Workgroup: Network Working Group
Internet-Draft:
draft-hoffman-random-candidate-selection-02
Published: 20 November 2023
Intended Status: Informational
Expires: 23 May 2024
Authors: P. Hoffman
ICANN

Simple Random Candidate Selection

Abstract

This document describes a process to randomly select a subset of named candidates from a larger set of candidates. The process uses an unpredictable value that can be trusted by all candidates.

This draft has a GitHub repository. Issues and pull requests can be made there.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 23 May 2024.

Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

It is common to need to pick a subset of people from a larger group using a random selection method. This is often done on an ad hoc basis, but for some selections, a more formal process is needed, particularly if the people in the larger group don't all trust the administrator of the selection process to be unbiased.

This document gives a simple, understandable process that can be done for groups and subsets of arbitrary size. The process is purposely transparent and reproducible. It works with any group of entities that have names: people, companies, locations, and so on.

As a simple example, a future leadership committee will have a fixed size. The members of the committee will be selected from a large pool of volunteers. Someone is in charge of collecting the names of the volunteers and making a randomized selection among them for the leadership committee. They can use the process in this document to make that selection in a way that is both provably random and understandable.

As described later in this document, the process can also be used for weighted selections (Section 5) and for randomly sorting lists of candidates (Section 6).

Due to the formatting used in this document, the reader is encouraged to read the HTML version, although the text version is still usable.

See [I-D.thomson-elegy-vrs] for a similar method as described here.

2. Overview of the Process

A few terms are used throughout this document:

ceremony: The act of collecting names into a pool, making a random selection from the pool, and publishing the entire process in a clear and transparent method.

ceremony administrator (CA): The person who performs the steps of the ceremony.

candidate: A person, organization, or other namable entity that is possibly being selected during the ceremony.

candidate name: The name used by each candidate in the pool. The candidate name is expressed as a string of Unicode characters in UTF-8 format [Unicode] [UTF-8].

difficult-to-predict string (*D*): A publicly-visible string that is only known after the pool of candidates has been closed. (Note that this is different from what is normally called a "random number" or a "random string". True random numbers or strings are designed to be nearly impossible to predict, whereas *D* in this process has weak but sufficient randomness.)

selection size (*S*): The number of candidates that will be selected from the pool.

2.1. Basic Steps

The steps in a ceremony that follows this process is given here. See Section 3 for more detail on the steps.

1. The CA starts the ceremony by performing the following steps at the same time:

   *Announces an end date for when the pool will be complete.

   *Announces a later date on which *D*, the difficult-to-predict string, will be selected.

*Announces the source where *D* will be found on that later date.

*Announces *S*, the number of candidates that will be selected.

*Opens up the pool of candidates for submission.

2. Candidates submit their names to the pool until the closing date, and the CA puts the allowed names in the pool.

3. On the closing date, the CA publishes the candidate names from the pool with the hexadecimal value of the UTF-8 encoding for each candidate name.

4. On the date for selecting *D*, the CA gets *D* from the announced source.

5. The CA calculates the hashes used to make the selection. The CA concatenates each candidate name with *D* (name first, then *D*), uses the SHA-256 hash function [SHA-2] on the resulting string, and records the value of the hash as a UTF-8 string.

6. The CA arranges the set of hash values in alphabetic order from highest to lowest. They then select the *S* candidates from the top of the list (that is, the names whose hash values are largest).

3. Specifics for the Process

3.1. Start of Ceremony

Much of the trust in the selection process is based on the CA not being able to influence the selection. If the CA can choose, or even influence, the value of *D*, they can help establish the outcome of the selection. Similarly, if one or more of the candidates can influence the value of *D*, they can increase their chance of being selected.

To make the process trustworthy, the value of *D* must be unrelated to the CA or the candidates, and it must be selected only after the list of candidates is completed. The most important things for a ceremony is that the source is announced before the ceremony starts, that all participants and viewers of a ceremony can find the source on the date specified by the CA, that all candidates believe that no candidate can influence *D* on that date, and that everyone gets the same value when they go to the source for that date.

3.1.1. Use of the FTSE 100 Index

The process described in this document uses the closing value for the FTSE 100 Index on the particular day selected by the CA. The FTSE 100 Index is a long-established index based on 100 stocks; it is sometimes known by its stock ticker as "UKX". A common open source of those values is the Wall Street Journal. The daily closing for the FTSE 100 Index at the Wall Street Journal can currently be found here.

Note that the location for sources of daily closing values can change over time. The CA must check that the intended source is still active, and still available when the ceremony starts.

Values from the FTSE 100 Index in this procedure are always encoded as four digits, followed by a period character (U+002E), followed by two more digits, such as:

```
7623.10
```

If the FTSE 100 Index ever goes above 10,000, the encoded values would be five digits, followed by a period character (U+002E), followed by two more digits.

3.1.2. Other Public Sources of Randomness

Although the procedure in this document uses the FTSE 100 Index as a public source of randomness, there are many other sources that can be used by a CA, as long as the source chosen is trusted by the candidates. There are many other stock indexes with enough stocks in them to make prediction of the exact value have less than a 0.1% chance. Having said that, using a future price of a single stock is probably not a good public source of randomness because candidates are likely to trust the variability of that less than the variability of a basket of stocks.

Some systems that use public sources of randomness use the results of an unrelated lottery, such as the type of lotteries that many countries hold. These are probably trusted by candidates not be able to be manipulated. However, lotteries normally are a set of numbers between 1 and 100, often five or more such numbers. If the CA uses such a lottery for this procedure, they need to specify how the numbers from the lottery of the chosen date will be combined, including whether or not the numbers from 1 to 9 need to be preceded by a "0" character.

There are other public sources of randomness, such as cameras pointed at lava lamps and so on. These are probably not good choices for the type of ceremony described in this document because the operators of such sources are not publicly trusted entities.

Note that some sources of randomness may have less randomness than it appears at first glance. There can be hidden biases towards certain values that are not obvious when looking at a small set of recent values. If a CA chooses a source for *D* other than the FTSE 100 Index, the data from source should be measured over a long period of time for unexpected biases toward values that a candidate can use to improve their chance of being selected.

3.2. Name Submission and Pool Creation

The CA is the sole arbitrator for whether a candidate is allowed to enter the pool. The CA is also the sole arbitrator of what name string (in UTF-8) the candidate can use in the pool.

The order that the candidates join the pool does not affect the outcome of the selection process. Said another way, the pool is kept as an unordered set of candidates, not an ordered list of candidates.

It is a good practice for the CA to have consistent rules for the names, such as only using ASCII space characters (U+0020), only one space between each name part, no trailing spaces, and so on. These rules can be more difficult when the candidates are company names (such as whether the legal standing of the company such as "Inc." is included), but making consistent rules is not that difficult.

3.3. Closing Submissions to the Pool

At the closing of submissions, the CA verifies that the number of candidates in the pool is larger than *S*. If the length is the same as *S*, the rest of the steps are unneeded (and could be confusing), because all candidates will automatically be selected. If the length is shorter than *S*, the ceremony stops because there are too few candidates.

The method for publishing the set of candidates is determined by the CA. Figure 3 gives an example of how a CA might publish this information.

3.4. Selecting *D*

On the day that the CA announced for the selection of *D*, the CA goes the the source they announced and gets *D*. After the CA retrieves *D* from the announced source, they encode *D* as a UTF-8 string. In the example of the FTSE 100 Index, a closing value for the day announced at the beginning of the ceremony might be "7623.10". This would be encoded in UTF-8 as the string of characters whose hex value is 0x373632332e3130.

3.5. Calculating Hashes

Different programming libraries have different requirements for the input to hash functions. Appendix A uses the built-in hashlib library in Python, which requires that text strings have a specified encoding.

3.6. Selecting *S* Candidates

The process of selecting is simply taking the *S* candidates whose hash value is highest. This can easily be determined by sorting the text representation of the hash values in descending order because in UTF-8 and ASCII, digits have lower codepoints than letters.

To complete the process in a transparent manner, the CA should publish all known data for the ceremony. This includes *S*, *D*, the hexadecimal value of *D*, all of the information for each candidate, and the full list of selected candidates. Figure 5 shows an example of what this publication might look like.

4. Handling Ceremony Process Issues

Ceremonies don't always go as planned. For example, after a ceremony completes, one or more of the selected candidates might be removed from the selected set due to voluntary withdrawal or established rules (such as no two candidates being from the same geographic region). In such cases, no new ceremony is needed: the CA simply selects the next candidate(s) on the list that is ordered by hash values.

Similarly, if after the selection process is completed, the size *S* of the selected set needs to increase, the CA simply selects the next candidate(s) on the list that is ordered by hash values.

5. Performing a Weighted Selection of Candidates

In some candidate selections, the CA wants to give candidates a weighted chance of being selected. For example, a legislature might select its leadership randomly, but weights the chance of being selected by the size of the membership of the political party in the legislature. The CA can create the pool with multiple names for each party, giving each name a number.

For example, assume a legislature has 27 members of the Orange party, 20 members of the Yellow party, and 7 members of the Green party. The CA could create a pool consisting of the names "Orange1", "Orange2", ... "Orange27", "Yellow1", "Yellow2", ... "Yellow20", "Green1", "Green2", ... "Green7". The selected party would be the one whose name appears in the first name of the list of hashes.

6. Performing a Random Ordering of Candidates

Some use cases do not involve a selection of candidates from a larger list, but instead sorting the list of candidates randomly. The process given in this document can be easily used to do this: set *S* to the size of the pool, peform the steps of the ceremony, and create the output list in the last step as all *S* candidates in alphabetic order from highest to lowest of the hash values.

7. IANA Considerations

This document has no IANA considerations.

8. Security Considerations

The value *D* used in this process is explicitly not cryptographically strong; in fact, it might provide only a few bits of randomness. The FTSE 100 Index might be predictable after the third digit from the right, but not the last three digits, meaning that they only have randomness of about 10 bits. The value of *D* is concatenated into each candidate string before the whole string is hashed, so incorrectly predicting even one character of *D* completely changes the value of the hash for comparison.

A cryptographic hash function like SHA-256 has the property that changing any individual bit of the input will change every bit in the output with a 50% chance, regardless of the position of the bit in the input. Appending a small amount of randomness at the end of the input is just as effective as prepending the randomness at the beginning of the input nd just as effective as mixing the randomness throughout the input. The procedure in this document appends the string from the FTSE 100 Index at the end of the candidate name because it makes viewing the pre-hashed result easier while still causing the maximum change to the resulting hash value.

A candidate who has a lot of leeway in choosing their name can possibly increase their chance of being selected by as much as 0.1% with such source of randomness. The procedure in this document assumes that candidates have very little leeway in choosing their names; the CA must accept each name before it is put into the pool. The combination of the limited leeway for choosing the names in the pool and the necessity to predict *D* exactly in order to gain any benefit means that *D* needs much less randomness that a random number that would be used during encryption or authentication.

9. References

9.1. Normative References

[SHA-2]

Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/rfc/rfc6234>.

[Unicode] The Unicode Consortium, "The Unicode Standard (latest version)", n.d., <https://www.unicode.org/versions/latest/>.

[UTF-8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/rfc/rfc3629>.

9.2. Informative References

[I-D.thomson-elegy-vrs] Thomson, M., "A Verifiable Random Selection Process", Work in Progress, Internet-Draft, draft-thomson-elegy-vrs-00, 22 June 2023, <https://datatracker.ietf.org/doc/html/draft-thomson-elegy-vrs-00>.

Appendix A. Sample Code

The following is a list of figures for an implementation of the procedure shown in this document.

* The Python script in Figure 1 implements the algorithm from this document.

* The file that contains the list of names is shown in Figure 2. (The names are the winners of the Nobel laureates in Literature for 2016 through 2021.)

* A file showing the UTF-8 representation of the names from Figure 2 is shown in Figure 3. This file is suitable for showing to the candidates.

* The file that contains the *S* and *D* on separate lines is shown in Figure 4.

* Figure 5 shows the result of running the program with that file as input.

```
#!/usr/bin/env python3

# Program to randomly select some candidates from a group
#  See draft-hoffman-random-candidate-selection

import hashlib, sys
from pathlib import Path

# Helper function to turn a UTF-8 string into its hex representation
def hexify(in_str):
  return "".join([hex(c)[2:] for c in in_str.encode("utf8")])

# Sanity check the input files given on the command line
if len(sys.argv) == 1:
  exit("Must give the name of the candidate file, and possibly " + \
    "the selection file, on the command line. Exiting.")
candidate_path = Path(sys.argv[1])
if not candidate_path.exists():
  exit(f"The file {str(candidate_path)} doesn't exist. Exiting.")
try:
  candidate_f = candidate_path.open(mode="rt", encoding="utf8")
except:
  exit("The candidates file doesn't appear to be in UTF-8. Exiting.")
candidate_lines = candidate_f.read().splitlines()
# See if there is a second file for selecting
if len(sys.argv) == 3:
  run_including_selection = True
  selection_path = Path(sys.argv[2])
  if not selection_path.exists():
    exit(f"The file {str(selection_path)} doesn't exist. Exiting.")
  try:
    selection_f = selection_path.open(mode="rt", encoding="utf8")
  except:
    exit("The selection file doesn't appear to be UTF-8. Exiting.")
  selection_lines = selection_f.read().splitlines()
  # Extract D and S from the selection file
  S_str = selection_lines[0]
  try:
    S = int(S_str)
  except:
    print(f"The first line of the selection file, '{S_str}', " + \
      "is not an integer. Exiting.")
  # D_str is the string for D, D_hex is the hex version for display
  D_str = selection_lines[1]
  D_hex = hexify(D_str)
else:
  run_including_selection = False

# Get the candidates information
```

```
C_info = []
for C_str in candidate_lines:
  C_hex = hexify(C_str)
  if run_including_selection:
    C_with_D_str = C_str + D_str
    C_with_D_hex = hexify(C_with_D_str)
    C_with_D_hash = hashlib.sha256(C_with_D_hex.encode("utf-8"))
    C_info.append([C_str, C_hex, C_with_D_str, C_with_D_hex, \
      C_with_D_hash.hexdigest()])
  else:
    C_info.append([C_str, C_hex])

# Print the results
if run_including_selection:
  print(f"S is {S}")
  print(f"D is \"{D_str}\"")
  print(f" {D_hex}\n")
  print("Candidate information, sorted by hash of name including D")
  selected = []
  # Sort by the hex of C_with_D_hash
  for this_info in sorted(C_info, key=lambda a: a[4], reverse=True):
    # Decrement S for each name that is selected
    if S > 0:
      selected.append(this_info[0])
      S -= 1
    print(f"{this_info[2]}")
    print(f" {this_info[3]}")
    print(f" {this_info[4]}")
  print("\nSelected:\n    " + "\n    ".join(selected))
else:
  for this_info in C_info:
    print(f"{this_info[0]}")
    print(f" {this_info[1]}")
```

Figure 1: Example Python code for this procedure

```
Bob Dylan

Olga Tokarczuk
Peter Handke
Louise Glück
Abdulrazak Gurnah
```

Figure 2: Sample name list file

```
Bob Dylan
 426f622044796c616e

 e79fb3e9bb9220e4b880e99b84
Olga Tokarczuk
 4f6c676120546f6b6172637a756b
Peter Handke
 506574657220486616e646b65
Louise Glück
 4c6f7569736520476cc3bc636b
Abdulrazak Gurnah
 416264756c72617a616b204775726e6168
```

Figure 3: Full information for the names

```
3
7623.10
```

Figure 4: Sample selection information file

```
S is 3
D is "7623.10"
 373632332e3130

Candidate information, sorted by hash of name including D
   7623.10
 e79fb3e9bb9220e4b880e99b84373632332e3130
 f2e0d3bbd8eac635d799702bead0fbdf07ff79ef94a261789de50e81adb38a13
Louise Glück7623.10
 4c6f7569736520476cc3bc636b373632332e3130
 a54e282cbaa1f29543cd13d9a29e07e3a38413360172b722f8259c2baa3c38dd
Peter Handke7623.10
 506574657220486 16e646b65373632332e3130
 8bb3bc197c6462b033e4d8e8cf703b13b1c55172572a85d56c639db5c57d3866
Olga Tokarczuk7623.10
 4f6c676120546f6b6172637a756b373632332e3130
 56166c4e0e6ca027f4150bac5ce83fbf5652e440214fd255308472fed9f8fb1b
Abdulrazak Gurnah7623.10
 416264756c72617a616b204775726e6168373632332e3130
 340413dc6b2574f5ddc5e88e1c986a229d9defccbae249789b07a5d2337981ff
Bob Dylan7623.10
 426f622044796c616e373632332e3130
 05eb403f4f59f5a7b21f5e5a4e8dbfbac59344fd5e8708ab618b5e2ed27a52de

Selected:

    Louise Glück
    Peter Handke
```

Figure 5: Output of running the program on the list of names and selection information

Author's Address

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org