Groupe de travail Réseau
Request for Comments: 5510
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

J. Lacan, ISAE/LAAS-CNRS
V. Roca, INRIA
J. Peltotalo, Tampere University of Technology
S. Peltotalo, Tampere University of Technology
avril 2009

Schémas Reed-Solomon de correction d'erreur directe (FEC)

Résumé
Le présent document décrit un schéma pleinement spécifié de correction d'erreur directe (FEC, *Forward Error Correction*) pour les codes de FEC Reed-Solomon sur GF(2^^m) où m est dans {2..16}, et son application à la livraison fiable des objets de données sur le canal d'écrasement de paquet (c'est-à-dire, un chemin de communication où les paquets sont soit reçus sans aucune corruption, soit éliminés durant la transmission). Le présent document décrit aussi un schéma de FEC pleinement spécifié pour le cas particulier des codes Reed-Solomon sur GF(2^^8) quand il n'y a pas de groupe de symbole de codage. Finalement, dans le contexte du schéma de FEC systéméatique de petit bloc sous spécifié (identifiant de codage de FEC 129) le présent document alloue un identifiant d'instance de FEC au cas particulier des codes Reed-Solomon sur GF(2^^8).

Les codes Reed-Solomon appartiennent à la classe des codes de distance séparable maximum (MDS, *Maximum Distance Separable*) c'est-à-dire, ils permettent à un receveur de récupérer les k symboles de source à partir de tout ensemble de k symboles reçus. Les schémas décrits ici sont compatibles avec la mise en œuvre de Luigi Rizzo.

Table des matières

1. Introduction

L'utilisation des codes de correction d'erreur directe (FEC, *Forward Error Correction*) est une solution classique pour améliorer la fiabilité des transmissions de diffusion et de diffusion groupée. La [RFC5052] décrit un cadre général pour utiliser la FEC dans les protocoles de livraison de contenu (CDP, *Content Delivery Protocol*). Le document d'accompagnement [RFC3453] décrit des applications de codes de FEC pour la livraison de contenu.

Des schémas récents de FEC comme les [RFC5053] et [RFC5170] ont proposé des codes de suppression fondés sur des graphes/matrices épars. Ces codes sont efficaces en termes de traitement mais ne sont pas optimaux en termes de capacités de correction quand on traite de "petits" objets.

Les schémas de FEC décrits dans le présent document appartiennent à la classe des codes de distance maximum séparable qui sont optimaux en termes de capacité de correction de suppression. En d'autres termes, ils permettent à un receveur de récupérer les k symboles de source provenant de tout ensemble de exactement k symboles de codage. Ce sont aussi des codes systématiques, ce qui signifie que les k symboles de source font partie des symboles de codage. Même si la complexité de codage/décodage est supérieure à celle des [RFC5053] ou [RFC5170], cette famille de codes est très utile.

De nombreuses applications qui traitent de transmission de contenu ou de mémorisation de contenu s'appuient déjà sur les codes Reed-Solomon fondés sur la paquet. En particulier, nombre d'entre eux utilisent le codec Reed-Solomon [RS-codec] de Luigi Rizzo [Rizzo97]. Le but du présent document est de spécifier une mise en œuvre des codes Reed-Solomon compatible avec ce codec.

Le présent document :
- o introduit le schéma de FEC pleinement spécifié avec l'identifiant de codage de FEC de 2, qui spécifie l'utilisation de codes Reed-Solomon sur GF(2^^m) où m est dans l'intervalle {2..16},
- o le schéma de FEC pleinement spécifié avec l'identifiant de codage de FEC de 5, qui se concentre sur le cas particulier de codes Reed-Solomon sur GF(2^^8) et aucun groupe de symbole de codage (c'est-à-dire, exactement un symbole par paquet), et
- o dans le contexte du schéma de FEC de petit bloc systématique sous spécifié (identifiant de codage de FEC de 129) [RFC5445], alloue l'identifiant d'instance de FEC de 0 au cas particulier des codes Reed-Solomon sur GF(2^^8) et aucun groupe de symbole de codage.

Pour une définition des termes de schéma de FEC pleinement spécifié et sous spécifié, voir la [RFC5052], Section 4.

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],.

3. Définitions, notations et abréviations

3.1 Définitions

Le présent document utilise les mêmes termes et définitions que spécifiés dans la [RFC5052]. De plus, il utilise les définitions suivantes :

Symbole de source : unité de données utilisée durant le processus de codage.

Symbole de codage : unité de données générée par le processus de codage.

Symbole de réparation : symbole de codage qui n'est pas un symbole de source.

Taux de code : ratio k/n, c'est-à-dire, le ratio entre le nombre de symboles de source et le nombre de symboles de codage. Par définition, le taux de code est tel que : 0 < taux de code ≤ 1. Un taux de code proche de 1 indique qu'un petit nombre de symboles de réparation ont été produits durant le processus de codage.

Code systématique : code de FEC dans lequel les symboles de source font partie des symboles de codage.

Bloc de source : bloc de k symboles de source qui sont considérés ensemble pour le codage.

Groupe de symboles de codage : groupe de symboles de codage qui sont envoyés ensemble dans le même paquet, et dont les relations au bloc de source peuvent être déduites d'un seul identifiant de symbole de codage.

Paquet de source : paquet de données qui contient seulement des symboles de source.

Paquet de réparation : paquet de données qui contient seulement des symboles de réparation.

Canal d'écrasement de paquets : chemin de communication où les paquets sont soit éliminés (par exemple, par un routeur encombré, ou parce que le nombre d'erreurs de transmission excède les capacités de correction des codes de couche physique) soit reçus. Quand un paquet est reçu, il est supposé que ce paquet n'est pas corrompu.

3.2 Notations

Le présent document utilise les notations suivantes :

L : longueur du transfert d'objet en octets.
k : nombre de symboles de source dans un bloc de source.
n_r : nombre de symboles de réparation générés pour un bloc de source.
n : longueur du bloc de codage, c'est-à-dire, nombre de symboles de codage générés pour un bloc de source. Donc : n = k + n_r.
max_n : nombre maximum de symboles de codage générés pour tout bloc de source.
B : longueur maximum de bloc de source en symboles, c'est-à-dire, nombre maximum de symboles de source par bloc de source.
N : nombre de blocs de source en lequel l'objet devra être partagé.
E : longueur de symbole de codage en octets.
S : taille de symbole en unités d'éléments de m bits. Quand m = 8, alors S et E sont égaux.
m : longueur des éléments dans le champ fini, en bits. Dans ce document, m appartient à {2..16}.
q : nombre d'éléments dans le champ fini. On a : q = 2^^m dans cette spécification.
G : nombre de symboles de codage par groupe, c'est-à-dire, le nombre de symboles envoyés dans le même paquet.
GM : matrice de générateur d'un code Reed-Solomon.
CR : "taux de code", c'est-à-dire, le ratio k/n.
a^^b : a à la puissance b.
a^^-1 : inverse de a.
I_k : matrice d'identité k*k.

3.3 Abréviations

Le présent document utilise les abréviations suivantes :

ESI *(Encoding Symbol ID)* : identifiant de symbole de codage.
OTI FEC *(FEC Object Transmission Information)* : informations de transmission d'objet de FEC.
RS : Reed-Solomon.
MDS *(Maximum Distance Separable)* : code de distance maximum séparable.
GF(q) : champ fini (aussi appelé champ de Galois) avec q éléments. On suppose que q = 2^^m dans ce document.

4. Formats et codes avec l'identifiant 2 de codage de FEC

Cette section introduit les formats et codes associés au schéma pleinement spécifié de FEC avec identifiant de codage de FEC de 2, qui specifie l'utilisation des codes Reed-Solomon sur GF(2^^m).

4.1 Identifiant de charge utile de FEC

L'identifiant de charge utile de FEC est composé du nombre de blocs de source et de l'identifiant de symbole de codage. Les longueurs de ces deux champs dépendent du paramètre m (qui est transmis dans les OTI de FEC) comme suit :

- Le nombre de bloc de source (champ de 32 m bits) identifie de quel bloc de source de l'objet sont générés le ou les symboles de codage dans la charge utile. Il y a un maximum de 2^^(32-m) blocs par objet.

- L'identifiant de symbole de codage (champ de m bits) identifie quels symboles de codage spécifiques générés à partir des blocs de source sont portés dans la charge utile du paquet. Il y a un maximum de 2^^m symboles de codage par bloc. Les k premières valeurs (0 à k - 1) identifient les symboles de source, les n-k valeurs restantes identifient les symboles de réparation.

Il DOIT y avoir exactement un identifiant de charge utile de FEC par paquet de source ou de réparation. Dans le cas d'un groupe de symboles de codage, quand plusieurs symboles de codage sont envoyés dans le même paquet, l'identifiifant de charge utile de FEC se réfère au premier symbole du paquet. Les autres symboles peuvent être déduits de l'ESI du premier symbole en incrémentant les ESI à la suite.

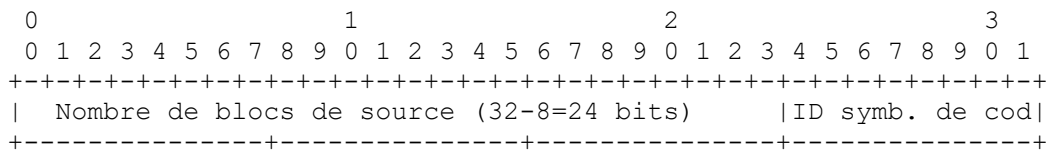


Figure 1 : Format de codage d'identifiant de charge utile de FEC pour m = 8 (défaut)

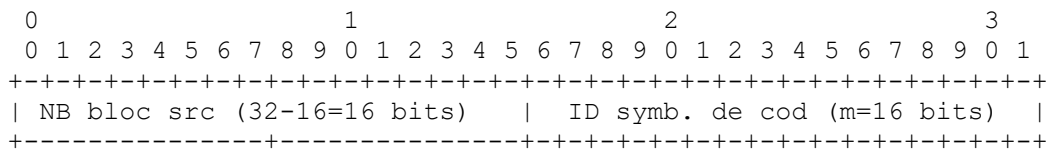


Figure 2 : Format de codage d'identifiant de charge utile de FEC pour m = 16

Les formats de l'identifiant de charge utile de FEC pour m = 8 et m = 16 sont illustrés dans les Figures 1 et 2, respectivement.

4.2 Informations de transmission d'objet de FEC

4.2.1 Éléments obligatoires

- Identifiant de codage de FEC : le schéma pleinement spécifié de FEC décrit dans cette section utilise l'identifiant de codage de FEC de 2.

4.2.2 Éléments communs

Les éléments suivants DOIVENT être définis avec le présent schéma de FEC.

- Longueur de transfert (L) : entier non négatif indiquant la longueur de l'objet en octets. Il y a des restrictions sur la longueur de transfert maximum qui peut être prise en charge :

  longueur_max_transfert = 2^^(32-m) * B * E

  Par exemple, pour m = 8, pour B = 2^^8 - 1 (parce que le codec opère sur un champ fini avec 2^^8 éléments) et si E = 1024 octets, alors la longueur maximum de transfert est approximativement égale à 2^^42 octets (c'est-à-dire, 4 teraoctets). De même, pour m = 16, pour B = 2^^16 - 1, et si E = 1024 octets, alors la longueur maximum de transfert est aussi approximativement égale à 2^^42 octets. Pour de plus grands objets, un autre schéma de FEC, avec un plus grand champ Nombre de blocs de source dans l'identifiant de charge utile de FEC, pourrait être défini. Une autre solution consiste à fragmenter les grands objets en plus petits objets, chacun se conformant aux limites ci-dessus.

- Longueur de symbole de codage (E) : entier non négatif indiquant la longueur de chaque symbole de codage en octets.

- Longueur maximum de bloc de source (B) : entier non négatif indiquant le nombre maximum de symboles de source dans un bloc de source.

- Nombre maximum de symboles de codage (max_n) : entier non négatif indiquant le nombre maximum de symboles de codage généré pour tout bloc de source.

La Section 6 explique comment déduire les valeurs de chacun de ces éléments.

4.2.3 Éléments spécifiques du schéma

L'élément suivant DOIT être défini avec le présent schéma de FEC. Il contient deux éléments d'information distincts :

G : entier non négatif indiquant le nombre de symboles de codage par groupe utilisé pour l'objet. La valeur par défaut est 1, ce qui signifie que chaque paquet contient exactement un symbole. Quand aucun paramètre G n'est communiqué au décodeur, alors ce dernier DOIT supposer que G = 1.

m : le paramètre m est la longueur des éléments du champ fini, en bits. Il caractérise aussi le nombre d'éléments dans le champ fini : q = 2^^m éléments. La valeur par défaut est m = 8. Quand aucun paramètre de taille de champ fini n'est communiqué au décodeur, ce dernier DOIT alors supposer que m = 8.

4.2.4 Format de codage

Ce paragraphe montre les deux formats de codage possibles des OTI de FEC ci-dessus. Le présent document ne spécifie pas quand un format de codage ou l'autre devrait être utilisé.

4.2.4.1 Utilisation du format général EXT_FTI

Le format binaire d'OTI de FEC est le suivant, quand le mécanisme EXT_FTI est utilisé (par exemple, dans les protocoles ALC [RFC5775] ou NORM [RFC5740]).

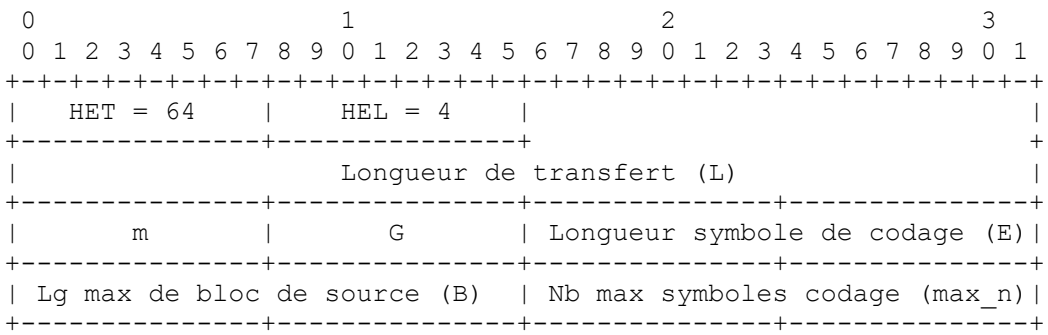


Figure 3 : Format d'en-tête EXT_FTI

4.2.4.2 Utilisation de l'instance FDT (spécifique de FLUTE)

Quand il est désiré que les OTI de FECsoient portées dans l'instance de tableau de livraison de fichier (FDT, *File Delivery Table*) d'une session FLUTE [RFC6726], les attributs XML suivants doivent être décrits pour l'objet associé :
- o FEC-OTI-FEC-Encoding-ID *(identifiant de codage de FEC d'informations de transmission d'objet de FEC)*
- o FEC-OTI-Transfer-Length (L)
- o FEC-OTI-Encoding-Symbol-Length (E)
- o FEC-OTI-Maximum-Source-Block-Length (B)
- o FEC-OTI-Max-Number-of-Encoding-Symbols (max_n)
- o FEC-OTI-Scheme-Specific-Info *(informations spécifiques du schéma d'OTI de FEC)*

Les informations spécifiques du schéma d'OTI de FEC contiennent la chaîne résultant du codage en Base64 (au sens du schéma XML xs:base64Binary) de la valeur suivante :

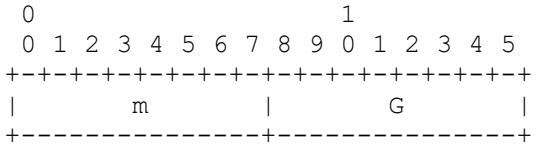


Figure 4 : Informations spécifiques du schéma d'OTI de FEC à inclure dans l'instance de FDT

Quand aucun paramètre m n'est à porter dans l'OTI de FEC, le champ m est réglé à 0 (qui n'est pas une valeur de germe valide). Autrement, le champ m contient une valeur valide comme expliqué au paragraphe 4.2.3. De même, quand aucun paramètre G n'est à porter dans les OTI de FEC, le champ G est réglé à 0 (qui n'est pas une valeur de germe valide). Autrement, le champ G contient une valeur valide comme expliqué au paragraphe 4.2.3. Quand ni m ni G ne sont à porter dans les OTI de FEC, alors l'envoyeur omet simplement l'attribut FEC-OTI-Scheme-Specific-Info.

Durant le codage Base64, les 2 octets des informations spécifiques de schéma d'OTI de FEC sont transformés en une chaîne de 4 caractères imprimables (dans l'alphabet de 64 caractères) qui est ajoutée à l'attribut FEC-OTI-Scheme-Specific-Info.

5. Formats et codes avec l'identifiant de codage de FEC 5

Cette section introduit les formats et codes associés au schéma pleinement spécifié de FEC avec identifiant de codage de FEC de 5, qui se concentre sur le cas particulier des codes Reed-Solomon sur GF(2^^8) et aucun groupe de symboles de codage.

5.1 Identifiant de charge utile de FEC

L'identifiant de charge utile de FEC est composé du nombre de blocs de source et de l'identifiant de symbole de codage :
- o Le nombre de blocs de source (champ de 24 bits) identifie de quel bloc de source de l'objet est généré le symbole de codage dans la charge utile. Il y a un maximum de 2^^24 blocs par objet.
- o L' identifiant de symbole de codage (champ de 8 bits) identifie quel symbole de codage spécifique généré à partir du bloc de source est porté dans la charge utile du paquet. Il y a un maximum de 2^^8 symboles de codage par bloc. Les k premières valeurs (0 à k - 1) identifient les symboles de source; les n-k valeurs restantes identifient les symboles de réparation.

Il DOIT y avoir exactement un identifiant de charge utile de FEC par paquet de source ou de réparation. Cet identifiant de charge utile de FEC se réfère à un et seulement un symbole du paquet.

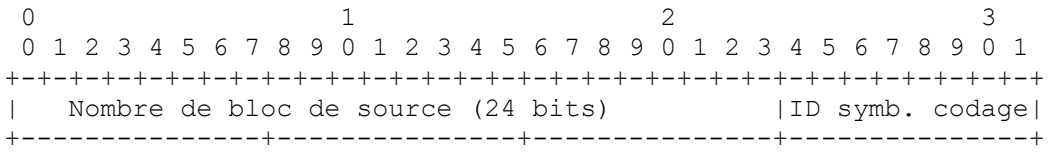


Figure 5 : Format de codage d'identifiant de charge utile de FEC avec identifiant de codage de FEC 5

5.2 Informations de transmission d'objet de FEC

5.2.1 Éléments obligatoires

o Identifiant de codage de FEC : le schéma pleinement spécifié de FEC décrit dans ce paragraphe utilise l'identifiant de codage de FEC de 5.

5.2.2 Éléments communs

Les éléments communs sont les mêmes que ceux spécifiés au paragraphe 4.2.2 quand m = 8 et G = 1.

5.2.3 Éléments spécifiques du schéma

Aucun élément spécifique du schéma n'est défini par ce schéma de FEC.

5.2.4 Format de codage

Ce paragraphe montre les deux formats de codage possibles des OTI de FEC ci-dessus. Le présent document ne spécifie pas quand un format de codage ou l'autre devrait être utilisé.

5.2.4.1 Utilisation du format général EXT_FTI

Le format binaire d'OTI de FEC est le suivant, quand le mécanisme EXT_FTI est utilisé (par exemple, dans les protocoles ALC [RFC5775] ou NORM [RFC5740]).

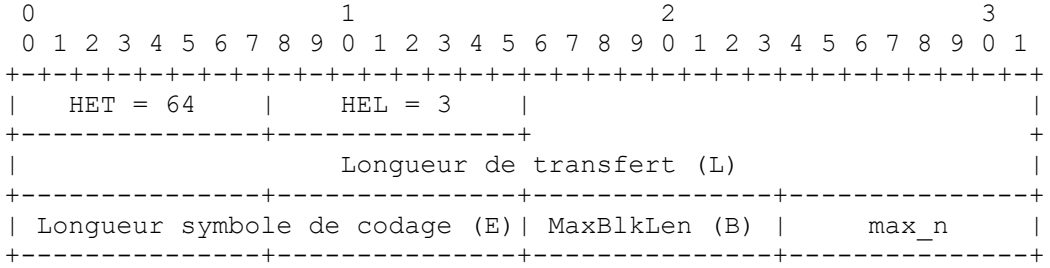


Figure 6 : Format d'en-tête EXT_FTI avec identifiant de codage de FEC 5

5.2.4.2 Utilisation de l'instance FDT (spécifique de FLUTE)

Quand il est désiré que les OTI de FEC soient portées dans l'instance de FDT d'une session FLUTE [RFC6726], les attributs XML suivants doivent être décrits pour l'objet associé :

o FEC-OTI-FEC-Encoding-ID
o FEC-OTI-Transfer-Length (L)
o FEC-OTI-Encoding-Symbol-Length (E)
o FEC-OTI-Maximum-Source-Block-Length (B)
o FEC-OTI-Max-Number-of-Encoding-Symbols (max_n)

6. Procédures avec les identifiants de codage de FEC 2 et 5

Cette Section définit les procédures qui sont communes aux identifiants de codage de FEC 2 et 5. Dans le cas d'un identifiant de codage de FEC de 5, m = 8 et G = 1. L'algorithme de partition de bloc défini au paragraphe 9.1 de la [RFC5052] DOIT être utilisé avec les identifiants de codage de FEC 2 et 5.

6.1 Détermination de la longueur maximum de bloc de source (B)

Le paramètre de taille de champ fini, m, définit le nombre d'éléments non zéro dans ce champ, qui est égal à : q - 1 = 2^^m - 1. Noter que q - 1 est aussi le nombre théorique maximum de symboles de codage qui peuvent être produits pour un bloc de source. Par exemple, quand m = 8 (défaut) il y a un maximum de 2^^8 - 1 = 255 symboles de codage.

Étant donné le taux de code cible de FEC (par exemple, fourni par l'utilisateur quand il débute une application d'envoi FLUTE) l'envoyeur calcule :

max1_B = plancher((2^^m - 1) * CR)

Cette valeur de max1_B laisse assez de place pour que l'envoyeur produise le nombre désiré de symboles de parité.

De plus, un codec PEUT imposer d'autres limitations à la taille maximum de bloc. Il n'est cependant pas attendu que de telles limites existent quand on utilise la valeur par défaut de m = 8. Cette décision DOIT être précisée au moment de la mise en œuvre, quand le cas d'utilisation cible est connu. Il en résulte une limitation de max2_B.

Alors, B est donné par : B = min(max1_B, max2_B)

Noter que ce calcul n'est exigé qu'au codeur, car le paramètre B est communiqué par le décodeur dans les OTI de FEC.

6.2 Détermination du nombre de symboles de codage d'un bloc

L'algorithme suivant, aussi appelé "n-algorithm", explique comment déterminer le nombre maximum de symboles de codage générés pour tout bloc de source (max_n) et le nombre de symboles de codage pour un bloc donné (n) comme une fonction du taux de code cible.

Chez un envoyeur :

Entrée :
B : longueur maximum de bloc de source, pour tout bloc de source. Le paragraphe 6.1 explique comment déterminer sa valeur.
k : longueur actuelle de bloc de source. Ce paramètre est donnés par l'algorithme de partition de bloc.
CR : taux de code de FEC, qui est donné par l'utilisateur (par exemple, quand il débute une application d'envoi FLUTE). Il est exprimé comme une valeur en virgule flottante.

Sortie :
max_n : nombre maximum de symboles de codage générés pour tout bloc de source.
n : nombre de symboles de codage générés pour ce bloc de source.

Algorithme :
max_n = plafond(B / CR) ;
si (max_n > 2^^m - 1), alors retourner une erreur ("taux de code invalide") ;
n = plancher(k * max_n / B);

Chez le receveur :

Entrée :
B : Extrait des OTI de FEC reçues.
max_n : Extrait des OTI de FEC reçues.
k : Donné par l'algorithme de partition de bloc.

Sortie : n

Algorithme : n = plancher(k * max_n / B);

Il est RECOMMANDÉ que le "n-algorithm" soit utilisé par un envoyeur, mais d'autres algorithmes restent possibles pour déterminer max_n et/ou n.

Chez un receveur, la valeur max_n est extraite des OTI de FEC reçues. Comme le décodeur Reed-Solomon n'a pas besoin de connaître la valeur réelle de n, utiliser la partie receveur du "n-algorithm" n'est pas nécessaire du point de vue du décodage.

Cependant, un receveur peut vouloir avoir une estimation de n pour d'autres raisons (par exemple, pour des besoins de gestion de mémoire). Dans ce cas, un receveur sait que le nombre de symboles de codage d'un bloc ne peut pas excéder max_n. De plus, si un receveur croit qu'un envoyeur utilise le "n-algorithm", ce receveur PEUT utiliser la partie receveur du "n-algorithm" pour obtenir une meilleure estimation de n. Quand c'est le cas, un receveur DOIT être prêt à traiter les symboles avec un identifiant de symbole de codage supérieur ou égal à la valeur n calculée (par exemple, il peut choisir de simplement les éliminer).

7. Schéma de FEC systématique de petit bloc (ID 129) et codes Reed-Solomon sur GF(2^^8)

Dans le contexte du schéma sous spécifié de petit bloc systématique de FEC (identifiant de codage de FEC 129) [RFC5445], le présent document alloue l'identifiant d'instance de FEC de 0 au cas particulier des codes Reed-Solomon sur GF(2^^8) et aucun groupe de symboles de codage.

L'identifiant d'instance de 0 utilise les formats et codes spécifiés dans la [RFC5445].

Le schéma de FEC avec l'identifiant d'instance de 0 PEUT utiliser l'algorithme de partition de bloc défini au paragraphe 9.1 de la [RFC5052] pour partager l'objet en blocs de source. Ce schéma de FEC PEUT aussi utiliser un autre algorithme. Par exemple, l'envoyeur CDP peut changer la longueur de chaque bloc de source de façon dynamique, selon des critères externes (par exemple, pour ajuster le taux de codage de FEC au taux de pertes actuel rencontré par les receveurs NORM) et informer les receveurs CDP de la longueur actuelle de bloc au moyen du mécanisme EXT_FTI. Ce choix sort du domaine d'application de ce document.

8. Spécification des codes Reed-Solomon pour le canal d'écrasement

Les codes Reed-Solomon (RS) sont des codes de bloc linéaires. Ils appartiennent aussi à la classe des codes MDS. Un code RS [n,k] code une séquence de k éléments de source définis sur un champ fini GF(q) dans une séquence de n éléments de codage, où n est a une limite supérieure de q - 1. La mise en œuvre décrite dans ce document se fonde sur une matrice génératrice construite à partir d'une matrice de Vandermonde mise en forme systématique.

Les paragraphes 8.1 à 8.3 spécifient les codes RS [n,k] quand ils sont appliqués aux éléments de m bits, et le paragraphe 8.4 spécifie l'utilisation des codes RS [n,k] quand ils sont appliqués aux symboles composés de plusieurs éléments de m bits. L'utilisation décrite au paragraphe 8.4 est le cœur de la présente spécification.

Le lecteur qui veut comprendre la théorie sous-jacente est invité à se reporter aux références [Rizzo97] et [MWS77].

8.1 Champ fini

Un champ fini GF(q) est défini comme un ensemble fini de q éléments qui a une structure de champ. Il contient nécessairement q = p^^m éléments, où p est un nombre premier. Avec les canaux d'écrasement de paquet, p est toujours réglé à 2. Les éléments du champ GF(2^^m) peuvent être représentés par des polynômes avec des coefficients binaires (c'est-à-dire, sur GF(2)) de degré inférieur ou égal à m-1. Les polynômes peuvent être associés à des vecteurs binaires de longueur m. Par exemple, le vecteur (11001) représente le polynôme 1 + x + x^^4. Cette représentation est souvent appelée une représentation polynomiale. L'addition de deux éléments est définie comme l'addition de polynômes binaires dans GF(2) et la multiplication est la multiplication modulo un polynôme irréductible donné sur GF(2) de degré m. Noter que toutes les racines de ce polynôme sont dans GF(2^^m) mais pas dans GF(2).

La représentation polynomiale choisie du champ fini GF(2^^m) est complètement caractérisée par le polynôme irréductible. Les polynômes suivants sont choisis pour représenter le champ GF(2^^m) pour m variant de 2 à 16 :

m = 2, "111" (1+x+x^^2)
m = 3, "1101", (1+x+x^^3)
m = 4, "11001", (1+x+x^^4)
m = 5, "101001", (1+x^^2+x^^5)
m = 6, "1100001", (1+x+x^^6)
m = 7, "10010001", (1+x^^3+x^^7)
m = 8, "101110001", (1+x^^2+x^^3+x^^4+x^^8)
m = 9, "1000100001", (1+x^^4+x^^9)
m = 10, "10010000001", (1+x^^3+x^^10)
m = 11, "101000000001", (1+x^^2+x^^11)
m = 12, "1100101000001", (1+x+x^^4+x^^6+x^^12)
m = 13, "11011000000001", (1+x+x^^3+x^^4+x^^13)
m = 14, "110000100010001", (1+x+x^^6+x^^10+x^^14)
m = 15, "1100000000000001", (1+x+x^^15)
m = 16, "11010000000010001", (1+x+x^^3+x^^12+x^^16)

Afin de faciliter la mise en œuvre, ces polynômes sont aussi primitifs. Cela signifie que tout élément de GF(2^^m) peut être exprimé comme une puissance d'une racine donnée de ce polynôme. Ces polynômes sont aussi choisis de telle façon qu'ils contiennent le nombre minimum de monômes.

8.2 Algorithme de codage Reed-Solomon

8.2.1 Principes de codage

Soit s = (s_0, ..., s_{k-1}) le vecteur source de k éléments sur GF(2^^m). Soit e = (e_0, ..., e_{n-1}) le vecteur de codage correspondant de n éléments sur GF(2^^m). Comme c'est un code linéaire, le codage est effectué en multipliant le vecteur de source par une matrice génératrice, GM, de k rangées et n colonnes sur GF(2^^m). Donc : e = s * GM.

La définition de la matrice génératrice caractérise complètement le code RS.

Considérons que n = 2^^m - 1 et que 0 < k ≤ n. On note par alpha la racine du polynôme primitif de degré m choisi dans la liste du paragraphe 8.1 pour la valeur correspondante de m. Considérons une matrice de Vandermonde de k rangées et n colonnes, notée par V_{k,n}, et construite comme suit : l'entrée {i, j} de V_{k,n} est v_{i,j} = alpha^^(i*j), où 0 ≤ i ≤ k - 1 et 0 ≤ j ≤ n - 1. Cette matrice génère un code MDS. Cependant, ce code MDS n'est pas systématique, ce qui pose un problème pour de nombreuses applications de réseautage. Pour obtenir une matrice (et code) systématique, la solution la plus simple consiste à considérer la matrice V_{k,k} formée par les k premières colonnes de V_{k,n}, puis de l'inverser et de multiplier cet inverse par V_{k,n}. En clair, le produit V_{k,k}^^-1 * V_{k,n} contient la matrice d'identité I_k sur ses k premières colonnes, ce qui signifie que les k premiers éléments de codage sont égaux aux éléments de source. Par ailleurs, le code associé conserve la propriété MDS.

Donc, la matrice génératrice du code considérée dans ce document est : GM = (V_{k,k}^^-1) * V_{k,n}

Noter que, en pratique, le code RS [n,k] peut être abrégé en un code RS [n',k] où k ≤ n' < n, en considérant la sous matrice formée par les n' premières colonnes de GM.

8.2.2 Complexité de codage

Le codage peut être effectué en pré calculant d'abord GM et en multipliant le vecteur de source (k éléments) par GM (k rangées et n colonnes). La complexité du pré calcul de la matrice génératrice peut être estimée comme la complexité de la multiplication de l'inverse d'une matrice de Vandermonde par n-k vecteurs (c'est-à-dire, les n-k dernières colonnes de V_{k,n}). Comme la complexité de l'inverse d'une matrice k*k de Vandermonde par un vecteur est O(k * (log(k))^^2), la matrice génératrice peut être calculée en 0((n-k)* k * (log(k))^^2)) opérations. Quand la matrice génératrice est pré calculée, le codage a besoin de k opérations par élément de réparation (multiplication de matrice de vecteurs).

Le codage peut aussi être effectué en calculant d'abord le produit s * V_{k,k}^^-1 et ensuite en multipliant le résultar par V_{k,n}. La multiplication par l'inverse d'une matrice de Vandermonde carrée est connu comme le problème de l'interpolation et sa complexité est O(k * (log(k))^^2). La multiplication par une matrice de Vandermonde, appelée le problème de l'évaluation multi points, exige O((n-k) * log(k)) en utilisant la transformation de Fourier rapide, comme expliqué dans [GO94]. La complexité totale de cet algorithme de codage est alors O((k/(n-k)) * (log(k))^^2 + log(k)) opérations par élément de réparation.

8.3 Algorithme de décodage Reed-Solomon

8.3.1 Principes de décodage

L'algorithme de décodage Reed-Solomon pour le canal d'écrasement permet la récupération de k éléments de source provenant de tout ensemble de k éléments reçus. Il se fonde sur la propriété fondamentale de la matrice génératrice, qui est telle que toute sous matrice k*k est inversable (voir [MWS77]). La première étape du décodage consiste en l'extraction de la sous matrice k*k de la matrice génératrice obtenue en considérant les colonnes correspondant aux éléments reçus. Bien sûr, comme tout élément codant est obtenu en multipliant le vecteur de source par une colonne de la matrice génératrice, le vecteur reçu de k éléments de codage peut être considéré comme le résultat de la multiplication du vecteur de source par une sous matrice k*k de la matrice génératrice. Comme cette sous matrice est inversable, la seconde étape de l'algorithme est d'inverser cette matrice et de multiplier le vecteur reçu par la matrice obtenue pour récupérer le vecteur de source.

8.3.2 Complexité de décodage

L'algorithme de décodage décrit précédemment inclut la multiplication de l'inversion de matrice et de la matrice de vecteur. Avec l'algorithme classique de Gauss-Jordan, l'inversion de matrice exige O(k^^3) opérations et la multiplication de la matrice de vecteur est effectuée en O(k^^2) opérations.

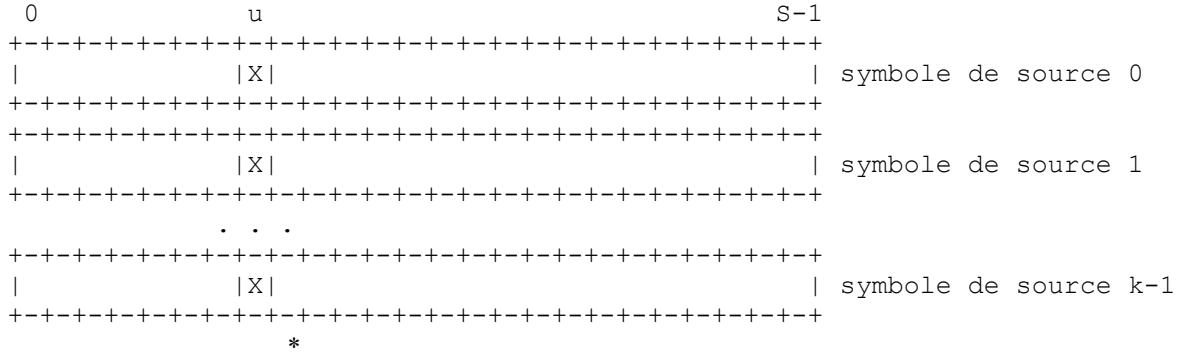
Cette complexité peut être améliorée en considérant que la sous matrice reçue de GM est le produit de l'inverse d'une matrice de Vandermonde (V_(k,k)^^-1) et d'une autre matrice de Vandermonde (notée V', qui est une sous matrice de V_(k,n)). Le décodage peut être fait en multipliant le vecteur reçu par V'^^-1 (le problème de l'interpolation avec la complexité O(k * (log(k))^^2)) puis par V_{k,k} (évaluation multipoints avec la complexité O(k * log(k))). La complexité de décodage globale est alors de O((log(k))^^2) opérations par élément de source.

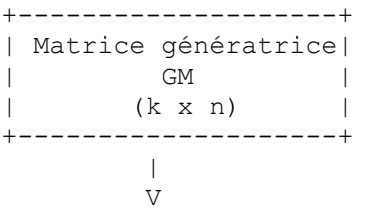
8.4 Mise en œuvre pour le canal d'écrasement de paquet

Dans un canal d'écrasement de paquets, chaque paquet (incluant son ou ses symboles, car les paquets contiennent G ≥ 1 symboles) est soit correctement reçu, soit écrasé. La localisation des symboles écrasés dans la séquence de symboles DOIT être connue. La spécification qui suit décrit l'utilisation des codes Reed-Solomon pour générer des symboles redondants à partir de k symboles de source et pour récupérer les symboles de source de tout ensemble de k symboles reçus.
Les k symboles de source d'un bloc de source sont supposés être composés de S éléments de m bits. Chaque élément de m bits correspond à un élément du champ fini GF(2^^m) à travers la représentation polynomiale (paragraphe 8.1). Si certains des symboles de source contiennent moins de S éléments, ils DOIVENT être virtuellement bourrés avec des éléments zéro (ce peut être le cas pour le dernier symbole du derniet bloc de l'objet). Cependant, ce bourrage n'a pas besoin d'être réellement envoyé avec les données aux receveurs.

Le processus de codage produit n symboles de codage de taille S d'éléments de m bits, dont k sont des symboles de source (c'est un code systématique) et n-k sont des symboles de réparation (Figure 7). Les éléments de m bits des symboles de réparation sont calculés en utilisant les éléments de m bits correspondants de l'ensemble de symboles de source. Un vecteur de source logique u-ième, composé des u-ièmes éléments provenant de l'ensemble de symboles de source, est utilisé pour calculer un u-ième vecteur de codage. Ce u-ième vecteur de codage fournit alors les u-ièmes éléments pour l'ensemble de symboles de codage calculés pour le bloc. Comme code systématique, les k premiers symboles de codage sont les mêmes que les k symboles de source, et les n-k derniers symboles de réparation sont le résultat du codage Reed-Solomon.

Entrée : k symboles de source





Résultat : n symboles de codage (source + réparation)

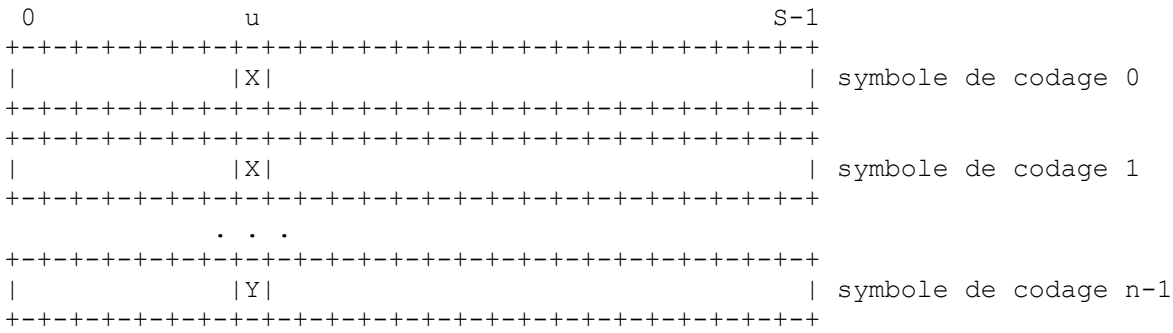


Figure 7 : Schéma de codage de paquet

Un acquit de ce schéma est que la perte de certains symboles de codage produit le même shéma d'écrasement pour chacun des S vecteurs de codage. Il s'ensuit que l'inversion de matrice doit être faite seulement une fois et va être utilisée par tous les S vecteurs de codage. Pour un S grand, le coût de cette inversion de matrice devient négligeable en face des S multiplications de matrice de vecteur.

Un autre acquit est que les n-k symboles de réparation peuvent être produits à la demande. Par exemple, un envoyeur peut commencer par produire un nombre limité de symboles de réparation et plus tard, selon les écrasements observés sur le canal, décider de produire des symboles de réparation supplémentaires, jusqu'à la limite supérieure de n-k. Bien sûr, pour produire le symbole de réparation e_j, où $k \leq j < n$, il est suffisant de multiplier les S vecteurs de source par la colonne j de GM.

9. Considérations sur la sécurité

9.1 Position du problème

Un système de livraison de contenus est potentiellement sujet à de nombreuses attaques : certaines d'entre elles ciblent le réseau (par exemple, pour compromettre l'infrastructure d'acheminement, en compromettant le composant de contrôle d'encombrement) d'autres ciblent le protocole de livraison de contenu (CDP, *Content Delivery Protocol*) (par exemple, pour compromettre son comportement normal) et finalement certaines attaques ciblent le contenu lui-même. Comme ce document se concentre sur un bloc de construction de FEC indépendamment de tout CDP particulier (même si ALC et NORM sont deux candidats naturels) cette section discute seulement des menaces supplémentaires auxquelles un CDP arbitraire peut être exposé quand il utilise ce bloc de construction.

Plus spécifiquement, plusieurs sortes d'attaques existent :
- o celles qui sont destinées à donner accès à du contenu confidentiel (par exemple, en cas de contenu non gratuit) ;
- o celles qui essayent de corrompre l'objet transmis (par exemple, pour injecter du code malveillant dans un objet ou pour empêcher un receveur d'utiliser un objet) ;
- o et celles qui esssayent de compromettre le comportement du receveur (par exemple, en rendant le décodage d'un objet coûteux en calcul).

Ces attaques peuvent être lancées contre le flux de données lui-même (par exemple, en envoyant des symboles falsifiés) ou contre les paramètres de FEC qui sont envoyés dans la bande (par exemple, dans une instance EXT_FTI ou FDT) ou hors bande (par exemple, dans une description de session).

9.2 Attaques contre le flux de données

Tout d'abord, considérons les attaques contre le flux de données.

9.2.1 Accès aux objets confidentiels

Le contrôle de l'accès à l'objet transmis est normalement assuré au moyen du chiffrement. Ce chiffrement peut être fait sur l'objet entier (par exemple, par le fournisseur du contenu, avant le processus de codage de FEC) ou être fait paquet par paquet (par exemple, quand l'encapsulation de charge utile de sécurité (ESP, *Encapsulating Security Payload*) IPsec est utilisée [RFC4303]). Si le contrôle d'accès est un problème, il est RECOMMANDÉ qu'une de ces solutions soit utilisée. Même si on mentionne ici ces attaques, elles ne sont pas relatives à l'utilisation de la FEC ni facilitées par elle.

9.2.2 Corruption du contenu

La protection contre la corruption (par exemple, après l'envoi de paquets falsifiés) est réalisée au moyen d'un schéma de vérification de l'intégrité du contenu / authentification de l'envoyeur. Ce service peut être fourni au niveau de l'objet, mais dans le cas où un receveur n'a pas de moyen d'identifier quels symboles sont corrompus si l'objet est détecté comme corrompu. Ce service peut aussi être fourni au niveau du paquet. Dans ce cas, après avoir supprimé tous les paquets falsifiéss, l'objet peut parfois être récupéré. Plusieurs techniques peuvent fournir ce service d'authentification de source / intégrité du contenu:

- Au niveau de l'objet, l'objet PEUT être signé numériquement (avec la cryptographie à clé publique) par exemple en utilisant RSASSA-PKCS1-v1_5 [RFC3447]. Cette signature permet à un receveur de vérifier l'intégrité de l'objet, une fois que l'objet a été complètement décodé. Même si les signatures numériques sont coûteuses en calcul, ce calcul n'intervient qu'une seule fois par objet, ce qui est généralement acceptable.

- Au niveau du paquet, chaque paquet peut être signé numériquement. Une limitation majeure est la quantité élevée de frais généraux de calcul et de transmission que cette solution exige (sauf si la cryptographie par courbe elliptique (ECC, *Elliptic Curve Cryptography*) est utilisée). Pour éviter ce problème, la signature peut être étendue sur un ensemble de symboles (au lieu d'un seul) afin d'amortir le calcul de signature. Mais si un seul symbole manque, l'intégrité de tout l'ensemble ne peut pas être vérifiée.

- Au niveau du paquet, un schéma de code d'authentification de message (MAC, *Message Authentication Code*) [RFC2104] de groupe peut être utilisé ; par exemple, en utilisant HMAC-SHA-256 avec une clé secrète partagée par tous les membres du groupe (c'est-à-dire, les envoyeurs et les receveurs). Grâce à la clé secrète, cette technique crée un résumé sécurisé cryptographiquement d'un paquet, qui est envoyé avec le paquet. Le schéma de MAC de groupe ne crée pas de charge de traitement prohibitive ni de frais généraux de transmission, mais il a une limitation majeure : il fournit seulement un service d'authentification/intégrité de groupe car tous les membres du groupe partagent la même clé de groupe secrète, ce qui signifie que chaque membre peut envoyer un paquet falsifié. Il est donc restreint aux situations où les membres du groupe sont pleinement de confiance (ou en association avec une autre technique comme une pré vérification).

- Au niveau du paquet, TESLA [RFC4082] est une solution très attirante et efficace qui est robuste aux pertes, founit un vrai service d'authentification/intégrité, et ne crée aucune charge de traitement prohibitive ou frais généraux de transmission. Mais la vérification d'un paquet requiert un petit délai (une seconde ou plus) après sa réception.

Des techniques s'appuyant sur la cryptographie à clé publique (signatures numériques et TESLA durant le processus d'amorçage, quand elles sont utilisées) exigent que les clés publiques soient associées de façon sûre aux entités. Cela peut être réalisé par une infrastructure de clé publique (PKI, *Public Key Infrastructure*) ou par une toile de confiance PGP, ou en pré distribuant les clés publiques de chaque membre du groupe.

Les techniques qui s'appuient sur la cryptographie à clé symétrique (MAC de groupe) exigent qu'une clé secrète soit partagée par tous les membres du groupe. Cela peut être réalisé au moyen d'un protocole de gestion de clé de groupe, ou simplement en pré distribuant la clé secrète (mais cette solution manuelle a de nombreuses limitations).

Il appartient au développeur et au déployeur, qui connaissent les exigences et caractéristiques de sécurité de la zone d'application cible, de définir quelle solution est la plus appropriée. Néanmoins, dans le cas où la menace de corruption de l'objet est un problème, il est RECOMMANDÉ qu'au moins une de ces techniques soit utilisée.

9.3 Attaques contre les paramètres de FEC

Considérons maintenant les attaques contre les paramètres de FEC (ou OTI de FEC). Les OTI de FEC peuvent être

envoyées dans la bande (c'est-à-dire, dans une instance EXT_FTI ou FDT contenant les OTI de FEC pour l'objet) ou hors bande (par exemple, dans une description de session). Les attaques sur ces paramètres de FEC peuvent empêcher le décodage de l'objet associé : par exemple, modifier le paramètre B va conduire à un partage de bloc différent chez le receveur compromettant ainsi le décodage ; ou régler le paramètre m à 16 au lieu de 8 avec l'identifiant de codage de FEC de 2 va augmenter la charge de traitement tout en compromettant le décodage.

Il est donc RECOMMANDÉ que des mesures de sécurité soient prises pour garantir l'intégrité des OTI de FEC. À cette fin, les paquets portant les paramètres de FEC envoyés dans la bande dans une extension d'en-tête EXT_FTI DEVRAIENT être protégés par une des techniques par paquet décrites si-dessus : signature numérique, MAC de groupe, ou TESLA. Quand les OTI de FEC sont contenues dans une instance de FDT, cet objet d'instance de FDT DEVRAIT être protégée, par exemple, en la signant numériquement avec des signatures numériques XML [RFC3275]. Finalement, quand les OTI de FEC sont envoyées hors bande (par exemple, dans une description de session) ces OTI de FEC DEVRAIENT être protégées, par exemple, en signant numériquement l'objet qui inclut ces OTI de FEC.

Les mêmes considérations concernant les aspects de gestion de clé s'appliquent aussi ici.

10. Considérations relatives à l'IANA

Les valeurs d'identifiants de codage de FEC et d'identifiants d'instance de FEC font l'objet d'un enregistrement par l'IANA. Pour des lignes directrices générales sur l'application des considérations relatives à l'IANA au présent document, voir la [RFC5052].

Le présent document alloue l'identifiant de codage pleinement spécifié de FEC de 2 sous l'espace de noms "ietf:rmt:fec:encoding" à "Codes Reed-Solomon sur GF(2^^m)".

Le présent document alloue l'identifiant de codage pleinement spécifié de FEC de 5 sous l'espace de noms "ietf:rmt:fec:encoding" à "Codes Reed-Solomon sur GF(2^^8)".

Le présent document alloue l'identifiant d'instance de FEC de 0 limité par l'identifiant de codage sous spécifié de FEC de 129 aux "Codes Reed-Solomon sur GF(2^^8)". Plus précisément, sous le sous espace de noms "ietf:rmt:fec:encoding:instance" qui est limité par le "ietf:rmt:fec:encoding" appelé "Codes de FEC systématique de petit bloc", le présent document alloue l'identifiant d'instance de FEC de 0 aux "Codes Reed-Solomon sur GF(2^^8)".

11. Remerciements

Les auteurs tiennent à remercier Brian Adamson, Igor Slepchin, Stephen Kent, Francis Dupont, Elwyn Davies, Magnus Westerlund, et Alfred Hoenes de leurs précieux commentaires.Les auteurs tiennent aussà à remercier Luigi Rizzo de ses commentaires et du concept de codec de référence Reed-Solomon.

12. Références

12.1 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)

[RFC5052] M. Watson et autres, "Bloc de construction de la correction d'erreur directe (FEC)", août 2007. (*Remplace* RFC3452) *(P.S.)*

[RFC5445] M. Watson, "Schémas de base de correction d'erreur directe (FEC)", mars 2009. (*Remplace* RFC3452, RFC3695) *(P. S.)*

12.2 Références pour information

[GO94] Gohberg, I. and V. Olshevsky, "Fast algorithms with preprocessing for matrix-vector multiplication problems", Journal of Complexity, pp. 411-427, vol. 10, 1994.

[MWS77] Mac Williams, F. and N. Sloane, "The Theory of Error Correcting Codes", North Holland, 1977.

[RFC2104] H. Krawczyk, M. Bellare et R. Canetti, "HMAC : Hachage de clés pour l'authentification de message", février 1997, DOI 10.17487/RFC2104.

[RFC3275] D. Eastlake 3rd , J. Reagle, D. Solo, "Syntaxe et traitement de signature en langage de balisage extensible (XML)", mars 2002. *(D.S.)*

[RFC3447] J. Jonsson et B. Kaliski, "Normes de cryptographie à clés publiques (PKCS) n° 1 : Spécifications de la cryptographie RSA version 2.1", février 2003. *(Obsolète, remplacée par* RFC8017*) (Information)*

[RFC3453] M. Luby et autres, "Utilisation de la correction d'erreur directe (FEC) en diffusion groupée fiable", décembre 2002. *(Info.)*

[RFC4082] A. Perrig et autres, "Authentification de flux tolérante aux pertes en temps efficace (TESLA) : Introduction à la transformation d'authentification de source de diffusion groupée", juin 2005. *(Information)*

[RFC4303] S. Kent, "Encapsulation de charge utile de sécurité dans IP (ESP)", décembre 2005. (*Remplace* RFC2406) *(P.S.)*

[RFC5053] M. Luby et autres, "Schéma de correction d'erreur directe du code Raptor pour la livraison d'objets", octobre 2007. *(P.S.)*

[RFC5170] V. Roca et autres, "Schémas d'escalier et de triangle de vérification de parité à basse densité (LDPC) pour la correction d'erreur directe (FEC)", juin 2008. *(P.S.)*

[RFC5740] B. Adamson, C. Bormann, M. Handley, J. Macker, "Protocole de transport de diffusion groupée fiable orientée NACK (NORM)", novembre 2009. *(Remplace* RFC3940 ; *P. S.*)

[RFC5775] M. Luby, M. Watson, L. Vicisano, "Instance de protocole de codage en couches asynchrone (ALC)", avril 2010. (*Remplace* RFC3450). *(P. S.)*

[RFC6726] T. Paila, et autres, "FLUTE – Livraison de fichier sur transport unidirectionnel", novembre 2012. (*Remplace la* RFC3926) *(P.S.)*

[Rizzo97] Rizzo, L., "Effective Erasure Codes for Reliable Computer Communication Protocols", ACM SIGCOMM Computer Communication Review Vol.27, No.2, pp.24-36, avril 1997.

[RS-codec] Rizzo, L., "Reed-Solomon FEC codec", disponible à http://info.iet.unipi.it/~luigi/vdm98/vdm980702.tgz et reflétée à http://planete-bcast.inrialpes.fr/, version révisée de juillet 1998.

Adresse des auteurs

Jerome Lacan
ISAE/LAAS-CNRS
1, place Emile Blouin
Toulouse 31056
France
mél : jerome.lacan@isae.fr
URI : http://pagespro.isae.fr/jerome-lacan/

Vincent Roca
INRIA
655, av. de l'Europe
Inovallee; Montbonnot
ST ISMIER cedex 38334
France
mél : vincent.roca@inria.fr
URI : http://planete.inrialpes.fr/people/roca/

Jani Peltotalo

Sami Peltotalo

Tampere University of Technology
P.O. Box 553 (Korkeakoulunkatu 1)
Tampere FIN-33101
Finlande
mél : jani.peltotalo@tut.fi
URI : http://mad.cs.tut.fi/

Tampere University of Technology
P.O. Box 553 (Korkeakoulunkatu 1)
Tampere FIN-33101
Finlande
mél : sami.peltotalo@tut.fi
URI : http://mad.cs.tut.fi/