Groupe de traavail Réseau
**Request for Comments : 3657**
Catégorie: Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

S. Moriai, Sony Computer Entertainment Inc.
A. Kato, NTT Software Corporation
janvier 2004

# Utilisation de l'algorithme de chiffrement Camellia dans la syntaxe de message cryptographique (CMS)



**Résumé**
Le présent document spécifie les conventions pour l'utilisation de l'algorithme de chiffrement Camellia pour un chiffrement avec la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*).

## 1. Introduction

Le présent document spécifie les conventions pour l'utilisation de l'algorithme de chiffrement Camellia [CamelliaSpec] pour un chiffrement avec la syntaxe de message cryptographique (CMS, *Cryptographic Message Syntax*) [RFC3369]. Les identifiants d'objet (OID, *object identifier*) pertinents et les étapes de traitement sont fournis afin que Camellia puisse être utilisé dans la spécification de CMS [RFC3369], [RFC3370] pour le chiffrement du contenu et des clés.

Note : Ce travail a été effectué alors que le premier auteur travaillait pour NTT.

### 1.1 Camellia

Camellia a été développé conjointement par Nippon Telegraph and Telephone Corporation et Mitsubishi Electric Corporation en 2000. Camellia spécifie la taille de bloc de 128 bits et les tailles de clé de 128, 192, et 256 bits, et la même interface que la norme de chiffrement évoluée (AES, *Advanced Encryption Standard*). Camellia se caractérise par sa convenance pour les mises en œuvre de logiciel et de matériel aussi bien que par son haut niveau de sécurité. D'un point de vue pratique, il est conçu pour apporter de la souplesse aux mises en œuvre de matériel et de logiciel sur les processeurs de 32 bits largement utilisés sur l'Internet et sur de nombreuses applications, sur les processeurs de 8 bits utilisés dans les cartes à mémoire, le matériel cryptographique, les systèmes incorporés, et ainsi de suite [CamelliaTech]. De plus, son délai d'établissement de clé est excellent, et son agilité de clé est supérieure à celle de AES.

Camellia a été examiné par toute la communauté de la cryptographie lors de plusieurs projets pour évaluer les algorithmes de chiffrement. En particulier, Camellia a été choisi comme primitive cryptographique recommandée par le projet de nouveaux schémas européens pour les signatures, l'intégrité et le chiffrement (NESSIE, *New European Schemes for Signatures, Integrity and Encryption*) [NESSIE] de l'Union Européenne et aussi inclus dans la liste des techniques de chiffrement pour les systèmes du gouvernement japonais qui ont été choisies par le comité de recherche et d'évaluation cryptographique (CRYPTREC, *Cryptography Research and Evaluation Committee*) [CRYPTREC] du Japon.

### 1.2 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 le BCP 14, [RFC2119].

## 2. Identifiants d'objet pour le chiffrement de contenu et de clé

Cette section donne les OID et les informations de traitement nécessaires pour l'utilisation de Camellia au chiffrement de contenu et de clés dans la CMS.

Camellia est ajouté à l'ensemble des algorithmes facultatifs de chiffrement symétriques dans la CMS en fournissant deux classes d'identifiants d'objet (OID, *Object Identifier*) uniques. Une classe d'OID définit les algorithmes de chiffrement de contenu et l'autre définit les algorithmes de chiffrement de clé. Donc, un agent de CMS peut appliquer Camellia pour le chiffrement de contenu ou pour le chiffrement de clé en choisissant l'identifiant d'objet correspondant, en fournissant les paramètres requis, et en lançant le code de programme.

### 2.1 OID pour le chiffrement de contenu

Camellia est ajouté à l'ensemble d'algorithmes de chiffrement de contenu symétriques définis dans la [RFC3370]]. L'algorithme de chiffrement de contenu Camellia, en mode de chaînage de bloc de chiffrement (CBC, *Cipher Block Chaining*) pour les trois tailles de clé différentes est identifié par les identifiants d'objet suivants :

IDENTIFIANT D'OBJET id-camellia128-cbc ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) symmetric-encryption-algorithm(1) camellia128-cbc(2) }

IDENTIFIANT D'OBJET id-camellia12-cbc ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) symmetric-encryption-algorithm(1) camellia192-cbc(3) }

IDENTIFIANT D'OBJET id-camellia256-cbc ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) symmetric-encryption-algorithm(1) camellia256-cbc(4) }

Le champ de paramètres AlgorithmIdentifier DOIT être présent, et le champ Paramètres DOIT contenir la valeur d'IV :

CamelliaCBCParameter ::= CamelliaIV -- valeur d'initialisation

CamelliaIV ::= CHAÎNE D'OCTETS (TAILLE(16))

Le texte source est bourré conformément au paragraphe 6.3 de la [RFC3369].

### 2.2 OID pour le chiffrement de clés

Les procédures d'enveloppement/désenveloppement de clé utilisées pour chiffrer/déchiffrer une clé de chiffrement de contenu (CEK, *content-encryption key*) Camellia avec une clé de chiffrement de clé (KEK, *key-encryption key*) Camellia sont spécifiées à la Section 3. La génération et la distribution des clés de chiffrement de clé sortent du domaine d'application du présent document.

L'algorithme de chiffrement de clé Camellia a l'identifiant d'objet suivant :

IDENTIFIANT D'OBJET id-camellia128-wrap ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) key-wrap-algorithm(3) camellia128-wrap(2) }

IDENTIFIANT D'OBJET id-camellia192-wrap ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) key-wrap-algorithm(3) camellia192-wrap(3) }

IDENTIFIANT D'OBJET id-camellia256-wrap ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) key-wrap-algorithm(3) camellia256-wrap(4) }

Dans tous les cas, le champ Paramètres de AlgorithmIdentifier DOIT être ABSENT, parce que la procédure d'enveloppement de clé définit elle-même comment et quand utiliser une IV. L'OID donne la taille de la clé de KEK, mais ne fait aucune déclaration sur la taille de la CEK Camellia enveloppée. Les mises en œuvre PEUVENT utiliser des tailles de KEK et de CEK différentes. Les mises en œuvre DOIVENT prendre en charge une CEK et une KEK de même longueur. Si des longueurs différentes sont prises en charge, la KEK DOIT être d'une longueur égale ou supérieure à celle de la CEK.

## 3. Algorithme d'enveloppement de clé

L'enveloppement et le désenveloppement de clé Camellia sont faits conformément à l'algorithme AES d'enveloppement de clé [RFC3394], parce que Camellia et AES ont les mêmes tailles de bloc et de clé, c'est-à-dire une taille de bloc de 128 bits et des tailles de clé de 128, 192, et 256 bits.

### 3.1 Notation et définitions

La notation suivante est utilisée dans la description des algorithmes d'enveloppement de clé :

Camellia(K, W) : chiffre W en utilisant le livre de code Camellia avec la clé K.
Camellia-1(K, W) : déchiffre W en utilisant le livre de code Camellia avec la clé K.
MSB(j, W) : retourne les j bits de poids fort de W.
LSB(j, W) : retourne les j bits de moindre poids de W.
B1 ^ B2 : l'ajout bit par bit exclusif ou (OUXé) de B1 et B2.
B1 | B2 : enchaînement de B1 et B2.
K : clé de chiffrement de clé K.
n : nombre de blocs de données de clé de 64 bits.
s : nombre d'étapes dans le processus d'enveloppement , s = 6n.
P[i]: ième bloc de données de clé en texte source.
C[i] : ième bloc de données de texte chiffré.
A : registre de vérification d'intégrité de 64 bits.
R[i] : matrice de registres de 64 bit s où i = 0, 1, 2, ..., n.
A[t], R[t][i] : contenu des registres A et R[i] après l'étape de chiffrement t.
IV : valeur initiale de 64 bits utilisée durant le processus d'enveloppement.

Dans l'algorithme d'enveloppement de clé, la fonction d'enchaînement va être utilisée pour enchaîner des quantités de 64 bits pour former l'entrée de 128 bits du livre de code Camellia. Les fonctions d'extraction seront utilisées pour partager le résultat de 128 bit s provenant du livre de code Camellia en deux quantités de 64 bits.

### 3.2 Enveloppement de clé Camellia

L'enveloppement de clé avec Camellia est identique au paragraphe 2.2.1 de la [RFC3394] avec "AES" remplacé par "Camellia".

Les entrées au processus d'enveloppement de clé sont la KEK et le texte source à envelopper. Le texte source consiste en n blocs de 64 bits, contenant les données de clé qui vont être enveloppées. Le processus d'enveloppement de clé est décrit ci-dessous.

Entrées : texte source, n valeurs de 64 bits {P[1], P[2], ..., P[n]}, et la clé, K (la KEK).
Résultats : texte chiffré, (n+1) valeurs de 64 bits {C[0], C[1], ..., C[n]}.

1) Initialiser les variables.
Régler A[0] à une valeur initiale (voir au paragraphe 3.4)
Pour i = 1 à n
R[0][i] = P[i]

2) Calculer les valeurs intermédiaires.
Pour t = 1 à s, où s = 6n
A[t] = MSB(64, Camellia(K, A[t-1] | R[t-1][1])) ^ t
Pour i = 1 à n-1
R[t][i] = R[t-1][i+1]
R[t][n] = LSB(64, Camellia(K, A[t-1] | R[t-1][1]))

3) Sortir les résultats.
Régler C[0] = A[t]
Pour i = 1 à n

C[i] = R[t][i]

Une autre description de l'algorithme d'enveloppement de clé implique une indexation plutôt qu'un glissement. Cette approche permet de calculer la clé enveloppée à la place, évitant la rotation de la description précédente. Cela produit un résultat identique et est plus facilement mis en œuvre par le logiciel.

Entrées : texte source, n valeurs de 64 bits {P[1], P[2], ..., P[n]}, et la clé K (la KEK).
Sorties : texte chiffré, (n+1) valeurs de 64 bits {C[0], C[1], ..., C[n]}.

1) Initialiser les variables.
Régler A = IV, une valeur initiale (voir au paragraphe 3.4)
Pour i = 1 à n
R[i] = P[i]

2) Calculer les valeurs intermédiaires.
Pour j = 0 à 5
Pour i = 1 à n
B = Camellia(K, A | R[i])
A = MSB(64, B) ^ t où t = (n*j)+i
R[i] = LSB(64, B)

3) Sortir les résultats.
Régler C[0] = A
Pour i = 1 à n
C[i] = R[i]

### 3.3 Désenveloppement de clé Camellia

Le désenveloppement de clé avec Camellia est identique au processus du paragraphe 2.2.2 de la [RFC3394], avec "AES" remplacé par "Camellia".

Les entrées au processus de désenveloppement sont la KEK et (n+1) blocs de 64 bits de texte chiffré consistant en la clé enveloppée précédemment. Cela retourne n blocs de texte source consistant en les n blocs de 64 bits de données de la clé déchiffrée.

Entrées : texte chiffré, (n+1) valeurs de 64 bits {C[0], C[1], ..., C[n]}, et la clé K (la KEK).
Sorties : texte source, n valeurs de 64 bits {P[1], P[2], ..., P[n]}.

1) Initialiser les variables.
Régler A[s] = C[0] où s = 6n
Pour i = 1 à n
R[s][i] = C[i]

2) Calculer les valeurs intermédiaires.
Pour t = s à 1
A[t-1] = MSB(64, Camellia-1(K, ((A[t] ^ t) | R[t][n]))
R[t-1][1] = LSB(64, Camellia-1(K, ((A[t]^t) | R[t][n]))
For i = 2 à n
R[t-1][i] = R[t][i-1]

3) Sortie des résultats.
Si A[0] est une valeur initiale appropriée (voir au paragraphe 3.4),
Alors
Pour i = 1 à n
P[i] = R[0][i]
Autrement
Retourner une erreur

L'algorithme de désenveloppement peut aussi être spécifié comme une opération fondée sur un indice, permettant que les calculs soient effectués à la place. Là encore, cela produit le même résultat que l'approche du glissement de registre.

Entrées : texte chiffré, (n+1) valeurs de 64 bits {C[0], C[1], ..., C[n]}, et la clé K (la KEK).
Sorties : texte source, n valeurs de 64 bits {P[0], P[1], ..., P[n]}.

1) Initialiser les variables.
Régler A = C[0]
Pour i = 1 à n
R[i] = C[i]

2) Calculer les valeurs intermédiaires.
Pour j = 5 à 0
Pour i = n à 1
B = Camellia-1(K, (A ^ t) | R[i]) où t = n*j+i
A = MSB(64, B)
R[i] = LSB(64, B)

3) Sortie des résultats.
Si A est une valeur initiale appropriée (voir le paragraphe 3.4),
Alors
Pour i = 1 à n, P[i] = R[i]
Autrement
Retourner une erreur.

### 3.4 Intégrité des données de clé -- valeur initiale

La valeur initiale (IV) se réfère à la valeur allouée à A[0] dans la première étape du processus d'enveloppement. Cette valeur est utilisée pour obtenir une vérification d'intégrité sur les données de clé. Dans l'étape finale du processus de désenveloppement, la valeur récupérée de A[0] est comparée à la valeur de A[0] attendue. Si il y a correspondance, la clé est acceptée comme valide, et l'algorithme de désenveloppement la retourne. Si elle ne correspond pas, la clé est alors rejetée, et l'algorithme de désenveloppement retourne une erreur.

Les propriétés exactes de cette vérification d'intégrité dépendent de la définition de la valeur initiale. Différentes applications peuvent invoquer des propriétés assez différentes ; par exemple, si il est besoin de déterminer l'intégrité des données de clé à travers son cycle de vie ou juste quand elles sont désenveloppées. La présente spécification définit une valeur initiale par défaut qui prend en charge l'intégrité des données de clé durant la période où elles sont enveloppées (au paragraphe 3.4.1). Des dispositions sont aussi prises pour prendre en charge des valeurs initiales de remplacement (paragraphe 3.4.2).

#### 3.4.1 Valeur initiale par défaut

La valeur initiale (IV) par défaut est définie comme étant la constante hexadécimale A[0] = IV = A6A6A6A6A6A6A6A6

L'utilisation d'une constante comme IV accepte une vérification d'intégrité forte sur les données de clé durant la période pendant laquelle elles sont enveloppées. Si le désenveloppement produit A[0] = A6A6A6A6A6A6A6A6, les chances que les données de clé soient corrompues sont alors de 2^-64. Si le désenveloppement produit toute autre valeur pour A[0] , le désenveloppement doit retourner une erreur et ne doit retourner aucune données de clé.

#### 3.4.2 Valeurs initiales de remplacement

Quand l'enveloppement de clé est utilisé au titre d'un plus large protocole ou système de gestion de clé, la portée désirée pour l'intégrité des données peut être plus que juste les données de clé ou la durée désirée plus que juste la période où elles sont enveloppées. Aussi, si les données de clé ne sont pas juste une clé Camellia, elles peuvent n'être pas toujours un multiple de 64 bits. D'autres définitions de la valeur initiale peuvent être utilisées pour traiter de tels problèmes. Selon la [RFC3394], le NIST va définir des valeurs initiales de remplacement dans de futures publications de gestion de clé en tant que de besoin. Afin de s'accommoder d'un ensemble de solutions de remplacement qui puisse évoluer dans le temps, les mises en œuvre d'enveloppement de clé qui ne sont pas spécifiques d'une application devront être assez souples quant à la façon dont la valeur initiale est établie et vérifiée.

## 4. Attribut SMIMECapabilities

Un client S/MIME DEVRAIT annoncer l'ensemble des fonctions cryptographiques qu'il prend en charge en utilisant l''attribut SMIMECapabilities *(capacités S/MIME)*. Cet attribut donne une liste partielle des OID des fonctions cryptographiques et DOIT être signé par le client. Les OID des fonctions DEVRAIENT être séparés logiquement par catégories fonctionnelles et DOIVENT être ordonnés par préférence.

Le paragraphe 2.5.2 de la [RFC2633] définit l'attribut signé SMIMECapabilities (défini comme séquence de SEQUENCES de SMIMECapability) comme étant utilisé pour spécifier une liste partielle d'algorithmes que peut prendre en charge le logiciel qui annonce les SMIMECapabilities.

Si il est demandé à un client S/MIME de prendre en charge le chiffrement symétrique avec Camellia, l'attribut de capacités DOIT contenir l'OID Camellia spécifié ci-dessus dans la catégorie des algorithmes symétriques. Le paramètre associé à cet OID DOIT être CamelliaSMimeCapability.

CamelliaSMimeCapabilty ::= NULL

La séquence SMIMECapability qui représente Camellia DOIT être codée en DER comme les chaînes hexadécimales suivantes :

| **Taille de clé** | **Capacité** |
|---|---|
| 128 | 30 0f 06 0b 2a 83 08 8c 9a 4b 3d 01 01 01 02 05 00 |
| 196 | 30 0f 06 0b 2a 83 08 8c 9a 4b 3d 01 01 01 03 05 00 |
| 256 | 30 0f 06 0b 2a 83 08 8c 9a 4b 3d 01 01 01 04 05 00 |

Quand un agent envoyeur crée un message chiffré, il doit décider quel type d'algorithme de chiffrement utiliser. En général le processus de décision implique des informations obtenues des listes de capacités incluses dans les messages reçus du receveur, ainsi que d'autres informations comme des accords privés, les préférences de l'utilisateur, des restrictions légales, et ainsi de suite. Si les utilisateurs exigent Camellia pour un chiffrement symétrique, il DOIT être pris en charge par les clients S/MIME des deux côtés envoyeur et receveur, et il DOIT être établi dans les préférences de l'utilisateur.

## 5. Considérations sur la sécurité

Le présent document spécifie l'utilisation de Camellia pour chiffrer le contenu d'un message de CMS et pour chiffrer la clé symétrique utilisée pour chiffrer le contenu d'un message de CMS, et les autres mécanismes sont les mêmes que ceux déjà existants. Donc, les considérations de sécurité décrites dans les spécifications de la CMS [RFC3369], [RFC3370] et l'algorithme d'enveloppement de clé AES [RFC3394] peuvent être appliquées au présent document.Aucun problème de sécurité n'a été trouvé sur Camellia [CRYPTREC], [NESSIE].

## 6. Déclaration de propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourrait être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org .

Des droits de propriété intellectuelle ont été notifiés à l'IETF concernant tout ou partie de la spécification contenue dans le présent document. Pour plus d'informations, consulter la liste en ligne des revendications de droits.

## 7. Références

### 7.1 Références normatives

[CamelliaSpec] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., and Tokita, T., "Specification of Camellia - a 128-bit Block Cipher". http://info.isl.ntt.co.jp/camellia/

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

[RFC2633] B. Rmasdell, "Spécification de message S/MIME version 3", juin 1999. (*Obsolète, voir* RFC3851) *(P.S.)*

[RFC3369] R. Housley, "Syntaxe de message cryptographique (CMS)", août 2002. (*Obsolète, voir* RFC3852) *(P.S.)*

[RFC3370] R. Housley, "Algorithmes de syntaxe de message cryptographique (CMS)", août 2002. *(P.S. ; MàJ par* RFC8702*)*

[RFC3394] J. Schaad, R. Housley, "Algorithme d'enveloppe de clés pour la norme de chiffrement évoluée (AES)", septembre 2002. *(Information)*

[RFC3565] J. Schaad, "Utilisation de l'algorithme de chiffrement de la norme de chiffrement évolué (AES) dans la syntaxe de message cryptographique (CMS)", juillet 2003. *(P.S.)*

### 7.2 Références pour information

[DES] National Institute of Standards and Technology. FIPS Pub 46 : "Data Encryption Standard". 15 janvier 1977.

[CamelliaTech] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., and Tokita, T., "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis -", dans "Selected Areas in Cryptography", 7th Annual International Workshop, SAC 2000, août 2000, Proceedings, notes de lecture dans Computer Science 2012, pp.39-56, Springer-Verlag, 2001.

[CRYPTREC] Information-technology Promotion Agency (IPA), Japan, CRYPTREC. http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html

[NESSIE] New European Schemes for Signatures, Integrity and Encryption (NESSIE) project. http://www.cryptonessie.org

## Appendice A Module ASN.1

CamelliaEncryptionAlgorithmInCMS
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) modules(0) id-mod-cms-camellia(23) }

ÉTIQUETTES IMPLICITES DE DÉFINITIONS ::=
DÉBUT

-- Camellia utilisant le mode de chaînage CBC pour les tailles de clé de 128, 192, 256 bits

IDENTIFIANT D'OBJET id-camellia128-cbc ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) symmetric-encryption-algorithm(1) camellia128-cbc(2) }

IDENTIFIANT D'OBJET id-camellia192-cbc ::={ iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) symmetric-encryption-algorithm(1) camellia192-cbc(3) }

IDENTIFIANT D'OBJET id-camellia256-cbc ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) symmetric-encryption-algorithm(1) camellia256-cbc(4) }

-- Camellia-IV est le paramètre pour tous les identifiants d'objet ci-dessus.

Camellia-IV ::= CHAÎNE D'OCTETS (TAILLE(16))

-- Paramètre de capacité Camellia S/MIME pour tous les identitifants d'objet ci-dessus.

CamelliaSMimeCapability ::= NULL

-- Identifiants d'algorithme d'enveloppement de clé Camellia - Paramètre est absent.

IDENTIFIANT D'OBJET id-camellia128-wrap ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) key-wrap-algorithm(3) camellia128-wrap(2) }

IDENTIFIANT D'OBJET id-camellia192-wrap ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) key-wrap-algorithm(3) camellia192-wrap(3) }

IDENTIFIANT D'OBJET id-camellia256-wrap ::= { iso(1) member-body(2) 392 200011 61 security(1) algorithm(1) key-wrap-algorithm(3) camellia256-wrap(4) }

FIN

## Adresse des auteurs

Shiho Moriai
Sony Computer Entertainment Inc.
téléphone : +81-3-6438-7523
Fax : +81-3-6438-8629
mél : camellia@isl.ntt.co.jp (équipe Camellia)
shiho@rd.scei.sony.co.jp (Shiho Moriai)

Akihiro Kato
NTT Software Corporation
téléphone : +81-45-212-7934
Fax : +81-45-212-9800
mél : akato@po.ntts.co.jp

## Déclaration de droits de reproduction



**Remerciement**

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