Network Working Group D. Eastlake
INTERNET-DRAFT Futurewei Technologies
Obsoletes: 3797
Intended Status: Best Current Practice
Expires: March 14, 2023 September 15, 2022

Publicly Verifiable Nominations Committee (NomCom) Random Selection
<draft-eastlake-rfc3797bis-01.txt>

Abstract

This document describes a method for making random selections in such a way that the unbiased nature of the choice is publicly verifiable. It focuses on the selection of the voting members of the IETF Nominations Committee (NomCom) from the pool of eligible volunteers; however, similar or, in some cases, identical techniques could be and have been applied to other cases.

Status of This Memo

Copyright Notice

Provisions Relating to IETF Documents

Table of Contents

1. Introduction..4
1.1 Requirements Language...............................4

2. General Flow of a Publicly Verifiable Process...........5
2.1 Determination of the Pool..............................5
2.2 Publication of the Algorithm...........................5
2.3 The Selection..5

3. Randomness..7
3.1 Sources of Randomness..................................7
3.2 Skew...8
3.3 Entropy Needed...8

4. A Specific Algorithm for Initial Selection.............10

5. Extended Selection.....................................12

6. Handling Real World Problems...........................15
6.1 Uncertainty as to the Pool............................15
6.2 Randomness Ambiguities................................15

7. Fully Worked Example...................................17

8. Security Considerations................................19
9. IANA Considerations....................................19

10. Reference Code.......................................20

Normative References......................................26
Informative References....................................26

Appendix A: History of NomCom Member Selection............27
Appendix B: Changes from RFC 3797.........................28

Acknowledgements..29

1. Introduction

Under the IETF rules, each year ten people are randomly selected from among eligible volunteers to be the voting members of the IETF nominations committee (NomCom). The NomCom nominates members of the Internet Engineering Steering Group (IESG), the Internet Architecture Board (IAB), and other bodies as described in [RFC8713]. The number of eligible volunteers in the early years of the use of the NomCom mechanism was around 50 but in recent years has been around 200.

It is highly desirable that the random selection of the voting NomCom be done in an unimpeachable fashion so that no reasonable charges of bias or favoritism can be brought. This is as much for the protection of the selection administrator (currently, the appointed non-voting NomCom Chair) from suspicion of bias as it is for the protection of the IETF.

A method such that public information will enable any person to verify the randomness of the selection meets this criterion. This document specifies such a method.

This method, in the form it appeared in RFC 2777, was also used by IANA in February 2003 to determine the ACE prefix for Internationalized Domain Names ("xn--", [RFC5890]) so as to avoid claim jumping.

1.1 Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. General Flow of a Publicly Verifiable Process

A selection of NomCom members publicly verifiable as unbiased or similar selection could follow the three steps given in the subsections below: Determination of the Pool, Publication of the Algorithm, and Publication of the Selection.

2.1 Determination of the Pool

First, determine the pool from which the selection is to be made as provided in [RFC8713] or its successor.

Currently, volunteers are solicited by the selection administrator. Their names are then checked for eligibility. The full list of eligible volunteers is made public early enough that a reasonable time can be given to resolve any disputes as to who should be in the pool before a deadline at which the pool is frozen. Although no one can be added after this deadline, someone included in the list who should not have been can be easily handled as described later in this document.

2.2 Publication of the Algorithm

The exact algorithm to be used, including the public future sources of randomness, is made public. For example, the members of the final list of eligible volunteers are ordered by publicly numbering them, some public future sources of randomness such as government run lotteries are specified, and an exact algorithm is specified whereby eligible volunteers are selected based on a hash function [RFC4086] of these future sources of randomness.

2.3 The Selection

When the pre-specified sources of randomness produce their output, those values plus a summary of the execution of the algorithm for selection should be announced so that anyone can verify that the correct randomness source values were used and the algorithm properly executed. The algorithm SHOULD be run to select, in an ordered fashion, a larger number than are actually necessary so that if any of those selected need to be passed over or replaced for any reason, an ordered set of additional alternate selections is available. Under some circumstances, additional rounds of extended selection may be useful as specified in Section 5.

A cut off time for any complaint that the algorithm was run with the wrong inputs or not faithfully executed MUST be specified to finalize the output and provide a stable selection.

3. Randomness

The crux of the unbiased nature of the selection is that it is based in an exact, predetermined fashion on random information which will be revealed in the future and thus cannot be known to the person executing the algorithm. That random information will be used to control the selection. The random information MUST be such that it will be publicly and unambiguously revealed in a timely fashion.

3.1 Sources of Randomness

The random sources MUST NOT include anything that any reasonable person would believe to be under the control or influence of the selection administrator or the IETF or its components, such as IETF meeting attendance statistics, numbers of documents issued, or the like.

Examples of good information to use are winning lottery numbers for specified runnings of specified public lotteries. Particularly for major government run lotteries, great care is taken to see that they occur on time (or with minimal delay) and produce random quantities. Even in the unlikely case one was to have been rigged, it would almost certainly be in connection with winning money in the lottery, not in connection with IETF use. Other possibilities are such things as the daily balance in the US Treasury on a specified day, the volume of trading on the New York Stock exchange on a specified day, etc. (However, the reference code given below will not handle integers that are too large.) Sporting events can also be used. Experience has indicated that individual stock prices and/or volumes are a poor source of unambiguous data due trading suspensions, company mergers, delistings, splits, multiple markets, etc. In all cases, great care MUST be taken to specify exactly what quantities are being presumed random and what will be done if their issuance is cancelled, delayed, or advanced.

It is important that the last source of randomness, chronologically, produce a substantial amount of the entropy needed. If most of the randomness has come from the earlier of the specified sources, and someone has even limited influence on the final source, they might do an exhaustive analysis and exert such influence so as to bias the selection in the direction they wanted. Thus, it is RECOMMENDED that the last source be an especially strong and unbiased source of a large amount of randomness such as a major government run lottery.

It is best not to use too many different sources. Every additional source increases the probability that one or more sources might be delayed, cancelled, or just plain screwed up somehow, calling into play contingency provisions or, worst of all, creating an unanticipated situation. This would either require arbitrary judgment by the selection administrator, defeating the randomness of the selection, or a re-run with a new set of sources, causing much delay. Three would be a good number of randomness sources. More than six is way too many.

3.2 Skew

Some of the sources of randomness produce data that is not uniformly distributed. This is certainly true of volumes, prices, and horse race results, for example. However, use of a strong mixing function [RFC4086] will extract the available entropy and produce a hash value whose bits and whose remainder modulo a small divisor, deviate from a uniform distribution only by an insignificant amount.

3.3 Entropy Needed

What we are doing is selecting N items without replacement from a population of P items. The number of different ways to do this is as follows, where "!" represents the factorial function:

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

To do this in a completely random fashion requires as many random bits as the logarithm base 2 of that quantity. Some sample calculated approximate number of random bits for the completely random selection of 10 items (e.g., NomCom members) from various pool sizes are given below:

Random Selection of Ten Items From Pool

Pool size	40	60	80	100	125	150	175	200
Bits needed	30	36	41	44	47	50	52	54

Using a smaller number of bits means that not all of the possible sets of ten selected items would be available. For a substantially smaller amount of entropy, there could be a significant correlation between the selection of two different members of the pool, for example. However, as a practical matter, for pool sizes likely to be encountered in IETF NomCom membership selection, 40 bits of entropy should be more than adequate. Even if more bits are needed for perfect randomness, 40 bits of entropy will assure only an insignificant deviation from completely random selection for the difference in probability of selection of different pool members, the correlation between the selection of any pair of pool members, or the like.

The current US Power Ball lottery drawing has 23.5 bits of entropy in the five selected regular numbers and about 6 bits of entropy in the Power Ball. A four-digit daily numbers game drawing that selects four decimal digits has a bit over 13 bits of entropy.

An MD5 [RFC1321] hash has 128 bits of output and therefore can preserve no more than that number of bits of entropy. However, this is much more than what is likely to be needed for IETF NomCom membership selection. There have also been defects noted in MD5 for cryptographic usage [RFC6151] but these are not significant here. The hash function is just being used to, effectively, compress, deskew, and extract selections from the random input. For example, it would not hurt this process if a hash function was used for which it was easy to compute a pre-image.

4. A Specific Algorithm for Initial Selection

It is important that a precise algorithm be given for mixing the random sources being used and making the selection based thereon. Sources suggested above produce either a single positive number (i.e., NY Stock Exchange volume in thousands of shares) or a small set of positive numbers (many lotteries provide 6 numbers in the range of 1 through 65 or the like, a sporting event could produce the scores of two teams, etc.). A precise algorithm is as follows:

1. For each source producing one or more numeric values, each value is canonicalized by representing each value as a decimal number terminated by a period (or with a period separating the whole from the fractional part), without leading zeroes (except for a single leading zero if the integer part is zero), and without trailing zeroes on the part after the period. Some examples follow:

```
Input    Canonicalized
-----    -------------
  0        0.
  0.0      0.
 42       42.
  7.0      7.
013.      13.
  .420     0.42
12.34     12.34
1.2340     1.234
```

2. If a source produced multiple values, order those values from smallest to the largest. (This sorting is necessary because the same lottery results, for example, are sometimes reported in the order numbers were drawn and sometimes in numeric order and such things as the scores of two sports teams that play a game have no inherent order.)

3. If a source produced multiple values, concatenate them and suffix the result with a "/". If a source produced a single number, simply represent it as above with a suffix "/".

4. At this point you have a string for each source, say s1/, s2/, ... for source 1, source 2, ... Concatenate these strings in a pre-specified order, the order in which the sources were listed when they were announced if no other order is specified, and represent each character as its ASCII code [RFC20] producing "s1/s2/.../" as the random seed from which selection is derived.

5. Produce a sequence of random values derived from a mixing of these sources by calculating the MD5 hash [RFC1321] of the seed

specified in step 4 prefixed and suffixed with an all zeros two byte sequence for the first value, the string prefixed and suffixed by 0x0001 for the second value, etc., treating the two bytes as a big-endian counter. Treat each of these derived "random" MD5 output values as a positive 128-bit multiprecision big endian integer.

6. Finally impose a total pseudo-random ordering on the pool of listed items (e.g., NomCom volunteers) as follows: If there are P volunteers, select the first by dividing the first derived random value by P and using the remainder plus one as the position of the selectee in the published list. Select the second by dividing the second derived random value by P-1 and using the remainder plus one as the position in the list with the first selected person eliminated. And so on.

Any ambiguity in the above procedure is resolved by consulting the reference code below.

It is strongly RECOMMENDED that alphanumeric random sources be avoided due to the much greater difficulty in canonicalizing them in an independently repeatable fashion; however, if the administrator of the selection process chooses to ignore this advice and use an ASCII or similar Roman alphabet source or sources, all white space, punctuation, accents, and special characters should be removed and all letters set to upper case. This will leave only an unbroken sequence of letters A-Z and digits 0-9 which can be treated as a canonicalized single number above and suffixed with a "./". The administrator MUST NOT use even more complex and harder to canonicalize quantities such as complex numbers or UNICODE internationalized text.

5. Extended Selection

There may be reasons why one or more of the selected members of the pool need to be eliminated and further selections made. This is particularly true given the strong recommendation above that, in case of doubt or not-yet-resolved eligibility dispute, possible pool members should be left in the pool with the understanding that, in the event they are initially selected, they can be later eliminated. For the IETF NomCom, there are two types of reasons for elimination as follows:

A. Elimination due to simple rule enforcement by the administrator. Examples would be someone that did not meet the eligibility requirements or whose inclusion would violate the rule limiting the number of voters with the same sponsor or all but one occurrence of someone included multiple times due to a name change or similar confusion. When there are such eliminations in the initial selectees, the administration simply goes further down the ordered list produced with the initial randomness sources until there are the desired number of selectees who are not eliminated by such decisions. The administrator SHOULD announce who has been eliminated and the reason for the administrator's decision to eliminate them.

B. Eliminations due to a selectee, that is, agreement from the selectee to serve cannot be obtained by the administrator before a deadline established by the administrator. For example, either the selectee declines to serve or, despite all reasonable efforts, the selectee is not adequately contactable.

   (The elimination of someone due to non-contactability may work a hardship for that individual if it was due to no fault of their own and they wanted to serve. But there is no reasonable alternative if a NomCom voting membership of volunteers with a confirmed agreement to serve is to be finalized in a timely manner. Since someone so eliminated will, as provided below, be replaced by another randomly selected pool member, there is no problem from the point of view of NomCom composition.)

It will frequently be the case that, after the initial selection from the pool and the handling of any Type A eliminations, there will be a small number of Type B eliminations. If no further actions were taken, there will be an insufficient number of people selected and not eliminated. If selection were extended in this case by just going further down the ordered list, as with Type A eliminations, this would give initially selected persons the ability to, in effect, transfer their voting NomCom membership to a known different person since the entire initial ordered list is, at that point, publicly known. Some perceive this as a problem, so it is resolved by the administrator iteratively using what is essentially a miniature version of the initial selection as follows:

1. The new pool consists of the initial pool without any selectees who have agreed to serve and without any pool members eliminated by any earlier Type A or B eliminations.

2. The new randomness is created using a specific instance of a public daily source announced at the same time as the initial sources. Since an extended selection is normally of a much lower number of selectees (typically 1 or 2) from a smaller pool, much less entropy is needed. For example, a 4 or 5 digit daily number announced by a government lottery would be adequate. This random source is treated as an additional source added to the initially announced list of random sources and processed as specified resulting in it being suffixed to the key produced by the initial randomness sources. (See worked example and reference code below.)

3. The administrator announces how many additional selections are needed and the specific future daily random source that will be used. At least a few hours should be allowed between this announcement and the public availability of the extension random source. As soon as the random source is available, the administrator announces the extended selections and any further extension of the extended selections due to Type A eliminations as above.

4. The administrator still needs to check for Type B eliminations among the new selectees. Unfortunately, at this point, the time constraints are likely to be quite tight so contacting an extensions selectees to be sure they are still willing to serve must be done urgently and with a tight deadline. Since there may be further Type B eliminations among the extended selectees, more than one cycle of extension may be needed. If so, these steps 1 through 4 are repeated with minor modifications as follows: For Step 1, those in the pool before the next extension are all those from the initial pool who have not, in the initial selection or any previous extensions, been selected or been subject to Type A or Type B elimination. In particular, someone who was uncontactable in an earlier round does NOT become eligible for later rounds even if they have become contactable. For Step 2, a different future version of the daily randomness source is used as the additional randomness; when multiple selection extensions have to be run, the additional randomness does not pile up making the random key longer and longer but rather each extension's additional randomness is used with the initial random sources. Step 3 and 4 are unaltered.

Unfortunately, multiple extension cycles may be required so the selection administration should allow enough time for 5 or so of them. For example, in the selection of the 2022/2023 NomCom, 3 extensions would have been required: The pool had 267 members, probably the largest ever. In the initial selection, one of the 10 selectees was Type B eliminated because confirmation of their willingness to serve could not be obtained in a timely fashion. In the 1st extended selection, the 11th selectee declined to serve and the 12th was Type A eliminated because there were already two selectees with the same sponsor. In the 2nd extended selection, the 13th person selected also decline to serve. In the 3rd extended selection, the 14th person selected became the final voting member of the Nomcom when they confirmed their willingness to serve.

6. Handling Real World Problems

In the real world, problems can arise in following the steps and flow outlined in Sections 2 through 5 above. Some problems that have actually arisen are described below with recommendations for handling them.

6.1 Uncertainty as to the Pool

Every reasonable effort should be made to see that the published pool, from which selection is made, is of certain and eligible persons. However, especially with compressed schedules or perhaps someone whose claim that they volunteered and are eligible has not been resolved by the deadline, or a determination that someone is not eligible which occurs after the publication of the pool, or the like, there may still be uncertainties.

The best way to handle this is to maintain the announced schedule, INCLUDE in the published pool all those whose eligibility is uncertain and to keep the published pool list numbering IMMUTABLE after its publication. If one or more people in the pool are later selected by the algorithm and random input but it has been determined they are ineligible, they can be skipped and subsequently selected persons used. Thus, the uncertainty only effects one selection and in general no more than a maximum of U selections where there are U uncertain pool members.

Other courses of action are far worse. Actual insertion or deletion of entries in the pool after its publication changes the length of the list and totally scrambles who is selected, possibly changing every selection. Insertion into the pool raises questions of where to insert: at the beginning, end, alphabetic order, ... Any such choices by the selection administrator after the random numbers are known destroys the public verifiability of unbiased choice. Even if done before the random numbers are known, such dinking with the list after its publication just smells bad. There should be clear fixed public deadlines and someone who challenges their absence from the pool after the published deadline should have their challenge automatically denied for tardiness.

6.2 Randomness Ambiguities

The best good faith efforts have been made to specify precise and unambiguous sources of randomness. These sources have been made public in advance and there has not been objection to them. However, it has happened that when the time comes to actually get and use this randomness, the real world has thrown a curve ball and it isn't quite clear what data to use. Problems have particularly arisen in connection with individual stock prices, volumes, and financial exchange rates or indices. If volumes that were published in thousands are published in hundreds, you have a rounding problem. Prices that were quoted in fractions or decimals can change to the other. If you take care of every contingency that has come up in the past, you can be hit with a new one. When this sort of thing happens, it is generally too late to announce new sources, an action which could raise suspicions of its own. About the only course of action is to make a reasonable choice within the ambiguity and depend on confidence in the good faith of the selection administrator. With care, such cases should be extremely rare.

Based on these experiences, it is again recommended that public lottery numbers or the like be used as the random inputs and financial volumes or prices avoided.

7. Fully Worked Example

<<Example needs to also cover the Section 5 Extension provisions.>>

Assume the eligible volunteers published in advance of selection are the numbered list of 30 past NomCom Chairs appearing below in Appendix A.

Assume the following (fake example) ordered list of randomness sources:

1. The Kingdom of Alphaland State Lottery daily number for 1 November 2022 treated as a single four-digit integer.
2. The People's Democratic Republic of Betastani State Lottery six winning numbers for 1 November 2022 and then the seventh "extra number" for that day as if it was a separate random source.

Hypothetical randomness publicly produced:

```
 Source 1:  9319
 Source 2a:  9, 61, 26, 34, 42, 41
 Source 2b:  55
```

Resulting key string:

```
 9319./9.26.34.41.42.61./55./
```

The table below gives the hex of the MD5 of the above key string bracketed with a two-byte string that is successively 0x0000, 0x0001, 0x0002, through 0x0010 (16 decimal). The divisor for the number size of the remaining pool at each stage is given and the index of the selectee as per the original number of those in the pool.

```
index        hex value of MD5        div  selected
 1  5A0EE2F8849A8C8DFC93BE36FE2D674A  30  -> 15 <-
 2  E390DA3449C586B6BBD9F56B23B86E25  29  -> 11 <-
 3  D053FC140209EADB8340C185B8EC58FD  28  -> 10 <-
 4  0C9DC84909A82D2203959EE54A8B1867  27  ->  6 <-
 5  BD92A498AEF2E60E7867E5B7B434892F  26  -> 30 <-
 6  28E9021C3788F54BF0FD6835BCD1E3C2  25  -> 27 <-
 7  FF6C6197802654B3B1B341DD754A4BE0  24  ->  1 <-
 8  991135A2767FB80D4CEBB736CD7E3BAE  23  ->  9 <-
 9  4E18F325603FF603FC24F43459C2CFAC  22  -> 25 <-
10  4A0AA0F72441B6345E69FCDD4C378558  21  -> 18 <-
11  4E9EBC623E2930D4DD61B0FDEC3B2875  20  -> 16 <-
12  8780D26F8C724EB09CDD155C3B66AF17  19  -> 24 <-
13  FFF90A6A23BE02D07BA2FA18E6275791  18  ->  5 <-
14  39FBCDC0CC4F0147CDEABC31D28D36A9  17  -> 28 <-
15  6F6C2DC3A682E11CF3BC90C682C9104C  16  -> 22 <-
```

Resulting first ten selected, in order selected:

```
 1.  L. Dondeti (15)      6.  V. Kuarsingh (27)
 2.  R. Draves (11)       7.  J. Case (1)
 3.  P. Roberts (10)      8.  T. Ts'o (9)
 4.  D. Eastlake (6)      9.  P. Yee (25)
 5.  R. Salz (30)        10.  T. Walsh (18)
```

Should one of the above turn out to be ineligible or uncontactable or decline to serve, the next would be J. Halpern, number 16.

8. Security Considerations

Careful choice should be made of randomness inputs so that there is no reasonable suspicion that they are under the control of the administrator. Guidelines given above to use a small number of inputs with a substantial amount of entropy from the last should be followed. And equal care needs to be given that the algorithm selected is faithfully executed with the designated inputs values.

Publication of the results and a one-week window for the community of interest to duplicate the calculations should give a reasonable assurance against implementation tampering.

9. IANA Considerations

This document requires no IANA actions.

10. Reference Code

This code makes use of the MD5 reference code from [RFC1321] ("The MD5 Message-Digest Algorithm"). The portion of the code dealing with multiple floating point numbers was written by Matt Crawford. The original code in RFC 2777 could only handle pools of up to 255 members and was extended to 2**16-1 by Erik Nordmark. This code has been extracted from this document, compiled, and tested. While no flaws have been found, it is possible that when used with some compiler on some system under some circumstances some flaw will manifest itself.

```
<CODE BEGINS>

<< CODE HAS NOT YET BEEN UPDATED TO COVER EXTENDED SELECTION. >>

/****************************************************************
 *
 *  Reference code for
 *      "Publicly Verifiable Random Selection"
 *          Donald E. Eastlake 3rd
 *              Original February 2004, Updated September 2022
 *
 * Redistribution and use in source and binary forms, with or
 * without modification, is permitted pursuant to, and subject
 * to the license terms contained in, the Revised BSD License
 * set forth in Section 4.c of the IETF Trust's Legal Provisions
 * Relating to IETF Documents
 * (http://trustee.ietf.org/license-info).
 ****************************************************************/

#include <limits.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* From RFC 1321 */
#include "global.h"
#include "MD5.h"

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


/* limited to up to 16 inputs of up to sixteen integers each */
```

```
/* pool limit of 2**8-1 extended to 2**16-1 by Erik Nordmark */
```

```
/*****************************************************************/

int main ()
{
int          i, j,  k, k2, err, keysize, usel;
unsigned short   remaining, *selected;
long int     pool, selection, temp, array[16];
MD5_CTX      ctx;
char         buffer[257], key [800], sarray[16][256];
unsigned char    uc16[16], unch1, unch2;

/* get basic parameters */
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 ) + 0.05 );
    if ( err <= 0 )
        exit ( 1 );
    }

/* get the "random" inputs. echo back to user so the user may
   be able to tell if truncation or other glitches occur.  */
for ( i = 0, keysize = 0; i < 16; ++i )
     {
     if ( keysize > 500 )
         {
         printf ( "Too much input.\n" );
         exit ( 1 );
```

```
        }
    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' )  /* "e"nd */
                break;     /* break out of "for i" */
            else
            {   /* floating point code by 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++ )
                      {
                            /* strip trailing zeros */
                      for ( k2=strlen(sarray[k]); sarray[k][--k2]=='0';)
                            sarray[k][k2] = ' ';
                      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
            {   /* sort values, not a very efficient algorithm */
```

```
            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 )
                {      /* print for user check */
                err = printf ( "%ld ", array[k] );
                if ( err <= 0 )
                    exit ( 1 );
                keysize += sprintf ( &key[keysize], "%ld.", array[k] );
                }
            keysize += sprintf ( &key[keysize], "/" );
            }
        }    /* end "for i" */
    if ( i == 0 )
        {
        printf ( "No key input.\n" );
        exit (1);
        }

    /* have obtained all the input, now produce the output */

    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);
        /* prefix/suffix extended to 2 bytes by 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;
}

/* prompt for a positive non-zero integer input */
/****************************************************************/
long int getinteger ( char *string )
{
long int     i;
int          j;
char    tin[257];

while ( 1 )
{
printf ( "%s", 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;
}   /* end while */
}

/* get remainder of dividing a 16 byte unsigned int
   by a small positive number */
/****************************************************************/
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;
   }   /* end longremainder */

   /* calculate how many bits of entropy it takes to select N from P */
   /****************************************************************/
   /*             P!
     log  ( ----------------- )
        2    N! * ( P - N )!
   */
   double NPentropy ( int N, int P )
   {
   int         i;
   double      result = 0.0;

   if (    ( N < 1 )   /* not selecting anything? */
      ||   ( N >= P )  /* selecting all of pool or more? */
      )
       return 0.0;     /* degenerate case */
   for ( i = P; i > ( P - N ); --i )
       result += log ( i );
   for ( i = N; i > 1; --i )
       result -= log ( i );
   /* divide by [ log (base e) of 2 ] to convert to bits */
   result /= 0.69315;

   return result;
   }   /* end NPentropy */

   <CODE ENDS>
```

Normative References

[RFC20] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Informative References

[RFC3797] Eastlake 3rd, D., "Publicly Verifiable Nominations Committee (NomCom) Random Selection", RFC 3797, DOI 10.17487/RFC3797, June 2004, <https://www.rfc-editor.org/info/rfc3797>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC8713] Kucherawy, M., Ed., Hinden, R., Ed., and J. Livingood, Ed., "IAB, IESG, IETF Trust, and IETF LLC Selection, Confirmation, and Recall Process: Operation of the IETF Nominating and Recall Committees", BCP 10, RFC 8713, DOI 10.17487/RFC8713, February 2020, <https://www.rfc-editor.org/info/rfc8713>.

Appendix A: History of NomCom Member Selection

For reference purposes, here is a list of the IETF Nominations Committee member selection techniques and chairs so far:

Num	YEAR	CHAIR	SELECTION METHOD
1	1993/1994	Jeff Case	Clergy
2	1994/1995	Fred Baker	Clergy
3	1995/1996	Guy Almes	Clergy
4	1996/1997	Geoff Huston	Spouse
5	1997/1998	Mike St.Johns	Algorithm
6	1998/1999	Donald Eastlake 3rd	RFC 2777
7	1999/2000	Avri Doria	RFC 2777
8	2000/2001	Bernard Aboba	RFC 2777
9	2001/2002	Theodore Ts'o	RFC 2777
10	2002/2003	Phil Roberts	RFC 2777
11	2003/2004	Rich Draves	RFC 2777
12	2004/2005	Danny McPherson	RFC 3797
13	2005/2006	Ralph Droms	RFC 3797
14	2006/2007	Andrew Lange	RFC 3797
15	2007/2008	Lakshminath Dondeti	RFC 3797
16	2008/2009	Joel M. Halpern	RFC 3797
17	2009/2010	Mary Barnes	RFC 3797
18	2010/2011	Tom Walsh	RFC 3797
19	2011/2012	Suresh Krishnan	RFC 3797
20	2012/2013	Matt Lepinski	RFC 3797
21	2013/2014	Allison Mankin	RFC 3797
22	2014/2015	Michael Richardson	RFC 3797
23	2015/2016	Harald Alvestrand	RFC 3797
24	2016/2017	Lucy Lynch	RFC 3797
25	2017/2018	Peter Yee	RFC 3797
26	2018/2019	Scott Mansfield	RFC 3797
27	2019/2020	Victor Kuarsingh	RFC 3797
28	2020/2021	Barbara Stark	RFC 3797
29	2021/2022	Gabriel Montenegro	RFC 3797
30	2022/2023	Rich Salz	RFC 2797

Clergy = Names were written on pieces of paper, placed in a receptacle, and a member of the clergy picked the NomCom members.

Spouse = Same as Clergy except chair's spouse made the selection.

Algorithm = Algorithmic selection based on similar concepts to those documented in RFC 2777 and herein.

RFC 2777 = Algorithmic selection using the algorithm and reference code provided in RFC 2777 (but not the fake example sources of randomness).

RFC 3797 = Algorithmic selection using the algorithm and reference code provided in RFC 3797 (but not the fake example sources of randomness).

Appendix B: Changes from RFC 3797

This document differs from [RFC3797], the previous version, in the following primary ways:

1. Many editorial changes. Add IANA Considerations section.

2. Use [RFC20] as the reference for ASCII.

3. Update Appendix A.

4. Add Section 5: Extended Selection.

Acknowledgements

Acknowledgements for this document: TBD

Acknowledgements for RFC 3797: Matt Crawford and Erik Nordmark made major contributions to this document. Comments by Bernard Aboba, Theodore Ts'o, Jim Galvin, Steve Bellovin, and others have been incorporated.

Authors' Address

Donald E. Eastlake 3rd
Futurewei Technologies
2386 Panoramic Circle
Apopka, FL 32703
USA

Phone: +1-508-333-2270
EMail: d3e3e3@gmail.com