Groupe de travail Réseau
Request for Comments : 3797
RFC rendue obsolète : 2777
Catégorie : Information

D. Eastlake 3rd, Motorola Laboratories
juin 2004

Traduction Claude Brière de L'Isle

Choix aléatoire publiquement vérifiable du comité des nominations (NomCom)

Résumé
Le présent document décrit une méthode pour faire des choix aléatoires d'une façon telle que la nature non biaisée du choix soit publiquement vérifiable. Comme exemple, on utilise la sélection des membres votant du comité des nominations de l'IETF (NomCom, *Nominations Committee*) à partir du réservoir de volontaires éligibles. Des techniques similaires seraient applicables à d'autres cas.

Table des Matières

1. Introduction

Selon les règles de l'IETF, dix personnes sont choisies chaque années au hasard parmi les volontaires éligibles pour être les membres votants du comité des nominations de l'IETF (NomCom). Le NomCom désigne les membres du groupe de pilotage de l'ingénierie de l'Internet (IESG, *Internet Engineering Steering Group*) et du bureau de l'architecture de l'Internet (IAB, *Internet Architecture Board*) comme décrit dans la [RFC3777]. Le nombre de volontaires éligibles ces dernières années était d'environ cent.

Il est très souhaitable que le choix aléatoire des membres votants du NomCom soit fait d'une façon inattaquable afin qu'aucune accusation raisonnable de biais ou de favoritisme ne puisse être portée. C'est autant pour la protection de l'administrateur de la sélection (actuellement, le président nommé non votant du NomCom) contre des soupçons de biais que pour la protection de l'IETF.

Une méthode telle que l'information du public permet à tout un chacun de vérifier que le caractère aléatoire du choix satisfera

ce critère. Le présent document donne un exemple d'une telle méthode.

La méthode, sous la forme où elle apparaît dans la RFC2777, a aussi été utilisée par l'IANA en février 2003 pour déterminer le préfixe ACE pour les noms de domaines internationalisés [RFC3490] de façon à éviter une avalanche de plaintes.

2. Flux général d'un processus publiquement vérifiable

Un choix des membres du NomCom vérifiable publiquement comme non biaisé ou un choix similaire pourrait suivre les trois étapes suivantes.

2.1 Détermination du réservoir

D'abord, on détermine le réservoir à partir duquel le choix doit être fait, comme prévu dans la [RFC3777] ou ses successeurs.

Les volontaires sont sollicités par l'administrateur de la sélection. Leurs noms sont alors passés au secrétariat de l'IETF pour vérifier leur éligibilité. (Les critères d'éligibilité actuels se rapportent à la présence aux réunions de l'IETF, dont les archives sont conservées par le secrétariat.) La liste complète des volontaires éligibles est rendue publique assez tôt pour qu'un délai raisonnable puisse être laissé afin de résoudre tout conflit quant à qui devrait figurer dans le réservoir.

2.2 Publication de l'algorithme

L'algorithme exact à utiliser, incluant les sources publiques future d'aléa, est rendu public. Par exemple, les membres de la liste finale des volontaires éligibles sont ordonnés en les numérotant publiquement, des sources d'aléa publiques futures comme des loteries gouvernementales sont spécifiées, et un algorithme exact est spécifié par lequel les volontaires éligibles sont choisis sur la base d'une forte fonction de hachage [RFC1750] de ces futures sources d'aléa.

2.3 Publication de la sélection

Lorsque les sources d'aléa pré-spécifiées produisent leur résultat, ces valeurs plus un résumé de l'exécution de l'algorithme de choix devraient être annoncées afin que tous puissent vérifier que les valeurs de source d'aléa correctes ont été utilisées et que l'algorithme est correctement exécuté. L'algorithme peut fonctionner pour choisir, de façon ordonnée, un plus grand nombre que ce qui est réellement nécessaire afin que si un de ceux qui sont choisis doit passer son tour ou être remplacé pour une raison quelconque, un ensemble ordonné de choix de remplacement supplémentaires soit disponible. Un délai de réclamation sur la façon dont l'algorithme a fonctionné, ou avec de mauvaises entrées, ou n'a pas été exécuté de façon équitable, doit être spécifié pour finaliser le résultat et fournir une sélection stable.

3. Aléa

L'essentiel de la nature non biaisée de la sélection est qu'elle se fonde d'une façon exacte et prédéterminée sur des informations aléatoires qui seront révélées plus tard et ne peuvent donc pas être connues de la personne qui spécifie l'algorithme. Ces informations aléatoires seront utilisées pour contrôler la sélection. Les informations aléatoires doivent être telles qu'elles seront révélées publiquement et sans ambiguïté le moment venu.

Les sources aléatoires ne doivent pas inclure des choses que toute personne raisonnable pourrait croire sous le contrôle ou l'influence de l'IETF ou de ses composants, comme des statistiques de participation aux réunions de l'IETF, le nombre des documents produits, ou des choses de ce genre.

3.1 Sources d'aléa

Des exemples de bonnes informations à utiliser sont les numéros de loterie gagnants pour des tirages spécifiques de certaines loteries publiques. En particulier pour les loteries faites par le gouvernement, un grand soin est pris pour voir si ils sortent à temps et produisent des quantités aléatoires. Même dans le cas improbable où untirage devrait être arrangé, se serait presque certainement en relation avec un gain d'argent à la loterie, pas avec l'utilisation par l'IETF.

D'autres possibilités sont des choses comme la balance journalière du trésor public US sur un certain jour, le volume des échanges sue la bourse de New York un certain jour, etc. (Cependant, le code de référence donné plus loin ne va pas traiter des entiers très grands.) Des événement sportifs peuvent aussi être utilisés. (L'expérience a montré que les prix et/ou volumes des marchés sont une mauvaise source de données non ambiguës à cause des suspensions des cours, des fusions de compagnies, des partages des marchés multiples, etc.) Dans tous les cas, un grand soin doit être apporté à spécifier exactement les quantités présumées aléatoires et ce qui sera fait si leur production est annulée, retardée ou avancée.

Il est important que la dernière source d'aléa, chronologiquement, produise une quantité substantielle de l'entropie nécessaire. Si la plus grande partie de l'aléa vient de la première des sources spécifiées, et si quelqu'un a une influence même limitée sur la source finale, il pourrait faire une analyse exhaustive et exercer une influence telle qu'elle biaise la sélection dans la direction qu'il veut. Donc, il est préférable que la dernière source soit une source particulièrement forte et non biaisée d'une grande quantité d'aléa comme celle d'une loterie gouvernementale.

Il vaut mieux ne pas utiliser trop de sources différentes. Toute source supplémentaire augmente la probabilité qu'une ou plusieurs sources soient retardée, annulée, ou complètement biaisée d'une certaine façon, appelant des dispositions contingentes ou, pire que tout, créant une situation qui n'était pas prévue. Cela exigerait un jugement arbitraire de la part de l'administrateur de la sélection, détruisant son caractère aléatoire, ou un nouveau tour avec des sources différentes, causant des retards. Trois ou quatre serait un bon nombre de sources. Dix est trop.

3.2 Biais

Certaines des sources d'aléa produisent des données qui ne sont pas d'une distribution uniforme. C'est certainement vrai des volumes, des prix, et des résultats des courses de chevaux, par exemple. Cependant, l'utilisation d'une fonction forte de mélange [RFC1750] va extraire l'entropie disponible et produire une valeur de hachage dont les bits, le reste modulo un petit diviseur, etc., dévient seulement d'une quantité insignifiante d'une distribution uniforme.

3.3 Entropie nécessaire

Nous allons choisir N éléments sans remplacement dans une population de P éléments. Le nombre de différentes façons de faire cela est comme suit, où "!" représente la fonction factorielle :

$$\frac{P!}{N! * (P - N)!}$$

Faire cela de façon complètement aléatoire exige autant de bits aléatoires que le logarithme de base 2 de cette quantité. Un échantillon de calcul du nombre approximatif de bits aléatoires pour un choix complètement aléatoire de dix membres du NomCom à partir de diverses tailles de réservoir est donné ci-dessous :

Choix aléatoire de dix éléments du réservoir

Taille du réservoir :	20	25	30	35	40	50	60	75	100	120
Bits nécessaires :	18	22	25	28	30	34	37	40	44	47

Utiliser un nombre de bits inadéquat signifie que tous les ensembles possibles de dix éléments choisis ne seront pas disponibles. Pour une quantité d'entropie substantiellement inadéquate, il pourrait y avoir une corrélation significative entre le choix de deux membres différents du réservoir, par exemple. Cependant, en pratique, pour des tailles de réservoir qu'on a des chances de rencontrer dans le choix des membres du NomCom de l'IETF, 40 bits d'entropie devrait toujours être adéquat. Même si il y a un grand réservoir et si plus de bits sont nécessaires pour une aléation parfaite, 40 bits d'entropie vont assurer que seule une déviation insignifiante d'un choix complètement aléatoire pour la différence de probabilité de choix des différents membres du réservoir, la corrélation entre le choix de toute paire de membres du réservoir, etc.

Un hachage MD5 [RFC1321] a 128 bits et ne peut donc pas produire plus que ce nombre de bits d'entropie. Cependant, c'est plus de trois fois ce qui sera probablement jamais nécessaire pour le choix des membres du NomCom de l'IETF. Un hachage encore plus fort, comme SHA-1 [RFC3174], peut être utilisé si désiré.

4. Suggestion d'un algorithme précis

Il est important qu'un algorithme précis puisse être donné pour mélanger les sources d'aléa spécifiées et faire le choix fondé sur

elles. Les sources suggérées ci-dessus produisent soit un seul nombre positif (c'est-à-dire, le volume des échange de la Bourse de New York en milliers d'actions) ou un petit ensemble de nombres positifs (de nombreuses loteries fournissent six chiffres dans la gamme de 1 à 40 ou quelque chose comme cela, un événement sportif pourrait produire les scores de deux équipes, etc.). Une suggestion d'algorithme précis est celle-ci :

1. Pour chaque source qui produit plusieurs valeurs numériques, on représente chaque valeur comme un nombre décimal terminé par une virgule (ou avec une virgule qui sépare l'entier de la partie fractionnelle) sans zéro en tête (sauf pour un seul zéro si la partie entière est zéro) et sans zéro en queue après la virgule.

2. Ordonner les valeurs provenant de chaque source de la plus petite à la plus grande et les enchaîner en mettant en suffixe au résultat un "/". Pour chaque source qui produit un seul nombre, le représenter simplement comme ci-dessus avec le suffixe "/". (Ce tri est nécessaire parce que le même résultat de loterie, par exemple, est parfois rapporté dans l'ordre où les numéros ont été tirés et parfois en ordre numérique croissant et des choses comme les résultats de deux équipes sportives qui jouent à un jeu n'ont pas d'ordre inhérent.)

3. On a à ce moment une chaîne pour chaque source, disons s1/, s2/, ... On enchaîne ces chaînes dans un ordre pré spécifié, l'ordre dans lequel les sources ont été énumérées si ce n'est pas autrement spécifié, et on représente chaque caractère par son code ASCII [ASCII] produisant "s1/s2/.../".

On produit alors une séquence de valeurs aléatoires dérivée d'un fort mélange de ces sources en calculant le hachage MD5 [RFC1321] de cette chaîne préfixée et suffixée avec une séquence de deux octets tout à zéro pour la première valeur, la chaîne préfixée et suffixée par 0x0001 pour la seconde valeur, etc., en traitant les deux octets comme un compteur gros boutien. Traiter chacune de ces valeurs dérivées de résultat MD5 "aléatoire" comme un entier positif gros boutien multi précision de 128 bits.

On ordonne ensuite complètement le réservoir des volontaires inscrits comme suit : si il y a P volontaires, on choisit le premier en divisant la première valeur aléatoire dérivée par P et on utilise le reste plus un comme position de celui qui est choisi dans la liste publiée. On choisit le second en divisant la seconde valeur aléatoire dérivée par P-1 et on utilisé le reste plus un comme position dans la liste avec la première personne choisie éliminée. Etc.

Il est FORTEMENT recommandé que les sources aléatoires alphanumériques soient évitées à cause de la trop grande difficulté de les canoniser d'une façon répétable indépendante ; cependant, si on choisit d'ignorer cet avis et d'utiliser une ou des sources ASCII ou d'alphabet romain similaire, toutes les espaces, la ponctuation, les accents, et les caractères spéciaux devraient être retirés et toutes les lettres mises en majuscules. Cela va laisser seulement une séquence ininterrompue de lettres A-Z et de chiffres 0-9 qui peuvent être traités comme un nombre canonique décrit ci-dessus et suffixé avec un "./". Si on choisit de ne pas ignorer mais de se moquer de façon flagrante de cet avis et d'essayer d'utiliser un texte internationalisé encore plus complexe et difficile à canoniser, comme de l'UNICODE, on vous souhaite bon courage.

5. Traitement de problèmes concrets

Dans la réalité, des problèmes peuvent surgir en suivant les étapes et le cours mentionnés dans les sections 2 à 4 ci-dessus. Certains problèmes qui se sont réellement produits sont décrits ci-dessous avec des recommandations pour les traiter.

5.1 Incertitude quant au réservoir

Tous les efforts raisonnables devraient être faits pour s'assurer que le réservoir publié à partir duquel est fait le choix est constitué de personnes certaines et éligibles. Cependant, en particulier avec des délais restreints ou peut-être si quelqu'un qui prétend être volontaire et éligible ne s'est pas résolu à la date limite, ou si la détermination que quelqu'un n'est pas éligible se produit après la publication du réservoir, il se peut qu'il y ait encore des incertitudes.

La meilleure façon de traiter cela est de s'en tenir au programme annoncé, Y COMPRIS dans le réservoir publié de tous ceux dont l'éligibilité est incertaine et de conserver la liste du réservoir publié INCHANGÉE après sa publication. Si quelqu'un dans le réservoir est ensuite choisi par l'algorithme et les entrées aléatoires mais qu'il est déterminé qu'il est inéligible, on peut le sauter et l'algorithme va faire un tour de plus pour faire un choix supplémentaire. Donc; l'incertitude n'affecte qu'une sélection et en général pas plus d'un maximum de U sélections lorsque il y a U membres du réservoir qui sont incertains.

D'autres plans d'action sont bien pires. L'insertion ou la suppression réelle d'entrées dans le réservoir après sa publication change la longueur de la liste et perturbe totalement les sélectionnés, changeant éventuellement toute la sélection. Une insertion dans le réservoir soulève des questions sur où insérer : au début, à la fin, dans l'ordre alphabétique, ... tous les choix de ce genre par l'administrateur de la sélection après que les numéros aléatoires sont connus détruit la possibilité de vérification par le public que le choix est équitable. Même si c'est fait avant que soient connus les numéros aléatoires, une telle

manipulation de la liste après sa publication ne sent pas bon. Il doit y avoir des délais clairs fixés publiquement et quelqu'un qui conteste son absence du réservoir après le délai publié devrait voir sa réclamation automatiquement rejetée comme tardive.

5.2 Ambiguïtés d'aléa

Les meilleurs efforts ont été faits de bonne foi pour spécifier des sources d'aléa précises et sans ambiguïté. Ces sources ont été rendues publiques à l'avance et il n'y a pas eu d'objection contre elles. Cependant, il est arrivé que lorsque vient le temps d'obtenir et de les utiliser, le monde réel se met en travers et il n'est plus aussi clair de savoir quelles données utiliser. Des problèmes sont survenus en particulier avec les cours de la bourse, les volumes, et les taux ou indices des échanges financiers. Si les volumes qui étaient publiés en milliers sont publiés en centaines, on a un problème d'arrondi. Les prix qui étaient cités en fractions ou avec des décimales peuvent changer de l'un à l'autre. Si on veut tenir compte de toutes les contingences qui sont intervenues dans le passé, on peut être frappé par une nouvelle. Quand cette sorte de chose arrive, il est généralement trop tard pour annoncer de nouvelles sources, action qui pourrait soulever des soupçons contre vous. Le seul plan d'action est de faire un choix raisonnable au sein de l'ambiguïté et de se reposer sur la confiance en la bonne foi de l'administrateur de la sélection. Avec un peu d'attention, de tels cas devraient être extrêmement rares.

Sur la base de ces expériences, on recommande encore que les numéros de loterie publique ou des choses de ce genre soient utilisés comme entrées d'aléa et d'éviter les prix et volumes du marché.

6. Exemple élaboré

Supposons que soit publiée la liste ordonnées suivante de 25 volontaires éligibles avant la sélection :

1. John	11. Pollyanna	21. Pride
2. Mary	12. Pendragon	22. Sloth
3. Bashful	13. Pandora	23. Envy
4. Dopey	14. Faith	24. Anger
5. Sleepy	15. Hope	25. Kasczynski
6. Grouchy	16. Charity	
7. Doc	17. Lee	
8. Sneazy	18. Longsuffering	
9. Handsome	19. Chastity	
10. Cassandra	20. Smith	

Supposons la liste ordonnée (exemple factice) de sources d'aléa suivante :

1. Le numéro de la loterie quotidienne du royaume de l'état d'Alphaland pour le premier novembre 2004 traité comme un seul entier de quatre chiffres.
2. Les numéros des chevaux gagnants à Hialeia pour toutes les courses du premier jour après le 13 octobre 2004 sur lequel au moins deux courses sont courues.
3. Les six numéros gagnants de la loterie d'état de la république populaire de Betastani (en ignorant le septième numéro "supplémentaire") du 1 novembre 2004.

L'aléa hypothétique produit publiquement est :
Source 1 : 9319
Source 2 : 2, 5, 12, 8, 10
Source 3 : 9, 18, 26, 34, 41, 45

Chaîne clé résultante : 9319./2.5.8.10.12./9.18.26.34.41.45./

Le tableau ci-dessous donne les valeurs en hexadécimal du MD5 de la chaîne clé ci-dessus entouré par une chaîne de deux octets qui sont successivement 0x0000, 0x0001, 0x0002, jusqu'à 0x0010 (16 en décimal). Le diviseur pour la taille du numéro du réservoir restant à chaque étape est donné et l'indice du sélectionné selon le numéro d'origine de ceux du réservoir.

indice	**valeur hexadécimale du MD5**	**diviseur**	**sélectionné**
1	990DD0A5692A029A98B5E01AA28F3459	25	-> 17 <-
2	3691E55CB63FCC37914430B2F70B5EC6	24	-> 7 <-
3	FE814EDF564C190AC1D25753979990FA	23	-> 2 <-
4	1863CCACEB568C31D7DDBDF1D4E91387	22	-> 16 <-
5	F4AB33DF4889F0AF29C513905BE1D758	21	-> 25 <-
6	13EAEB529F61ACFB9A29D0BA3A60DE4A	20	-> 23 <-
7	992DB77C382CA2BDB9727001F3CDCCD9	19	-> 8 <-
8	63AB4258ECA922976811C7F55C383CE7	18	-> 24 <-

9	DFBC5AC97CED01B3A6E348E3CC63F40D	17	-> 19 <-
10	31CB111C4A4EBE9287CEAE16FE51B909	16	-> 13 <-
11	07FA46C122F164C215BBC72793B189A3	15	-> 22 <-
12	AC52F8D75CCBE2E61AFEB3387637D501	14	-> 5 <-
13	53306F73E14FC0B2FBF434218D25948E	13	-> 18 <-
14	B5D1403501A81F9A47318BE7893B347C	12	-> 9 <-
15	85B10B356AA06663EF1B1B407765100A	11	-> 1 <-
16	3269E6CE559ABD57E2BA6AAB495EB9BD	10	-> 4 <-

Les dix premiers sélectionnés résultants , dans l'ordre de la sélection sont :
1. Lee (17) 2. Doc (7) 3. Mary (2) 4. Charity (16) 5. Kasczynski (25) 6. Envy (23) 7. Sneazy (8) 8. Anger (24) 9. Chastity (19) 10. Pandora (13)
Si un d'eux se révèle inéligible ou décline de servir, le suivant serait Sloth, numéro 22.

7. Considérations pour la sécurité

Tous les soins devraient être apportés au choix des entrées d'aléa afin qu'on ne puisse pas raisonnablement soupçonner qu'elles sont sous le contrôle de l'administrateur. Les lignes directrices données ci-dessus d'utiliser un petit nombre d'entrées avec une quantité substantielle d'entropie à partir de la dernière devraient être appliquées. Un soin égal doit être apporté à l'exécution équitable de l'algorithme choisi avec les valeurs d'entrée désignées.

La publication des résultats et une fenêtre d'une semaine ou deux pour que la communauté intéressée duplique les calculs devraient donner une assurance raisonnable contre toute manipulation de la mise en œuvre.

8. Code de référence

Ce code utilise le code de référence MD5 provenant de la [RFC1321] de RSA Data Security, Inc. "Algorithme MD5 de résumé de message". La portion du code qui traite de numéros multiples à virgule flottante a été écrite par Matt Crawford. Le code d'origine dans la RFC2777 ne pouvait traiter des réservoirs que jusqu'à 255 membres et a été étendu à 2**16-1 par Erik Nordmark. Ce code a été extrait de ce document, compilé, et essayé. Bien qu'aucune faute n'ait été trouvée, il est possible qu'il s'en manifeste avec un autre compilateur sur un autre système.

```
/*****************************************************************
 *
 * Code de référence pour un "choix aléatoire publiquement vérifiable"
 *         Donald E. Eastlake 3rd
 *            février 2004
 *
 *****************************************************************/
#include <limits.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* D'après la RFC 1321 */
#include "global.h"
#include "MD5.h"

/* prototypes locaux */
int longremainder ( unsigned short divisor, unsigned char dividend[16] );
long int getinteger ( char *string );
double NPentropy ( int N, int P );


/* limité à 16 entrées de jusqu'à seize entiers de chaque réservoir limité à 2**8-1 étendu à 2**16-1 par Erik Nordmark */
/*****************************************************************/

main ()
{
```

```
   int        i, j,  k, k2, err, keysize, selection, usel;
   unsigned short   remaining, *selected;
   long int    pool, temp, array[16];
   MD5_CTX     ctx;
   char        buffer[257], key [800], sarray[16][256];
   unsigned char    uc16[16], unch1, unch2;
   pool = getinteger ( "Type size of pool:\n" );
   if ( pool > 65535 )

      {
      printf ( "Pool too big.\n" );
      exit ( 1 );
      }
   selected = (unsigned short *) malloc ( (size_t)pool );
   if ( !selected )
      {
      printf ( "Out of memory.\n" );
      exit ( 1 );
      }
   selection = getinteger ( "Type number of items to be selected:\n" );
   if ( selection > pool )
      {
      printf ( "Pool too small.\n" );
      exit ( 1 );
      }
   if ( selection == pool )
      printf ( "All of the pool is selected.\n" );
   else
      {
      err = printf ( "Approximately %.1f bits of entropy needed.\n",
                  NPentropy ( selection, pool ) + 0.1 );
      if ( err <= 0 ) exit ( 1 );
      }
   for ( i = 0, keysize = 0; i < 16; ++i )
      {
      if ( keysize > 500 )
         {
         printf ( "Too much input.\n" );
         exit ( 1 );
         }
/* obtenir les entrées "aléatoires". Les donner à l'usager afin qu'il sache si il y a troncature ou d'autres dysfonctionnement.  */
      err = printf (
         "\nType #%d randomness or 'end' followed by new line.\n"
         "Up to 16 integers or the word 'float' followed by up\n"
         "to 16 x.y format reals.\n", i+1 );
      if ( err <= 0 ) exit ( 1 );
      gets ( buffer );
      j = sscanf ( buffer,
            "%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld%ld",
         &array[0], &array[1], &array[2], &array[3],
         &array[4], &array[5], &array[6], &array[7],
         &array[8], &array[9], &array[10], &array[11],
         &array[12], &array[13], &array[14], &array[15] );
      if ( j == EOF )
         exit ( j );
      if ( !j )
         if ( buffer[0] == 'e' )
            break;
         else
            {                                          /* code de virgule flottante par Matt Crawford */
            j = sscanf ( buffer,
               "float %ld.%[0-9]%ld.%[0-9]%ld.%[0-9]%ld.%[0-9]"
               "%ld.%[0-9]%ld.%[0-9]%ld.%[0-9]%ld.%[0-9]"
```

```
                "%ld.%[0-9]%ld.%[0-9]%ld.%[0-9]%ld.%[0-9]"
                "%ld.%[0-9]%ld.%[0-9]%ld.%[0-9]%ld.%[0-9]",
                &array[0], sarray[0], &array[1], sarray[1],
                &array[2], sarray[2], &array[3], sarray[3],
                &array[4], sarray[4], &array[5], sarray[5],
                &array[6], sarray[6], &array[7], sarray[7],
                &array[8], sarray[8], &array[9], sarray[9],
                &array[10], sarray[10], &array[11], sarray[11],
                &array[12], sarray[12], &array[13], sarray[13],
                &array[14], sarray[14], &array[15], sarray[15] );
            if ( j == 0 || j & 1 )
                printf ( "Bad format." );
            else {
                for ( k = 0, j /= 2; k < j; k++ )
                {                                       /* supprimer les zéros en queue */
                for ( k2=strlen(sarray[k]); sarray[k][--k2]=='0';)
                    sarray[k][k2] = '\0';
                err = printf ( "%ld.%s\n", array[k], sarray[k] );
                if ( err <= 0 ) exit ( 1 );
                keysize += sprintf ( &key[keysize], "%ld.%s",
                            array[k], sarray[k] );
                }
                keysize += sprintf ( &key[keysize], "/" );
                }
            }
        else
            {   /* trier les valeurs, un algorithme pas très efficace */
            for ( k2 = 0; k2 < j - 1; ++k2 )
                for ( k = 0; k < j - 1; ++k )
                    if ( array[k] > array[k+1] )
                        {
                        temp = array[k];
                        array[k] = array[k+1];
                        array[k+1] = temp;
                        }
            for ( k = 0; k < j; ++k )
                { /* imprimer pour que l'usager vérifie */
                err = printf ( "%ld ", array[k] );
                if ( err <= 0 ) exit ( 1 );
                keysize += sprintf ( &key[keysize], "%ld.", array[k] );
                }
            keysize += sprintf ( &key[keysize], "/" );
            }
        }                                               /* fin de i */

    /* on a obtenu toutes les entrées, on produit maintenant le résultat */
        err = printf ( "Key is:\n %s\n", key );
        if ( err <= 0 ) exit ( 1 );
        for ( i = 0; i < pool; ++i )
            selected [i] = (unsigned short)(i + 1);
        printf ( "index       hex value of MD5       div  selected\n" );
        for (   usel = 0, remaining = (unsigned short)pool;
                usel < selection;
                ++usel, --remaining )
            {
            unch1 = (unsigned char)usel;
            unch2 = (unsigned char)(usel>>8);
    /* préfixe/suffixe étendu de deux octets par Donald Eastlake */
            MD5Init ( &ctx );
            MD5Update ( &ctx, &unch2, 1 );
            MD5Update ( &ctx, &unch1, 1 );
            MD5Update ( &ctx, (unsigned char *)key, keysize );
            MD5Update ( &ctx, &unch2, 1 );
```

```
        MD5Update ( &ctx, &unch1, 1 );
        MD5Final ( uc16, &ctx );
        k = longremainder ( remaining, uc16 );
/* printf ( "Remaining = %d, remainder = %d.\n", remaining, k ); */
        for ( j = 0; j < pool; ++j )
           if ( selected[j] )
              if ( --k < 0 )
                 {
                 printf ( "%2d  "
    "%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X  "
    "%2d  -> %2d <-\n",
    usel+1, uc16[0],uc16[1],uc16[2],uc16[3],uc16[4],uc16[5],uc16[6],
    uc16[7],uc16[8],uc16[9],uc16[10],uc16[11],uc16[12],uc16[13],
    uc16[14],uc16[15], remaining, selected[j] );
                 selected[j] = 0;
                 break;
                 }
     }
     printf ( "\nDone, type any character to exit.\n" );
     getchar ();
     return 0;
     }

/* invite pour une entrée d'entier positif non zéro */
     /*****************************************************************/
     long int getinteger ( char *string )
     {
     long int     i;
     int          j;
     char    tin[257];
     while ( 1 )
     {
     printf ( string );
     printf ( "(or 'exit' to exit) " );
     gets ( tin );
     j = sscanf ( tin, "%ld", &i );
     if (    ( j == EOF )
        ||  ( !j && ( ( tin[0] == 'e' ) || ( tin[0] == 'E' ) ) )
           )
        exit ( j );
     if ( ( j == 1 ) &&
         ( i > 0 ) )
        return i;
     }                                                  /* fin de while */
     }

/* prendre le reste de la division d'un entier non signé de 16 octets par un petit nombre positif */
     /*****************************************************************/
     int longremainder ( unsigned short divisor, unsigned char dividend[16] )
     {
     int i;
     long int kruft;
     if ( !divisor )
        return -1;
     for ( i = 0, kruft = 0; i < 16; ++i )
        {
        kruft = ( kruft << 8 ) + dividend[i];
        kruft %= divisor;
        }
     return kruft;
     }                                                  /* fin du reste de longueur */

/* calculer combien de bits d'entropie il faut pour choisir N parmi P */
```

```
   /*****************************************************************/
   /*            P!
     log  ( ----------------- )
        2    N! * ( P - N )!
   */

   double NPentropy ( int N, int P )
   {
   int        i;
   double     result = 0.0;

   if (    ( N < 1 )                          /* ne rien choisir du tout ? */
     ||   ( N >= P )                          /* choisir tout le réservoir ou plus ? */
     )
      return 0.0;                             /* cas limite */
   for ( i = P; i > ( P - N ); --i )
      result += log ( i );
   for ( i = N; i > 1; --i )
      result -= log ( i );
/* diviser par [ log (base e) de 2 ] pour convertir en bits */
   result /= 0.69315;
   return result;
   }                                          /* fin de NPentropy */
```

Appendice A Historique du choix des membres du NomCom

À titre de référence, voici la liste des présidents et des techniques de choix des membres du comité des nominations de l'IETF jusqu'à présent :

Année	**Président**	**méthode de choix**
1993/1994	Jeff Case	Clergy
1994/1995	Fred Baker	Clergy
1995/1996	Guy Almes	Clergy
1996/1997	Geoff Huston	Épouse
1997/1998	Mike St.Johns	Algorithme
1998/1999	Donald Eastlake 3rd	RFC2777
1999/2000	Avri Doria	RFC2777
2000/2001	Bernard Aboba	RFC2777
2001/2002	Theodore Ts'o	RFC2777
2002/2003	Phil Roberts	RFC2777
2003/2004	Rich Draves	RFC2777

Clergy = Les noms sont écrits sur une feuille de papier, placés dans un réceptacle, et un membre du clergé tire les membres du NomCom.

Épouse = Comme pour Clergy sauf que c'est l'épouse du président qui fait le tirage.

Algorithme = Choix algorithmique fondé sur des concepts similaires à ceux documentés dans la RFC2777 et ici même.

RFC2777 = Choix algorithmique utilisant l'algorithme et le code de référence fourni dans la RFC2777 (mais pas les exemples factices de sources d'aléa).

Appendice B Changements depuis la RFC 2777

Le présent document diffère de la [RFC 2777], la version précédente, par les principaux éléments suivants :

- La Section 5, sur les problèmes rencontrés actuellement dans l'usage de ces recommandations pour choisir un NomCom de l'IETF et comment les traiter a été ajoutée.

- Le code de l'algorithme de sélection a été modifié pour traiter des réservoirs comportant jusqu'à 2**16-1 éléments et le

préfixe et suffixe fondés sur un compteur enchaînés avec la chaîne de clé avant le hachage ont été étendus à deux octets.

- Une mention a été ajoutée que l'algorithme documenté ici a été utilisé par l'IANA pour choisir le préfixe ACE de nom de domaine internationalisé et quelques changements rédactionnels mineurs ont été faits.

- La liste de l'Appendice A a été mise à jour.

Remerciements

Matt Crawford et Erik Nordmark ont fait des contributions majeures au présent document. Des commentaires de Bernard Aboba, Theodore Ts'o, Jim Galvin, Steve Bellovin, et d'autres ont été incorporés.

Références

[ASCII] American National Standards Institute, "USA Standard Code for Information Interchange", X3.4, New York, 1968.

[RFC1321] R. Rivest, "Algorithme de résumé de message MD5", avril 1992. *(Information)*

[RFC1750] D. Eastlake 3rd et autres, "Recommandations d'aléa pour la sécurité", décembre 1994. *(Info., remplacée par la* RFC4086)

[RFC2777] D. Eastlake 3rd, "La sélection aléatoire de Nomcom est publiquement vérifiable", février 2000. (*Obs., voir* RFC3797) *(Info.)*

[RFC3174] D. Eastlake 3 et P. Jones, "Algorithme US de hachage sécurisé n° 1 (SHA1)", sept. 2001. *(Info, MàJ par 4634 et 6234)*

[RFC3490] P. Faltstrom et autres, "Internationalisation des noms de domaine dans les applications (IDNA)", mars 2003. (*Remplacée par les* RFC5890 et 5891, *P.S.*)

[RFC3777] J. Galvin, éd., "Processus de choix, confirmation et révocation de l'IAB et de l'IESG : fonctionnement des comités de désignation et de révocation", juin 2004. (*MàJ par* RFC5078) (BCP0010)

Adresse de l'auteur

Donald E. Eastlake, 3rd
Motorola Laboratories
155 Beaver Street
Milford, MA 01757 USA

téléphone : +1-508-786-7554(w)
+1-508-634-2066(h)
mél : Donald.Eastlake@motorola.com

Déclaration complète de droits de reproduction

Propriété intellectuelle
L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourraient 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.

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