Publicly Verifiable NomCom Random Selection

Donald E. Eastlake 3rd

d3e3e3@gmail.com

<draft-eastlake-rfc3797bis-04>
History

• The voting members of the IETF NomCom are selected randomly for the qualified volunteers. (Current volunteer qualifications are in RFC 9389.)

• From 1993 to 1997 names were put in a hat and one name drawn by hand.

• Since 1997 an algorithm has been used where the list, the algorithm, and future sources of randomness, such as lottery winning numbers, are announced in advance.
  • Then the algorithm is used to select from the list based on the randomness.
  • From 1998 to 2004 algorithm based on Informational RFC 2777.
  • From 2004 to 2023 algorithm based on Informational RFC 3797.
Why Change?

• When the previously announced sources produce their randomness, a total ordering of the volunteers is established. The first ten would be the NomCom voters except for eliminations:
  1. No more than two voters with the same sponsor.
  2. Volunteer must be contactable and agree to serve.

• The selection administrator (the Chair of the NomCom) can do type 1 eliminations by going further down the list. In the past, they have done this for type 2 eliminations also, but that may enable someone who would have been selected to, by refusing to serve, “transfer” their vote to a known person a little further down the list.
Why Change? (continued)

• Here is a simplified example:

• Assume a pool of 22 people numbered (1) through (22) in alphabetic order. Using a method like RFC 3797, these are randomly ordered.

• The first ten selected are underlined.

• Under the previous system, any of the first ten selectee could effectively transfer their voting membership to Charles by refusing to serve.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Jari (10)</td>
</tr>
<tr>
<td>2.</td>
<td>Ethan (5)</td>
</tr>
<tr>
<td>3.</td>
<td>Allen (1)</td>
</tr>
<tr>
<td>4.</td>
<td>Udeep (21)</td>
</tr>
<tr>
<td>5.</td>
<td>Victor (22)</td>
</tr>
<tr>
<td>6.</td>
<td>Oscar (15)</td>
</tr>
<tr>
<td>8.</td>
<td>Frederick (6)</td>
</tr>
<tr>
<td>9.</td>
<td>Quin (17)</td>
</tr>
<tr>
<td>11.</td>
<td>Charles (3)</td>
</tr>
</tbody>
</table>
A Solution

• If extended selection is required due to type 2 eliminations, use some new randomness to re-randomize the remaining pool members.
  • This is not an abstract concern. Three extensions were needed to finish selection of the 2022-2023 NomCom due to type 2 eliminations.

• But the process is time constrained.
  • Elaborate announcement of future publicly trusted randomness sources might take too long.
  • On-line rapid sources of “randomness” do not inspire confidence.

• Solution Refinement: Use a hash chain starting with a secret seed so “new randomness” is immediately available.
Solution Hash Chain Details

• When selecting the public sources of future randomness, the administrator also selected a secret random number R.

• Using a hash function H, the administrator secretly computes $H[1](R) = H(R)$, $H[2](R) = H(H(R))$, $H[3](R) = H(H[2](R)) = H(H(H(R)))$, $H[4](R) = H(H[3](R)) = H(H(H(H(R))))$, ..., $H[N](R) = H(H[H[N-1](R)])$.

• The administrator announces H and $H[N](R)$ at the same time the future randomness sources are announced and uses $H[N](R)$ as part of the input to the selection algorithm.

• When an extension to the selection process is needed due to a type 2 elimination, the administrator replaces $H[N](R)$ in the randomness used for selection with $H[N-k](R)$ where this is the k’th extension.

• By announcing H and $H[N](R)$ initially, the administrator is bound to subsequent earlier hash chain values but these cannot be derived from $H[N](R)$ due to the non-invertibility assumed for H. The administrator can influence $H[N](R)$ but gains nothing thereby due to the future public randomness.
Other Changes

draft-eastlake-rfc3797bis changes from RFC 3797

• Uses HMAC-SHA-256 instead of MD-5.
• Example code has provisions for
  • Generating a hash chain using SHA-256.
  • Doing both
  • a simple pick from a pool without extensions or
  • doing selection with provisions for extension.
  • Worked example will be extended to cover these.
An Alternative “Hash” Method Has Been Suggested

• And alternative “hash” method has been suggested recently and a similar “hash” method was suggested years ago.

• I discuss the current “probe” method and the “hash” alternative, in the following slides.

• While I’m happy with the probe method, the hash method works, and I have no problem with adding it as an option.
An Alternative “Hash” Method Has Been Suggested

• Current method successively selects an entry in the pool, marking it as selected, then picking from the remaining pool, etc. I call this the “Probe” method because it repeatedly probes into the remaining pool. This is very sensitive to adding/deleting people after the list is frozen.
  • Extended selection is the same with different randomness.

• The “Hash” method appends randomness to each name in the list, hashes the entries, and then sorts by that hash. This is less sensitive to adding/deleting people, but the exact canonical names must be absolutely fixed.
  • Extended selection involves removing those selected from the pool and then appending different randomness to the remaining members, hashing, and sorting.
Comparison of Probe and Hash Methods

**Probe**
- Very sensitive to adding/deleting volunteers after list is frozen.
- Names in volunteer list can be fixed or adjusted.
- Has been used for IETF NomCom voter selection successfully for 25 years.

**Hash**
- Less sensitive to adding/deleting volunteers after list is frozen.
- Names in volunteer list must be immutable and should be canonicalized.
- Has not previously been used for IETF NomCom voter selection.
Next Step

- Unless there is significant support for the Hash method, I suggest the draft not be made more complex by adding it.

- Dispatch: Rechartered ELERGY WG? GENDISPATCH? A Security WG? AD Sponsored? ...