryptionAlgorithm identifie l'algorithme de chiffrement de microcode, et tous les paramètres associés, utilisé pour chiffrer le paquetage de microcode porté dans le champ encryptedContent de la structure EncryptedContentInfo de EncryptedData. Donc, il DOIT correspondre exactement à la valeur de la structure EncryptedContentInfo de

EncryptedData comme décrit au paragraphe 2.1.3.1.
- encryptedContent est facultatif, et dans ce cas, il NE DOIT PAS être présent.

3. Récépissé de charge de paquetage de microcode

La syntaxe de message cryptographique (CMS) est utilisée pour indiquer qu'un paquetage de microcode a été chargé avec succès. La prise en charge de récépissés de chargement de paquetage de microcode est FACULTATIVE. Cependant, les modules de matériel qui choisissent de générer de tels récépissés DOIVENT respecter les conventions spécifiées dans cette section. Parce que tous les modules de matériel n'ont pas de clé de signature privée, le récépissé de chargement de paquetage de microcode peut être signé ou non signé. L'utilisation de récépissés de chargement de paquetage de microcode signés est RECOMMANDÉE.

Les modules de matériel qui prennent en charge la génération de récépissés DOIVENT avoir un numéro de série univoque. Les modules de matériel qui prennent en charge la génération de récépissés signés DOIVENT avoir une clé de signature privée pour signer le récépissé et le certificat de validation de signature correspondant ou son désignataire. Le désignataire est le nom du producteur du certificat et le numéro de série du certificat, ou c'est l'identifiant de clé publique. Les modules de matériel à mémoire restreinte vont généralement mémoriser l'identifiant de clé publique car il exige moins de capacité mémoire.

Le récépissé de chargement de paquetage de microcode non signé est encapsulé par ContentInfo. Autrement, le récépissé de chargement de paquetage de microcode signé est encapsulé par SignedData, qui est à son tour encapsulé par ContentInfo.

Le récépissé de chargement de paquetage de microcode est résumé comme suit (voir la syntaxe complète dans la [RFC3852]) :

```
ContentInfo {
  contentType    id-signedData,                  -- (1.2.840.113549.1.7.2)
                 -- OU --
                 id-ct-firmwareLoadReceipt,      -- (1.2.840.113549.1.9.16.1.17)
  content        SignedData
                 -- OU --
                 FirmwarePackageLoadReceipt
}

SignedData {
  version              CMSVersion,                -- toujours réglé à 3
  digestAlgorithms     DigestAlgorithmIdentifiers, -- seulement un
  encapContentInfo     EncapsulatedContentInfo,
  certificates         CertificateSet,            -- certificat de module facultatif
  crls                 CertificateRevocationLists, -- facultatif
  signerInfos          ENSEMBLE DE SignerInfo     -- seulement un
}

SignerInfo {
  version              CMSVersion                 -- réglé à 1 ou à 3
  sid                  SignerIdentifier,
  digestAlgorithm      DigestAlgorithmIdentifier,
  signedAttrs          SignedAttributes,          -- exigé
  signatureAlgorithm   SignatureAlgorithmIdentifier,
  signature            SignatureValue,
  unsignedAttrs        UnsignedAttributes         -- Omis
}

EncapsulatedContentInfo {
  eContentType   d-ct-firmwareLoadReceipt,  - (1.2.840.113549.1.9.16.1.17)
  eContent       CHAINE D'OCTETS            - Contient les récépissés
}

FirmwarePackageLoadReceipt {
  version              ENTIER                     -- la valeur DEFAUT est toujours utilisée
  hwType               IDENTIFIANT D'OBJET,       -- type de module de matériel
```

```
  hwSerialNum          CHAINE D'OCTETS,                -- numéro de série du module de
                                                       matériel/microcode
  fwPkgName            PreferredOrLegacyPackageIdentifier,
  trustAnchorKeyID     CHAINE D'OCTETS,                -- facultatif
  decryptKeyID         CHAINE D'OCTETS,                -- facultatif
}
```

3.1 Profil de type de contenu de CMS de réception de charge de paquetage de microcode

Ce paragraphe spécifie les conventions pour l'utilisation des types de contenu de CMS ContentInfo et SignedData pour les récépissés de chargement de paquetage de microcode. Il définit aussi le type de contenu de récépissé de chargement de paquetage de microcode.

3.1.1 ContentInfo

La CMS exige que l'encapsulation la plus externe soit ContentInfo [RFC3852]. Les champs de ContentInfo sont utilisés comme suit :

contentType indique le type du contenu associé. Si le récépissé de chargement de paquetage de microcode est signé, le type encapsulé DOIT alors être SignedData, et l'identifiant d'objet id-signedData (1.2.840.113549.1.7.2) DOIT être présent dans ce champ. Si le récépissé n'est pas signé, le type encapsulé DOIT alors être FirmwarePackageLoadReceipt, et l'identifiant d'objet id-ct-firmwareLoadReceipt (1.2.840.113549.1.9.16.1.17) DOIT être présent dans ce champ.

content renferme le contenu associé. Si le récépissé de chargement de paquetage de microcode est signé, ce champ DOIT alors contenir le SignedData. Si le récépissé n'est pas signé, ce champ DOIT alors contenir FirmwarePackageLoadReceipt.

3.1.2 SignedData

Le type de contenu SignedData contient le récépissé de chargement du paquetage de microcode et une signature numérique. Si le module de matériel mémorise en local son certificat, celui-ci peut aussi être inclus. Les champs de SignedData sont utilisés comme suit :

version est le numéro de version de syntaxe, et dans ce cas, il DOIT être réglé à 3.

digestAlgorithms est une collection d'identifiants d'algorithme de résumé de message, et dans ce cas, il DOIT contenir un seule identifiant d'algorithme de résumé de message. Les algorithmes de résumé de message employés par le module de matériel DOIVENT être présents.

encapContentInfo est le contenu signé, consistant en un identifiant de type de contenu et le contenu lui-même. L'utilisation du type EncapsulatedContentInfo est expliquée au paragraphe 3.1.2.2.

certificates est une collection facultative de certificats. Si le module de matériel mémorise en local son certificat, le certificat X.509 du module de matériel DEVRAIT alors être inclus. Si il ne le fait pas, le champ certificates est alors omis. Les certificats étendus [PKCS#6] et les certificats d'attribut (version 1 ou version 2) [X.509-97], [X.509-00], [RFC32871] NE DOIVENT PAS être inclus dans l'ensemble des certificats.

crls est une collection facultative de listes de révocation de certificats (CRL). Les CRL PEUVENT être incluses, mais elles seront normalement omises car les modules de matériel n'auront généralement pas accès à la CRL la plus récente. Les receveurs de récépissés signés DEVRAIENT être capables de traiter la présence du champ facultatif crls.

signerInfos est une collection d'informations par signataire, et dans ce cas, la collection DOIT contenir exactement une SignerInfo. L'utilisation du type SignerInfo est exposée au paragraphe 3.1.2.1.

3.1.2.1 SignerInfo

Le module de matériel est représenté dans le type SignerInfo *(informations provenant du signataire)*. Les champs de SignerInfo sont utilisés comme suit :

version est le numéro de version de syntaxe, et il DOIT être 1 ou 3, selon la méthode utilisée pour identifier la clé publique du module de matériel. L'utilisation de subjectKeyIdentifier est RECOMMANDÉE, qui résulte en l'utilisation de la version 3.

sid spécifie le certificat du module de matériel (et par là la clé publique du module de matériel). La CMS prend en charge issuerAndSerialNumber et subjectKeyIdentifier. Le module de matériel DOIT prendre en charge un des deux ou les deux pour la génération des récépissés ; cependant, la prise en charge de subjectKeyIdentifier est RECOMMANDÉE. issuerAndSerialNumber identifie le certificat du module de matériel par le nom distinctif du producteur et le numéro de série du certificat. Le certificat identifié contient lui aussi la clé publique du module de matériel. subjectKeyIdentifier identifie directement la clé publique du module de matériel. Lorsque cette clé publique est contenue dans un certificat, cet identifiant DEVRAIT apparaître dans l'extension X.509 subjectKeyIdentifier.

digestAlgorithm identifie l'algorithme de résumé de message, et tous les paramètres associés, utilisé par le module de matériel. Il DOIT contenir les algorithmes de résumé de message employés pour signer le récépissé. (Noter que cet identifiant d'algorithme de résumé de message DOIT être le même que celui porté dans la valeur digestAlgorithms dans SignedData.)

signedAttrs est une collection facultative d'attributs qui sont signés avec le contenu. Les signedAttrs sont facultatifs dans la CMS, mais dans la présente spécification, les signedAttrs sont EXIGÉS pour l'utilisation avec le contenu du récépissé de chargement de paquetage de microcode. L'ENSEMBLE DE attributs DOIT être codé en DER [X.509-88]. Le paragraphe 3.2 du présent document fait la liste des attributs qui DOIVENT être inclus dans la collection. D'autres attributs PEUVENT être inclus, mais le receveur va ignorer tout attribut signé non reconnu.

signatureAlgorithm identifie l'algorithme de signature, et tout paramètre associé, utilisé pour signer le récépissé.

signature est la signature numérique.

unsignedAttrs est une collection facultative d'attributs qui ne sont pas signés, et dans ce cas, il NE DOIT PAS y avoir d'attribut non signé présent.

3.1.2.2 EncapsulatedContentInfo

Le récépissé de chargement de paquetage de microcode est encapsulé dans une CHAINE D'OCTETS, et il est porté au sein du type EncapsulatedContentInfo. Les champs de EncapsulatedContentInfo sont utilisés comme suit :

eContentType est un identifiant d'objet qui spécifie de façon univoque le type de contenu, et dans ce cas, il DOIT avoir la valeur de id-ct-firmwareLoadReceipt (1.2.840.113549.1.9.16.1.17).

eContent est le récépissé de chargement de paquetage de microcode, encapsulé dans une CHAINE D'OCTETS. La chaîne d'octets eContent n'a pas besoin d'être codée en DER.

3.1.3 FirmwarePackageLoadReceipt

L'identifiant d'objet suivant identifie le type du contenu du récépissé de chargement de paquetage de microcode :

```
IDENTIFIANT D'OBJET id-ct-firmwareLoadReceipt ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                                                    pkcs9(9) smime(16) ct(1) 17 }
```

Le type de contenu du récépissé de chargement de paquetage de microcode a le type ASN.1 FirmwarePackageLoadReceipt :

```
FirmwarePackageLoadReceipt ::= SEQUENCE {
    version FWReceiptVersion DEFAULT v1,
    hwType IDENTIFIANT D'OBJET,
    hwSerialNum CHAINE D'OCTETS,
    fwPkgName PreferredOrLegacyPackageIdentifier,
    trustAnchorKeyID CHAINE D'OCTETS FACULTATIF,
    decryptKeyID [1] CHAINE D'OCTETS FACULTATIF }

FWReceiptVersion ::= ENTIER { v1(1) }
```

Les champs du type FirmwarePackageLoadReceipt ont la signification suivante :

version est un entier qui donne le numéro de version de syntaxe pour la compatibilité avec les futures révisions de la présente spécification. Les mises en œuvre qui se conforment à la présente spécification DOIVENT régler la version à la valeur par défaut, qui est v1.

hwType *(type de matériel)* est un identifiant d'objet qui identifie le type de module de matériel sur lequel le paquetage de microcode a été chargé.

hwSerialNum est le numéro de série du module de matériel sur lequel le paquetage de microcode a été chargé. Aucune structure particulière n'est imposée au numéro de série ; il n'est pas nécessaire qu'il soit un entier. Cependant, la combinaison de hwType et de hwSerialNum identifie de façon univoque le module de matériel.

fwPkgName identifie le paquetage de microcode qui a été chargé. Comme décrit au paragraphe 2.2.3, deux approches pour nommer les paquetages de microcode sont prises en charge : héritage et préférée. Un nom de paquetage de microcode hérité est une chaîne d'octets. Un nom préféré de paquetage de microcode est une combinaison de l'identifiant d'objet du paquetage de microcode et d'un numéro de version entier.

trustAnchorKeyID est facultatif, et lorsque il est présent, il identifie l'ancre de confiance qui a été utilisée pour valider la signature du paquetage de microcode.

decryptKeyID est facultatif, et lorsque il est présent, il identifie la clé de déchiffrement de microcode qui a été utilisée pour déchiffrer le paquetage de microcode.

Le récépissé de chargement de paquetage de microcode DOIT inclure les champs version, hwType, hwSerialNum, et fwPkgName, et il DEVRAIT inclure le champ trustAnchorKeyID. Le récépissé de chargement de paquetage de microcode NE DOIT PAS inclure le decryptKeyID, sauf si le paquetage de microcode associé au récépissé est chiffré, si la clé de déchiffrement de microcode est disponible au module de matériel, et si le paquetage de microcode a été déchiffré avec succès.

3.2 Attributs signés

Le module de matériel DOIT signer numériquement une collection d'attributs avec le récépissé de chargement de paquetage de microcode. Chaque attribut de la collection DOIT être codé en DER [X.509-88]. La syntaxe des attributs est définie dans la [RFC3852], et elle a été répétée au paragraphe 2.2 à des fins d'illustration.

Chacun des attributs utilisés avec ce profil a une seule valeur d'attribut, même si la syntaxe est définie comme un ENSEMBLE DE AttributeValue. Il DOIT y avoir exactement une instance de AttributeValue présente.

La syntaxe de SignedAttributes au sein de signerInfo est définie comme un ENSEMBLE DE Attributes. SignedAttributes DOIT inclure seulement une instance de tout attribut particulier.

Le module de matériel DOIT inclure les attributs content-type et message-digest. Si le module de matériel inclut une horloge en temps réel, le module de matériel DEVRAIT alors aussi inclure l'attribut signing-time. Le module de matériel PEUT inclure tout autre attribut réputé approprié.

3.2.1 Type de contenu

Le module de matériel DOIT inclure un attribut content-type avec la valeur de id-ct-firmwareLoadReceipt (1.2.840.113549.1.9.16.1.17). Le paragraphe 11.1 de la [RFC3852] définit l'attribut content-type.

3.2.2. Résumé de message

Le module de matériel DOIT inclure un attribut message-digest, ayant comme valeur de résumé de message le contenu FirmwarePackageLoadReceipt. Le paragraphe 11.2 de la [RFC3852] définit l'attribut message-digest.

3.2.3 Heure de signature

Si le module de matériel inclut une horloge en temps réel, le module de matériel DEVRAIT alors inclure un attribut signing-time, spécifiant l'heure de génération du récépissé. Le paragraphe 11.3 de la [RFC3852] définit l'attribut signing-time.

4. Erreur de chargement de paquetage de microcode

La syntaxe de message cryptographique (CMS) est utilisée pour indiquer qu'une erreur s'est produite lors d'une tentative de chargement d'un paquetage de microcode protégé. La prise en charge des rapports d'erreur de chargement de paquetage de microcode est FACULTATIVE. Cependant, les modules de matériel qui choisissent de générer de tels rapports d'erreur DOIVENT suivre les conventions spécifiées dans cette section. Tous les modules de matériel n'ont pas de clé de signature privée ; donc, le rapport d'erreur de chargement de paquetage de microcode peut être soit signé, soit non signé. L'utilisation de rapport d'erreur de chargement de paquetage de microcode signé est RECOMMANDÉE.

Les modules de matériel qui prennent en charge la génération de rapport d'erreur DOIVENT avoir un numéro de série univoque. Les modules de matériel qui prennent en charge la génération de rapport d'erreur signé DOIVENT aussi avoir une clé de signature privée pour signer le rapport d'erreur et le certificat de validation de signature correspondant ou son désignateur. Le désignateur est le nom du producteur du certificat et le numéro de série du certificat, ou son identifiant de clé publique. Les modules de matériel qui ont des contraintes de mémoire vont généralement mémoriser l'identifiant de clé publique car il exige moins de mémoire.

Le rapport d'erreur de chargement de paquetage de microcode non signé est encapsulé par ContentInfo. Autrement, le rapport d'erreur de chargement de paquetage de microcode signé est encapsulé par SignedData, qui à son tour est encapsulé par ContentInfo.

Le rapport d'erreur de chargement de paquetage de microcode est résumé comme suit (voir la syntaxe complète dans la [RFC3852]) :

```
ContentInfo {
  contentType    id-signedData,                  -- (1.2.840.113549.1.7.2)
                 -- OU --
                 id-ct-firmwareLoadError,        -- (1.2.840.113549.1.9.16.1.18)
  content        SignedData
                 -- OU --
                 FirmwarePackageLoadError
}

SignedData {
  version              CMSVersion,                -- toujours réglé à 3,
  digestAlgorithms     DigestAlgorithmIdentifiers, -- seulement un,
  encapContentInfo     EncapsulatedContentInfo,
  certificates         CertificateSet,            -- certificat de module facultatif.
  crls                 CertificateRevocationLists, -- facultatif.
  signerInfos          ENSEMBLE DE SignerInfo     --  seulement un.
}

SignerInfo {
  version              CMSVersion,                -- réglé soit à 1, soit à 3
  sid                  SignerIdentifier,
  digestAlgorithm      DigestAlgorithmIdentifier,
  signedAttrs          SignedAttributes,          -- exigé.
  signatureAlgorithm   SignatureAlgorithmIdentifier,
  signature            SignatureValue,
  unsignedAttrs        UnsignedAttributes         -- omis.
}

EncapsulatedContentInfo {
  eContentType    id-ct-firmwareLoadError,    -- (1.2.840.113549.1.9.16.1.18)
  eContent        CHAINE D'OCTETS             -- contient le rapport d'erreur
}

FirmwarePackageLoadError {
  version              ENTIER,                      -- la valeur par DEFAUT est toujours utilisée.
  hwType               IDENTIFIANT D'OBJET,         -- type de module de matériel
  hwSerialNum          CHAINE D'OCTETS,             -- numéro de série du module de matériel ou de microcode
  errorCode            FirmwarePackageLoadErrorCode, -- identifiant d'erreur
  vendorErrorCode      VendorErrorCode,             -- facultatif
```

```
  fwPkgName          PreferredOrLegacyPackageIdentifier,    -- facultatif
  config             SEQUENCE DE CurrentFWConfig,           -- facultatif
}

CurrentFWConfig {                                           -- répété pour chaque paquetage dans la configuration.
  fwPkgType          ENTIER,                                -- type de paquetage de microcode ; facultatif.
  fwPkgName          PreferredOrLegacyPackageIdentifier
}
```

4.1 Profil de type de contenu de CMS d'erreur de chargement de paquetage de microcode

Cette section spécifie les conventions d'utilisation des types de contenu de CMS ContentInfo et SignedData pour les rapports d'erreur de chargement de paquetage de microcode. Elle définit aussi le type de contenu d'erreur de chargement de paquetage de microcode.

4.1.1 ContentInfo

La CMS exige que l'encapsulation la plus externe soit ContentInfo [RFC3852]. Les champs de ContentInfo sont utilisés comme suit :

contentType indique le type de contenu associé. Si le rapport d'erreur de chargement de paquetage de microcode est signé, le type encapsulé DOIT alors être SignedData, et l'identifiant d'objet id-signedData (1.2.840.113549.1.7.2) DOIT être présent dans ce champ. Si le rapport n'est pas signé, le type encapsulé DOIT alors être FirmwarePackageLoadError, et l'identifiant d'objet id-ct-firmwareLoadError (1.2.840.113549.1.9.16.1.18) DOIT être présent dans ce champ.

content détient le contenu associé. Si le rapport d'erreur de chargement de paquetage de microcode est signé, ce champ DOIT alors contenir les SignedData. Si le rapport n'est pas signé, ce champ DOIT alors contenir la FirmwarePackageLoadError *(erreur de chargement de paquetage de microcode)*.

4.1.2 SignedData

Le type de contenu SignedData contient le rapport d'erreur de chargement de paquetage de microcode et une signature numérique. Si le module de matériel mémorise en local son certificat, celui-ci peut alors être inclus aussi. Les champs de SignedData sont utilisés exactement comme décrit au paragraphe 3.1.2.

4.1.2.1 SignerInfo

Le module de matériel est représenté dans le type SignerInfo. Les champs de SignerInfo sont utilisés exactement comme décrit au paragraphe 3.1.2.1.

4.1.2.2 EncapsulatedContentInfo

FirmwarePackageLoadError est encapsulée dans une CHAINE D'OCTETS, et est portée au sein du type EncapsulatedContentInfo. Les champs de EncapsulatedContentInfo sont utilisés comme suit :

eContentType est un identifiant d'objet qui spécifie de façon univoque le type de contenu, et dans ce cas, il DOIT être la valeur de id-ct-firmwareLoadError (1.2.840.113549.1.9.16.1.18).

eContent est le rapport d'erreur de chargement de paquetage de microcode, encapsulé dans une CHAINE D'OCTETS. La chaîne d'octets eContent n'a pas besoin d'être codée en DER.

4.1.3 FirmwarePackageLoadError

L'identifiant d'objet suivant identifie le type de contenu du rapport d'erreur de chargement de paquetage de microcode :

```
IDENTIFIANT D'OBJET id-ct-firmwareLoadError ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                                  smime(16) ct(1) 18 }
```

Le type de contenu du rapport d'erreur de chargement de paquetage de microcode a le type ASN.1 FirmwarePackageLoadError :

```
FirmwarePackageLoadError ::= SEQUENCE {
```

```
        version FWErrorVersion DEFAUT v1,
        hwType IDENTIFIANT D'OBJET,
        hwSerialNum CHAINE D'OCTETS,
        errorCode FirmwarePackageLoadErrorCode,
        vendorErrorCode VendorLoadErrorCode FACULTATIF,
        fwPkgName PreferredOrLegacyPackageIdentifier FACULTATIF,
        config [1] SEQUENCE DE CurrentFWConfig FACULTATIF }

    FWErrorVersion ::= ENTIER { v1(1) }

    CurrentFWConfig ::= SEQUENCE {
        fwPkgType ENTIER FACULTATIF,
        fwPkgName PreferredOrLegacyPackageIdentifier }

    FirmwarePackageLoadErrorCode ::= ENUMERATED {
      decodeFailure              (1), (échec de décodage)
      badContentInfo             (2), (mauvaises informations de contenu)
      badSignedData              (3), (mauvaises données signées)
      badEncapContent            (4), (mauvais contenu encapsulé)
      badCertificate             (5), (mauvaises certificat)
      badSignerInfo              (6), (mauvaises informations de signataire)
      badSignedAttrs             (7), (mauvais attributs signés)
      badUnsignedAttrs           (8), (mauvais attributs non signés)
      missingContent             (9), (contenu manquant)
      noTrustAnchor              (10), (pas d'ancre de confiance)
      notAuthorized              (11), (non autorisé)
      badDigestAlgorithm         (12), (mauvais algorithme de résumé)
      badSignatureAlgorithm      (13), (mauvais algorithme de signature)
      unsupportedKeySize         (14), (taille de clé non accepté)
      signatureFailure           (15), (échec de signature)
      contentTypeMismatch        (16), (discordance de type de contenu)
      badEncryptedData           (17), (mauvaises données chiffrées)
      unprotectedAttrsPresent    (18), (attributs présents non protégés)
      badEncryptContent          (19), (mauvais contenu chiffré)
      badEncryptAlgorithm        (20), (mauvais algorithme de chiffrement)
      missingCiphertext          (21), (texte chiffré manquant)
      noDecryptKey               (22), (pas de clé de déchiffrement)
      decryptFailure             (23), (échec de déchiffrement)
      badCompressAlgorithm       (24), (mauvais algorithme de compression)
      missingCompressedContent   (25), (contenu compressé manquant)
      decompressFailure          (26), (échec de décompression)
      wrongHardware              (27), (mauvais matériel)
      stalePackage               (28), (paquetage périmé)
      notInCommunity             (29), (pas dans la communauté)
      unsupportedPackageType     (30), (type de paquetage non accepté)
      missingDependency          (31), (dépendance manquante)
      wrongDependencyVersion     (32), (mauvaise version de dépendance)
      insufficientMemory         (33), (mémoire insuffisante)
      badFirmware                (34), (mauvais microcode)
      unsupportedParameters      (35), (paramètre non pris en charge)
      breaksDependency           (36), (rupture de dépendance)
      otherError                 (99) } (autre erreur)

    VendorLoadErrorCode ::= ENTIER
```

Les champs du type FirmwarePackageLoadError ont la signification suivante :

version est un entier, et il donne le numéro de version de syntaxe pour la compatibilité avec de futures révisions de la présente spécification. Les mises en œuvre qui se conforment à la présente spécification DOIVENT régler la version à la valeur par défaut, qui est v1.

hwType est un identifiant d'objet qui identifie le type de module de matériel sur lequel le chargement du paquetage de microcode a été tenté.

hwSerialNum est le numéro de série du module de matériel sur lequel le chargement du paquetage de microcode a été tenté. Aucune structure particulière n'est imposée au numéro de série ; il n'est pas nécessaire qu'il soit un entier. Cependant, la combinaison de hwType et hwSerialNum identifie de façon univoque le module de matériel.

errorCode identifie l'erreur qui s'est produite.

vendorErrorCode est facultatif ; cependant, il DOIT être présent si le errorCode contient une valeur de otherError. Lorsque errorCode contient une valeur autre que otherError, le vendorErrorCode peut fournir des informations spécifiques du fabricant supplémentaires.

fwPkgName est facultatif. Lorsque il est présent, il identifie le paquetage de microcode qui était en cours de chargement lorsque l'erreur s'est produite. Comme décrit au paragraphe 2.2.3, deux approches sont prises en charge pour désigner les paquetages de microcode : héritée et préférée. Un nom hérité de paquetage de microcode est une chaîne d'octets. Un nom préféré de paquetage de microcode est une combinaison de l'identifiant d'objet du paquetage de microcode et d'un entier représentant le numéro de version.

config identifie la configuration actuelle du microcode. Le champ est FACULTATIF, mais la prise en charge de ce champ est RECOMMANDÉE pour les modules de matériel qui permettent le chargement de plus d'un paquetage de microcode. Une instance de CurrentFWConfig est utilisée pour fournir des informations sur chaque paquetage de microcode dans le module de matériel.

Les champs du type CurrentFWConfig ont la signification suivante :

fwPkgType identifie le type de paquetage de microcode. Le type de paquetage de microcode est un ENTIER, et la signification de la valeur de l'entier est spécifique de chaque module de matériel.

fwPkgName identifie le paquetage de microcode. Comme décrit au paragraphe 2.2.3, deux approches de dénomination des paquetages de microcode sont prises en charge : héritée et préférée. Un nom de paquetage de microcode hérité est une chaîne d'octets. Un nom préféré de paquetage de microcode est une combinaison de l'identifiant d'objet du paquetage de microcode et d'un entier représentant le numéro de version.

Les valeurs de errorCode ont la signification suivante :

decodeFailure : le décodage ASN.1 du chargement du paquetage de microcode a échoué. L'entrée fournie ne se conformait pas aux BER, ou n'était pas du tout de l'ASN.1.

BadContentInfo : syntaxe de ContentInfo invalide, ou le contentType porté au sein du ContentInfo est inconnu ou non pris en charge.

BadSignedData : syntaxe de SignedData invalide, la version est inconnue ou non prise en charge, ou plus d'une entrée sont présentes dans digestAlgorithms.

BadEncapContent : syntaxe de EncapsulatedContentInfo invalide, ou le contentType porté au sein du eContentType est inconnu ou non pris en charge. Cette erreur peut être générée à cause de problèmes situés dans SignedData ou CompressedData.

BadCertificate : syntaxe invalide pour un ou plusieurs certificats dans CertificateSet.

BadSignerInfo : syntaxe de SignerInfo invalide, ou la version est inconnue ou non prise en charge.

BadSignedAttrs : syntaxe de signedAttrs invalide au sein de SignerInfo.

BadUnsignedAttrs : les unsignedAttrs au sein de SignerInfo contiennent un attribut autre que l'attribut wrapped-firmware-decryption-key, qui est le seul attribut non signé accepté par la présente spécification.

MissingContent : le eContent facultatif manque dans EncapsulatedContentInfo, qui est exigé dans la présente spécification. Cette erreur peut être générée par des problèmes situés dans SignedData ou CompressedData.

NoTrustAnchor : deux situations peuvent conduire à cette erreur. Dans un cas, le subjectKeyIdentifier n'identifie pas la clé publique d'une ancre de confiance ou d'un chemin de certification qui se termine par une ancre de confiance installée. Dans l'autre cas, le issuerAndSerialNumber n'identifie pas la clé publique d'une ancre de confiance ou d'un chemin de certification qui se termine sur une ancre de confiance installée.

NotAuthorized : le sid au sein de SignerInfo conduit à une ancre de confiance installée, mais cette ancre de confiance n'est pas un signataire autorisé du paquetage de microcode.

BadDigestAlgorithm : le digestAlgorithm dans SignerInfo ou SignedData est inconnu ou non pris en charge.

BadSignatureAlgorithm : le signatureAlgorithm dans SignerInfo est inconnu ou non pris en charge.

UnsupportedKeySize : le signatureAlgorithm dans SignerInfo est connu et pris en charge, mais la signature du paquetage de microcode n'a pas pu être validée parce qu'une taille de clé non prise en charge a été employée par le signataire.

SignatureFailure : le signatureAlgorithm dans SignerInfo est connu et pris en charge, mais la signature dans SignerInfo n'a pas pu être validée.

ContentTypeMismatch : le contentType porté au sein de eContentType ne correspond pas au type de contenu porté dans l'attribut signé.

BadEncryptedData : syntaxe EncryptedData invalide ; la version est inconnue ou non prise en charge.

UnprotectedAttrsPresent : EncryptedData contient des unprotectedAttrs, ce qui n'est pas permis dans la présente spécification.

BadEncryptContent : syntaxe EncryptedContentInfo invalide, ou le contentType porté au sein de contentType est inconnu ou non pris en charge.

BadEncryptAlgorithm : l'algorithme de chiffrement de microcode identifié par contentEncryptionAlgorithm dans EncryptedContentInfo est inconnu ou non pris en charge.

MissingCiphertext : le encryptedContent facultatif manque dans EncryptedContentInfo, alors qu'il est exigé dans la présente spécification.

NoDecryptKey : le module de matériel n'a pas désigné de clé de déchiffrement de microcode dans l'attribut signé d'identifiant de clé de déchiffrement.

DecryptFailure : le paquetage de microcode ne s'est pas déchiffré correctement.

BadCompressAlgorithm : l'algorithme de compression identifié par compressionAlgorithm dans CompressedData est inconnu ou non pris en charge.

MissingCompressedContent : le eContent facultatif manque dans EncapsulatedContentInfo, alors qu'il est exigé dans la présente spécification.

DecompressFailure : le paquetage de microcode ne s'est pas décompressé correctement.

wrongHardware: le module de matériel traitant ne figure pas sur la liste des identifiants de module de matériel cible de l'attribut signé.

StalePackage : le paquetage de microcode est rejeté parce qu'il est périmé.

NotInCommunity : le module de matériel n'est pas membre de la communauté décrite dans les identifiants de communauté de l'attribut signé.

UnsupportedPackageType : le type de paquetage de microcode identifié dans les informations de paquetage de microcode de l'attribut signé n'est pas pris en charge par la combinaison de module de matériel et de chargeur d'amorçage.

MissingDependency : le paquetage de microcode à charger dépend de sous programmes qui font partie d'un autre paquetage de microcode, mais ce paquetage de microcode n'est pas disponible.

WrongDependencyVersion : le paquetage de microcode à charger dépend de sous programmes qui font partie d'un autre paquetage de microcode, et la version disponible de ce paquetage a un numéro de version qui est plus ancien que ce qui est exigé. Le paquetage de microcode disponible ne satisfait pas aux dépendances.

InsufficientMemory : le paquetage de microcode n'a pas pu être chargé parce que le module de matériel n'a pas une

mémoire suffisante.

BadFirmware : la signature sur le paquetage de microcode a été validée, mais le paquetage de microcode lui-même était d'un format non acceptable. Les détails vont être spécifiques de chaque module de matériel. Par exemple, un module de matériel composé de multiples composants de microcode programmable n'a pas pu trouver l'étiquetage interne au sein du paquetage de microcode pour distribuer le code exécutable à chaque composant.

UnsupportedParameters : la signature sur le paquetage de microcode n'a pas pu être validée parce que le signataire a utilisé des paramètres d'algorithme de signature qui ne sont pas pris en charge par le sous programme de vérification de signature de module de matériel.

BreaksDependency : un autre paquetage de microcode a une dépendance qui n'est plus satisfaite si le paquetage de microcode à charger est acceptée.

OtherError : une erreur s'est produite qui ne correspond à aucun des codes d'erreur précédents.

4.2 Attributs signés

Le module de matériel DOIT signer numériquement une collection d'attributs avec le rapport d'erreur de chargement du paquetage de microcode. Chaque attribut de la collection DOIT être codé en DER [X.509-88]. La syntaxe des attributs est définie dans la [RFC3852], et elle a été répétée au paragraphe 2.2.

Chacun des attributs utilisés avec ce profil a une seule valeur d'attribut, même si la syntaxe est définie comme ENSEMBLE DE AttributeValue. Il DOIT y avoir exactement une instance de AttributeValue présente.

La syntaxe de SignedAttributes au sein de signerInfo est définie comme un ENSEMBLE DE Attributes. SignedAttributes DOIT inclure seulement une instance de tout attribut particulier.

Le module de matériel DOIT inclure les attributs content-type et message-digest. Si le module de matériel inclut une horloge en temps réel, il DEVRAIT aussi inclure l'attribut signing-time. Le module de matériel PEUT inclure tout autre attribut estimé approprié.

4.2.1 Type de contenu

Le module de matériel DOIT inclure un attribut content-type avec la valeur de id-ct-firmwareLoadError (1.2.840.113549.1.9.16.1.18). Le paragraphe 11.1 de la [RFC3852] définit l'attribut content-type.

4.2.2 Résumé de message

Le module de matériel DOIT inclure un attribut message-digest, ayant comme valeur le résumé de message du contenu de FirmwarePackageLoadError. Le paragraphe 11.2 de la [RFC3852] définit l'attribut message-digest.

4.2.3 Heure de signature

Si le module de matériel comporte une horloge en temps réel, le module de matériel DEVRAIT inclure un attribut signing-time, spécifiant l'heure à laquelle le rapport d'erreur de chargement de paquetage de microcode a été généré. Le paragraphe 11.3 de la [RFC3852] définit l'attribut signing-time.

5. Nom de module de matériel

La prise en charge des récépissés de chargement de paquetage de microcode, telle que discutée à la Section 3, est FACULTATIVE, et la prise en charge des rapports d'erreur de chargement de paquetage de microcode, telle que discutée à la Section 4, est FACULTATIVE. Les modules de matériel qui prennent en charge la génération de récépissés ou de rapports d'erreur DOIVENT avoir des numéros de série univoques. De plus, les modules de matériel qui prennent en charge la génération de récépissés ou rapports d'erreur signés DOIVENT avoir les clés de signature privées et les certificats de validation de signature correspondants [RFC3280] ou leur désignateurs. Les conventions de désignation de module de matériel dans les certificats de validation de signature sont spécifiées dans cette section.

Le fabricant de module de matériel ou un tiers de confiance DOIT produire le certificat de validation de signature avant le

déploiement du module de matériel. Le certificat sera probablement produit au moment de la fabrication. Le nom de remplacement de sujet dans ce certificat identifie le module de matériel. Le nom distinctif du sujet est vide, mais une extension critique de nom de remplacement de sujet contient le nom de module de matériel, en utilisant le choix otherName au sein de la structure GeneralName.

La forme du nom du module de matériel est identifiée par l'identifiant d'objet id-on-hardwareModuleName :

```
IDENTIFIANT D'OBJET id-on-hardwareModuleName ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5)
                                                   mechanisms(5) pkix(7) on(8) 4 }
```

Un HardwareModuleName se compose d'un identifiant d'objet et d'une chaîne d'octets :

```
HardwareModuleName ::= SEQUENCE {
     hwType IDENTIFIANT D'OBJET,
     hwSerialNum CHAINE D'OCTETS }
```

Les champs du type HardwareModuleName ont la signification suivante :

hwType est un identifiant d'objet qui identifie le type de module de matériel. Un identifiant d'objet univoque désigne un modèle de matériel et ses révisions.

hwSerialNum est le numéro de série du module de matériel. Aucune structure particulière n'est imposée au numéro de série ; il n'est pas nécessairement un entier. Cependant, la combinaison de hwType et de hwSerialNum identifie de façon univoque le module de matériel.

6. Considérations sur la sécurité

Le présent document décrit l'utilisation de la syntaxe de message cryptographique (CMS) pour protéger les paquetages de microcode ; donc, les considérations sur la sécurité discutées dans la [RFC3852] s'appliquent aussi à la présente spécification.

Les conventions spécifiées dans ce document soulèvent quelques considérations de sécurité particulières.

6.1 Clés et algorithmes de chiffrement

Les clés de signature privées doivent être protégées. La compromission de la clé privée utilisée pour signer les paquetages de microcode permet à des parties non autorisées de générer des paquetages de microcode qui soient acceptables aux modules de matériel. La compromission de la clé privée du module de matériel permet à des parties non autorisées de générer des récépissés de chargement de paquetage de microcodes et des rapports d'erreur.

La clé de déchiffrement de microcode doit être protégée. La compromission de la clé peut résulter en la divulgation du paquetage de microcode à des parties non autorisées.

Les algorithmes de chiffrement s'affaiblissent avec le temps. Avec le développement de nouvelles techniques de cryptanalyse et l'amélioration des performances de calcul, le facteur travail pour casser un algorithme de chiffrement particulier se réduit. La capacité de changer le paquetage de microcode fournit une opportunité de mettre à jour ou remplacer les algorithmes de chiffrement. Bien que cette capacité soit désirable, le remplacement d'un algorithme de chiffrement peut conduire à des échecs d'interopérabilité. Donc, la sortie de nouveaux algorithmes de chiffrement doit être gérée. Généralement, il faut que la précédente génération d'algorithmes de chiffrement et leurs remplaçants soit pris en charge simultanément afin de faciliter une transition harmonieuse.

6.2 Génération de nombres aléatoires

Lorsque les paquetages de microcode sont chiffrés, la source du paquetage de microcode doit générer de façon aléatoire les clés de chiffrement du microcode. Aussi, la génération de paires de clés de signature publique/privée repose sur des nombres aléatoires. L'utilisation de générateurs de nombres pseudo aléatoires inadéquats (PRNG) pour générer des clés de chiffrement peut résulter en peu, ou pas du tout, de sécurité. Un attaquant peut trouver beaucoup plus facile de reproduire l'environnement de PRNG qui a produit les clés, et de chercher dans le petit ensemble de possibilités résultant, que de chercher en force brute dans tout l'espace de clés. La génération de nombres aléatoires de qualité est difficile. La [RFC4086] offre d'importantes lignes directrices dans ce domaine.

6.3 Numéro de version de paquetage de logiciel périmé

Le signataire du microcode détermine si un numéro de version périmée est inclus. La politique du signataire du microcode doit considérer de nombreux facteurs. Considérons la faille trouvée par Ian Goldberg et David Wagner dans le générateur de nombres aléatoires du navigateur Netscape en 1996 [DDJ]. Cette faille ruinait complètement la protection de la confidentialité. Un signataire du microcode pourrait utiliser le numéro de version périmée pour s'assurer que les modules de matériel mis à niveau ne reviennent pas à l'utilisation du microcode fautif. Cependant, un autre signataire du microcode peut considérer que ce n'est pas une situation appropriée pour employer le numéro de version périmée, préférant déléguer cette décision à quelqu'un plus proche du fonctionnement du module de matériel. Une telle personne sera probablement dans une meilleure position pour évaluer si d'autres fautes introduites dans le nouveau paquetage de microcode imposent de pire soucis de fonctionnement que ceux de confidentialité causés par la faille du générateur de nombres aléatoire. Par exemple, un usager qui ne se sert jamais du dispositif de chiffrement du navigateur Netscape fautif va déterminer la version la plus appropriée à utiliser sans considérer la faille des nombres aléatoires ni son remède.

Le numéro de version périmée est particulièrement utile lorsque les intérêts de sécurité de la personne qui choisit quelle version de paquetage de microcode charger dans un certain module de matériel ne se recouvrent pas avec ceux du signataire du paquetage de microcode. Par exemple, les numéros de version périmée peuvent être utiles dans des modules de matériel qui fournissent la gestion des droits numériques (DRM, *digital rights management*). Aussi, les numéros de version périmée seront utiles lorsque l'organisation utilisatrice (par opposition au fabricant du paquetage de microcode) est le signataire du microcode. De plus, les numéros de version périmée seront utiles pour les paquetages de microcode qui doivent être de confiance pour mettre en œuvre une politique de sécurité d'organisation (par opposition à l'organisation de déploiement) sans considérer si le signataire du microcode est l'organisation de déploiement ou le fabricant. Par exemple, des matériels utilisés par les militaires vont probablement utiliser les numéros de version périmée.

L'utilisation d'un numéro de version périmée dans un paquetage de microcode qui emploie la forme de nom préféré de paquetage de microcode ne peut pas complètement empêcher l'utilisation ultérieure d'un paquetage de microcode périmé. En dépit de ce défaut, ce dispositif est inclus car il est utile dans certaines situations importantes. En chargeant différents types de paquetages de microcode, chacun avec son propre numéro de version périmé de paquetage de microcode jusqu'à ce que la capacité de mémorisation interne pour les numéros de version périmée soit dépassée, l'utilisateur peut circonvenir le mécanisme. Considérons un module de matériel qui a la mémorisation pour deux numéros de version périmée. Supposons que FWPKG-A version 3 soit chargé, indiquant que FWPKG-A version 2 est périmé. L'utilisateur peut charger séquentiellement ce qui suit :

- FWPKG-B version 8, indiquant que FWPKG-B version 4 est périmé. (Note : la mémorisation interne indique que FWPKG-A version 2 et FWPKG-B version 4 sont périmées.)

- FWPKG-C version 5, indiquant que FWPKG-C version 3 est périmé. (Note : la mémorisation interne indique que FWPKG-B version 4 et FWPKG-C version 3 sont périmées.)

- FWPKG-A version 2.

Parce que de nombreux modules de matériel sont supposés avoir très peu de paquetages de microcode écrits pour eux, le dispositif de version périmée de paquetage de microcode fournit une protection importante. La quantité de mémoire non volatile qui doit être dédiée à la sauvegarde des identifiants et numéros de version de paquetage de microcode dépend du nombre de paquetages de microcode qui seront vraisemblablement développés pour le module de matériel.

L'utilisation de la forme de nom hérité de paquetage de microcode n'améliore pas cette situation. En fait, les noms hérités de paquetage de microcode sont généralement plus longs qu'un identifiant d'objet. Donc, la protection des versions périmées comparables exige plus de mémoire.

Un signataire du microcode peut s'assurer que les numéros de version périmée sont respectés en limitant le nombre de différents types de paquetages de microcode qui sont signés. Si tous les modules de matériel sont capables de mémoriser un numéro de version périmée pour chacun des différents types de paquetage de microcode, alors le module de matériel va être capable de fournir la protection désirée. Cela exige que le signataire du microcode ait une profonde compréhension de tous les modules de matériel qui pourraient accepter le paquetage de microcode.

6.4 Identifiants de communauté

Lorsque un paquetage de microcode inclut un identifiant de communauté, la confiance que le paquetage n'est utilisé que par la communauté prévue dépend du mécanisme utilisé pour configurer l'appartenance à la communauté. Le présent document

ne spécifie pas de mécanisme pour l'attribution de la qualité de membre d'une communauté aux modules de matériel, et les diverses solutions ont des propriétés de sécurité différentes. Aussi, l'autorité qui fait les allocations d'identifiant de communauté aux modules de matériel peut être différente de l'autorité qui génère les paquetages de microcode.

7. Références

7.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC2634] P. Hoffman, éd., "Services de sécurité améliorés pour S/MIME", juin 1999. (*MàJ par* RFC5035) *(P.S.)*

[RFC3274] P. Gutmann, "Type de contenu Données compressées pour la syntaxe de message cryptographique (CMS)", juin 2002. *(P.S.)*

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

[RFC3629] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", STD 63, novembre 2003.

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

[SHA1] National Institute of Standards and Technology. FIPS Pub 180-1, "Secure Hash Standard". 17 avril 1995.

[X.208-88] Recommandation UIT-T X.208, "Spécification de la notation n° 1 de syntaxe abstraite (ASN.1)". 1988.

[X.209-88] Recommandation UIT-T X.209, "Spécification des règles de codage de base pour la notation n° 1 de syntaxe abstraite (ASN.1)". 1988.

[X.509-88] Recommandation UIT-T X.509, "L'annuaire – cadre d'authentification". 1988.

7.2 Références pour information

[AES] National Institute of Standards and Technology. FIPS Pub 197: "Advanced Encryption Standard (AES)". 26 novembre 2001.

[DDJ Goldberg, I. and D. Wagner. "Randomness and the Netscape Browser". Dr. Dobb's Journal, janvier 1996.

[PKCS#6] RSA Laboratories. " PKCS #6: Extended-Certificate Syntax Standard, Version 1.5". novembre 1993.

[RFC2560] M. Myers, R. Ankney, A. Malpani, S. Galperin et C. Adams, "Protocole d'état de certificat en ligne d'infrastructure de clé publique X.509 pour l'Internet - OCSP", juin 1999. *(P.S.) (Remplacée par* RFC6960)

[RFC3281] S. Farrell et R. Housley, "Profil de certificat d'attribut Internet pour l'autorisation", avril 2002. *(Obsolète, voir* RFC5755)

[RFC3379] D. Pinkas, R. Housley, "Exigences pour le protocole de validation de chemin délégué et de découverte de chemin délégué," septembre 2002. *(Information)*

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

[SECREQMTS] National Institute of Standards and Technology. FIPS Pub 140-2, "Security Requirements for Cryptographic Modules". 25 mai 2001.

[X.509-97] Recommandation UIT-T X.509, "L'annuaire – cadre d'authentification". 1997.

[X.509-00] Recommandation UIT-T X.509, "L'annuaire – cadre d'authentification". 2000.

Appendice A Module ASN.1

Le module ASN.1 contenu dans cet appendice définit les structures qui sont nécessaires pour mettre en œuvre l'enveloppe de paquetage de microcode fondé sur la CMS. Il est prévu qu'il soit utilisé en conjonction avec les modules ASN.1 des [RFC3852], [RFC3274], et [RFC3280].

```
IDENTIFIANT D'OBJET CMSFirmwareWrapper ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
                                             smime(16) modules(0) cms-firmware-wrap(22) }

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::= DÉBUT

IMPORTE
      EnvelopedData
      DE CryptographicMessageSyntax -- [RFC3852]
         { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) };

-- Type de contenu et identifiant d'objet de paquetage de microcode

IDENTIFIANT D'OBJET id-ct-firmwarePackage ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                                smime(16) ct(1) 16 }

FirmwarePkgData ::= CHAINE D'OCTETS

-- Attributs signés et identifiant d'objet de paquetage de microcode

IDENTIFIANT D'OBJET id-aa-firmwarePackageID ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                                  smime(16) aa(2) 35 }

FirmwarePackageIdentifier ::= SEQUENCE {
    name PreferredOrLegacyPackageIdentifier,
    stale PreferredOrLegacyStalePackageIdentifier FACULTATIF }

PreferredOrLegacyPackageIdentifier ::= CHOIX {
    preferred PreferredPackageIdentifier,
    legacy CHAINE D'OCTETS }

PreferredPackageIdentifier ::= SEQUENCE {
    fwPkgID IDENTIFIANT D'OBJET,
    verNum ENTIER (0..MAX) }

PreferredOrLegacyStalePackageIdentifier ::= CHOIX {
    preferredStaleVerNum ENTIER (0..MAX),
    legacyStaleVersion CHAINE D'OCTETS }

IDENTIFIANT D'OBJET id-aa-targetHardwareIDs ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                                  smime(16) aa(2) 36 }

TargetHardwareIdentifiers ::= SEQUENCE DE IDENTIFIANT D'OBJET

IDENTIFIANT D'OBJET id-aa-decryptKeyID ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                             smime(16) aa(2) 37 }

DecryptKeyIdentifier ::= CHAINE D'OCTETS

IDENTIFIANT D'OBJET id-aa-implCryptoAlgs ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                               smime(16) aa(2) 38 }

ImplementedCryptoAlgorithms ::= SEQUENCE DE IDENTIFIANT D'OBJET

IDENTIFIANT D'OBJET  id-aa-implCompressAlgs ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
```

```
                                                    smime(16) aa(2) 43 }

ImplementedCompressAlgorithms ::= SEQUENCE DE IDENTIFIANT D'OBJET

IDENTIFIANT D'OBJET id-aa-communityIdentifiers ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                                                      pkcs9(9) smime(16) aa(2) 40 }

CommunityIdentifiers ::= SEQUENCE DE CommunityIdentifier

CommunityIdentifier ::= CHOIX {
    communityOID IDENTIFIANT D'OBJET,
    hwModuleList HardwareModules }

HardwareModules ::= SEQUENCE {
    hwType IDENTIFIANT D'OBJET,
    hwSerialEntries SEQUENCE DE HardwareSerialEntry }

HardwareSerialEntry ::= CHOIX {
    all NULL,
    single CHAINE D'OCTETS,
    block SEQUENCE {
      low CHAINE D'OCTETS,
      high CHAINE D'OCTETS } }

IDENTIFIANT D'OBJET id-aa-firmwarePackageInfo ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                                                      pkcs9(9) smime(16) aa(2) 42 }

FirmwarePackageInfo ::= SEQUENCE {
    fwPkgType ENTIER FACULTATIF,
    dependencies SEQUENCE DE
      PreferredOrLegacyPackageIdentifier FACULTATIF }

-- Attributs non signés et identifiant d'objet de paquetage de microcode

IDENTIFIANT D'OBJET id-aa-wrappedFirmwareKey ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                                                      pkcs9(9) smime(16) aa(2) 39 }

WrappedFirmwareKey ::= EnvelopedData

-- Type de contenu de récépissé de chargement et identifiant d'objet de paquetage de microcode

IDENTIFIANT D'OBJET id-ct-firmwareLoadReceipt ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                                                      pkcs9(9) smime(16) ct(1) 17 }

FirmwarePackageLoadReceipt ::= SEQUENCE {
    version FWReceiptVersion DEFAULT v1,
    hwType IDENTIFIANT D'OBJET,
    hwSerialNum CHAINE D'OCTETS,
    fwPkgName PreferredOrLegacyPackageIdentifier,
    trustAnchorKeyID CHAINE D'OCTETS FACULTATIF,
    decryptKeyID [1] CHAINE D'OCTETS FACULTATIF }

FWReceiptVersion ::= ENTIER { v1(1) }

-- Type de contenu de rapport d'erreur de chargement et ientifiant d'objet de paquetage de microcode

IDENTIFIANT D'OBJET id-ct-firmwareLoadError ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
                                                      smime(16) ct(1) 18 }

FirmwarePackageLoadError ::= SEQUENCE {
    version FWErrorVersion DEFAUT v1,
    hwType IDENTIFIANT D'OBJET,
    hwSerialNum CHAINE D'OCTETS,
```

```
    errorCode FirmwarePackageLoadErrorCode,
    vendorErrorCode VendorLoadErrorCode FACULTATIF,
    fwPkgName PreferredOrLegacyPackageIdentifier FACULTATIF,
    config [1] SEQUENCE DE CurrentFWConfig FACULTATIF }

FWErrorVersion ::= ENTIER { v1(1) }

CurrentFWConfig ::= SEQUENCE {
    fwPkgType ENTIER FACULTATIF,
    fwPkgName PreferredOrLegacyPackageIdentifier }

FirmwarePackageLoadErrorCode ::= ENUMERATED {
    decodeFailure                (1),
    badContentInfo               (2),
    badSignedData                (3),
    badEncapContent              (4),
    badCertificate               (5),
    badSignerInfo                (6),
    badSignedAttrs               (7),
    badUnsignedAttrs             (8),
    missingContent               (9),
    noTrustAnchor                (10),
    notAuthorized                (11),
    badDigestAlgorithm           (12),
    badSignatureAlgorithm        (13),
    unsupportedKeySize           (14),
    signatureFailure             (15),
    contentTypeMismatch          (16),
    badEncryptedData             (17),
    unprotectedAttrsPresent      (18),
    badEncryptContent            (19),
    badEncryptAlgorithm          (20),
    missingCiphertext            (21),
    noDecryptKey                 (22),
    decryptFailure               (23),
    badCompressAlgorithm         (24),
    missingCompressedContent     (25),
    decompressFailure            (26),
    wrongHardware                (27),
    stalePackage                 (28),
    notInCommunity               (29),
    unsupportedPackageType       (30),
    missingDependency            (31),
    wrongDependencyVersion       (32),
    insufficientMemory           (33),
    badFirmware                  (34),
    unsupportedParameters        (35),
    breaksDependency             (36),
    otherError                   (99) }

VendorLoadErrorCode ::= ENTIER

-- Syntaxe des autres noms pour le nom de module de matériel

IDENTIFIANT D'OBJET id-on-hardwareModuleName ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5)
                                                   mechanisms(5) pkix(7) on(8) 4 }

HardwareModuleName ::= SEQUENCE {
    hwType IDENTIFIANT D'OBJET,
    hwSerialNum CHAINE D'OCTETS }

FIN
```

Adresse de l'auteur

Russell Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA

mél : housley@vigilsec.com

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par la Internet Society.