MUNICIPALITY OF ANCHORAGE

ASSEMBLY MEMORANDUM

No. AM 520-2022

Meeting Date: September 27, 2022

Municipal Clerk's Office
Approved
Date: September 27, 2022

From: **MOA Elections Team**

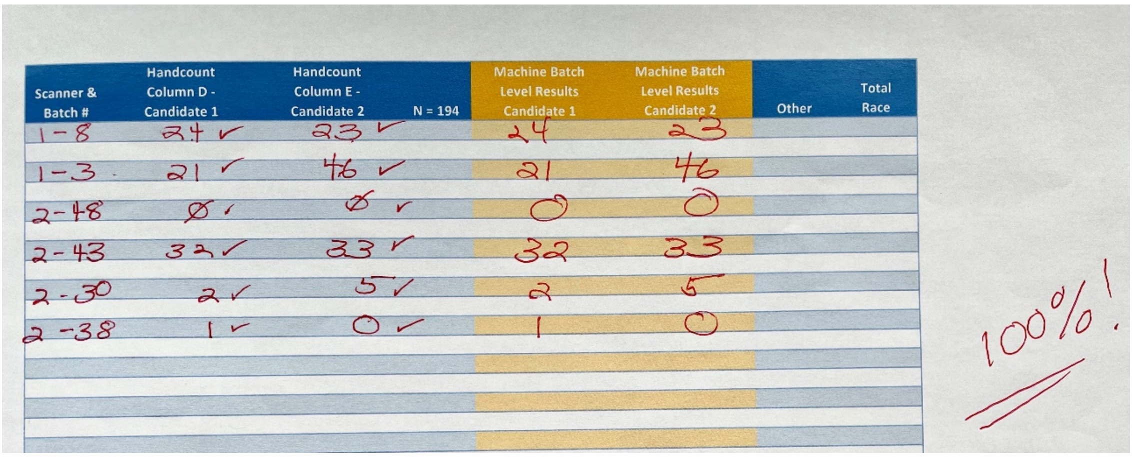
Subject: **Risk Limiting Audit for the June 21, 2022 Special Municipal Election**

I. Executive Summary

The MOA Elections Team conducted a post-election Risk Limiting Audit (RLA) that contained three areas of focus. The audit was publicly noticed and candidates were invited to attend.

1. ***Hand-Count.*** A pre-determined percentage of ballots was selected and the actual ballots were hand counted.
2. ***Machine Review.*** After hand-counting the selected ballots the adjudication of votes was confirmed when necessary. Cast Vote Records were produced from the tabulation system and tallied for the ballots selected.
3. ***Comparison of Hand-Count and Machine Review.*** The totals from the hand-count, detailed in paragraph 1, and the totals from the machine count, detailed in paragraph 2, were compared.

Results of the Risk Limiting Audit = 100% Accurate



Scanner & Batch #	Handcount Column D - Candidate 1	Handcount Column E - Candidate 2	N = 194	Machine Batch Level Results Candidate 1	Machine Batch Level Results Candidate 2	Other	Total Race
1-8	24 ✓	23 ✓		24	23		
1-3	21 ✓	46 ✓		21	46		
2-48	Ø ✓	Ø ✓		0	0		
2-43	32 ✓	33 ✓		32	33		
2-30	2 ✓	5 ✓		2	5		
2-38	1 ✓	0 ✓		1	0		

100%!

The results of the MOA post-election Risk Limiting Audit are that the scanning, adjudication, and tabulation system performed as expected and the results reflect

the will of the voters. All ballots were adjudicated and tabulated as expected. The results of the hand-count and the machine tabulation were identical.[1]

II. Procedure for the Audit

Risk limiting audits use statistically developed audit techniques that allow selection of a number of ballots to be audited that provide statistical confidence that the tabulation system performed as expected. The MOA Election Team conducts a "Batch-Level Comparison Audit," which is a type of RLA that most resembles a "traditional" audit.

A. Selection of Races and Measure to be audited.

1. ***Only One Contest to Audit.*** Given there was only one contest on the ballot; that is the contest the Elections Team audited, with a total of 4,548 ballots cast.

2. ***Target Number of Ballots.*** The MOA procedures for the RLA require a review of 3% of the ballots cast in the election to be audited, rounded down to the nearest hundred. The target number was calculated to be 135 ballots. We actually reviewed 187 ballots because two teams of three were working side-by-side and, when we paused to see how many ballots we had reviewed, we realized we were already at 187.

3. ***Random Selection of Batches.*** To reach the 135 ballots targeted for review, the MOA Elections Team estimated a minimum of 2 batches would be required to be audited since during the processing of the election, the team scans approximately 100 ballots per batch (135/100 = 1.35). The team selected 10 batches for this audit – in the likely event that some of the batches could contain less than the 100 ballots typically scanned per batch.

   Then, staff calculated the percentage of total ballots processed in the election on two of three ballot scanning machines regularly used during elections. Those scanners are labeled ICC 1 (scanner 1) and ICC 2 (scanner 2). ICC 3 did not scan any ballots in this election and thus was not included in the audit. Based on the number of ballots scanned by each machine, the target number of batches to audit was calculated to be 2 batches from ICC1 and 8 batches from ICC2. The exact calculations are provided at the bottom of this page and the beginning of the next page.

   The method for calculating the target number of ballots follows:
   First, calculate 3% of ballots cast in the Assembly race, regardless of the number of votes cast or spread. And then, round down to the nearest 100. For example, if 4,548 ballots cast, staff rounds down to 4,500 for ease of count. Then we take 3% of the total:

   - 4,500 x .03 = 135

[1] For more detailed information on the results of the audit, see image above or Exhibit A - RLA Worksheet.

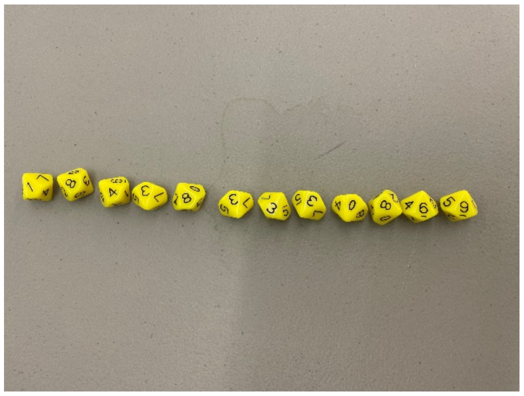
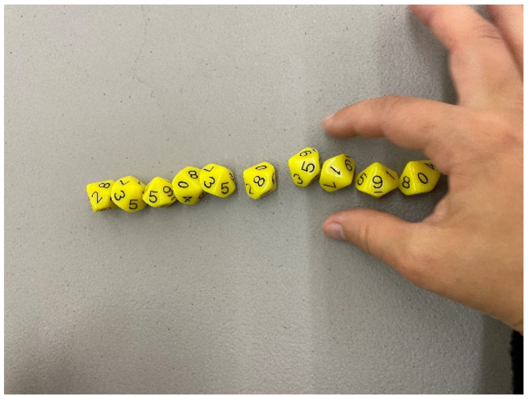
The exact calculations for the number of batches selected from each scanner are determined as follows:

- Sum the total number of batches scanned by each selected ICC. In this election, ICC 1 scanned 8 batches of ballots and ICC 2 scanned 50 batches of ballots. Thus, there were a total of 58 batches in the universe of ballots to be verified.
  Again, we did not select batches from ICC 3 because it was not used.
- Determine the percentage of total batches each ICC scanned:
  - ICC 1 = 8/58 = 14%
  - ICC 2 = 50/58 = 86%
- For each ICC selected, use the percentage of total batches each ICC scanned to determine the number of batches needed from each ICC.
  - Since 10 batches were selected for verification, the total number of batches for verification form each ICC is as follows:
    - ICC 1 = 14% of total batches x 10 batches for verification = 2
    - ICC 2 = 86% of total batches x 10 batches for verification = 8

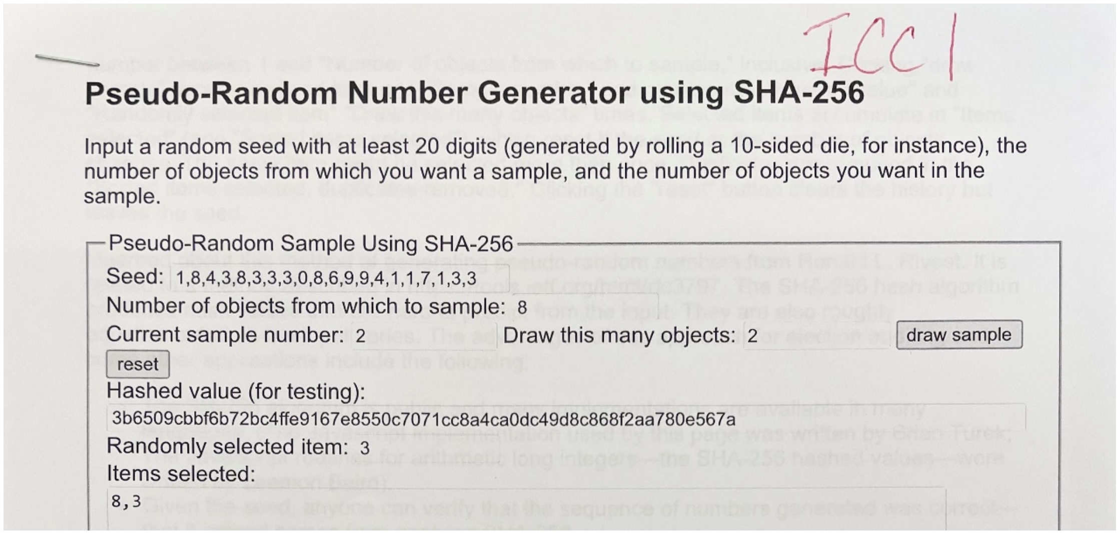
B. Use Pseudo-Random Number Generator for Random Selection of Batches.

The staff then used the Pseudo-Random Number generator at https://www.stat.berkeley.edu/~stark/Java/Html/sha256Rand.htm to randomly select the batches of ballots from each ICC. Following the instructions on the Pseudo-Random Number Generator, the selected were as follows:

(1) Roll the ten-sided dice one time, and then a second time and input all twenty numbers into the "Seed".

(2) Enter the “Seed” and other information into the random number generator and press “Draw Sample.” The result is the list of randomly selected items. Here, batches 8 and 3 were selected for ICC 1.



ICC 1

Pseudo-Random Number Generator using SHA-256

Input a random seed with at least 20 digits (generated by rolling a 10-sided die, for instance), the number of objects from which you want a sample, and the number of objects you want in the sample.

Pseudo-Random Sample Using SHA-256

Seed: 1,8,4,3,8,3,3,3,0,8,6,9,9,4,1,1,7,1,3,3

Number of objects from which to sample: 8

Current sample number: 2 Draw this many objects: 2 draw sample

reset

Hashed value (for testing):

3b6509cbbf6b72bc4ffe9167e8550c7071cc8a4ca0dc49d8c868f2aa780e567a

Randomly selected item: 3

Items selected:

8,3

The process was repeated for ICC 2 and batches 48, 43, 30, 38, 13, 1, 19 and 39 were selected.

The two batches for ICC 1 and eight batches for ICC 2 were pulled and delivered to counting teams.

C. *The Hand-Count*

The hand-count is the manual tallying of the preselected ballots.

Assembly District 1 – For each batch selected as detailed in section II.B. (Random Selection of Batches), the ballots were sorted for hand-counting by the top two candidates - Candidate 1 and Candidate 2 - and Other. The “Other” category included votes for other candidates, a blank race, or an overvote. After the ballots were sorted, the ballots were hand-counted and tallied by batch and double checked by a second election worker. The batch results were added to the RLA Worksheet. The results of the hand-count from the RLA Worksheet are as follows:

Category	**Hand-Count**
Candidate 1	80
Candidate 2	107
Total	**187**

Again, we reviewed 187 ballots, which is more than the identified 135. This was because two teams of three were working side-by-side and, when we paused to see how many ballots we’d reviewed, we realized we were already at 187. This ultimately gives us more confidence because it increases the possibility that an incorrect outcome could be detected.

D. Machine Count Verification

After the batches of ballots were hand-counted, the Cast Vote Records for the selected batches of ballots were produced and tallied. The batch totals were transferred to the RLA Worksheet[2] and are as follows:

Assembly District 1

Category	**Machine-Count Total**
Candidate 1	80
Candidate 2	107
Total	**187**

E. Comparison of the Hand-Count to the Machine Count

The third and final step in the post-election audit was to compare the hand-count to the machine count. The comparison is as follows:

Assembly District 1

Category	**Hand-Count**	**Machine-Count Total**
Candidate 1	80	80
Candidate 2	107	107
Total	**187**	**187**

The result of the post-election audit is that of 187 randomly selected ballots, the hand count and machine count of those ballots was identical. The conclusion is that the scanning, adjudication, and tabulation system performed as expected and the results of the election demonstrated the will of the voters.

Respectfully Submitted:
MOA Elections Team
Jamie Heinz, Election Administrator
Barbara A. Jones, Municipal Clerk

[2] See Exhibit A – RLA Worksheet

Exhibit A - RLA Worksheet

Scanner & Batch #	Handcount Column A - Candidate 1	Handcount Column B - Candidate 2	N=135	Machine Batch Level Results Candidate 1	Machine Batch Level Results Candidate 2	Total Race
1-8	24	23		24	23	
1-3	21	46		21	46	
2-48	0	0		0	0	
2-43	32	33		32	33	
2-30	2	5		2	5	
2-38	1	0		1	0	
Total Actual Results:						
	80	107	187	80	107	4548

Risk Limiting Audit Instructions

Purpose: To audit Election results to confirm accurate tabulation

Frequency
Once after each election.

Performed By
Core Team, Project Manager, Four Election Workers

Required Materials

- 12 Ten-Sided Dice
- 1 Six-Sided Dice
- Random Number Generator (https://www.stat.berkeley.edu/~stark/Java/Html/sha256Rand.htm)
- Ballots
- Tally Sheets
- Anomaly Log
- Adjudication System
- Completed Batch Cover Sheets "Not in Numeric Order Due to RLA"
- White, yellow, green, and red trays
- Red pens, markers

A. Identify races or measures to audit.

1. Mayor's race. Will count top two candidates only.
2. Using six-sided dice, randomly select 1 of the Assembly races, by District number. Will count top two candidates only.
3. Using six-sided dice, randomly select 1 or more propositions each year.

B. Identify Target Number of ballots per race or measure.

1. Calculate 3% of ballots cast in the Mayor's race or in the Proposition selected, regardless of the number of votes cast or spread. Round down to nearest 1,000. E.g., change 71,345 to 71,000 for ease of count. (Use the same methodology for a School Board race.)

2. Because the potential number of ballots cast in an Assembly race is approximately 1/6 of the total ballots cast in Municipal wide races or measures, calculate 1.5% of total votes cast in the Assembly race. Round down to nearest 100 for ease of count.

C. **Randomly select batches in the race.**

1. Based on the number of ballots to be reviewed per race or measure as determined in paragraph B., if approximately 100 ballots were scanned per batch, select the total number of batches to comfortably reach the 3% or 1.5% of ballots to be targeted.

FOR EXAMPLE: Assume 75,000 ballots cast in mayor's race X 3% = 2,250 ballots to verify. Assuming approximately 100 ballots were scanned per batch, randomly select approximately 25 batches, (25x100 = 2,500) which would exceed the 2,250 ballots to be selected for verification.

2. Determine the total number of batches scanned by each ICC.

3. Determine the percentage of total batches each ICC scanned.

4. For each ICC selected, use the percentage of total batches each ICC scanned to determine the random number of batches needed from each ICC, and then to determine which batch numbers for each ICC to pull.

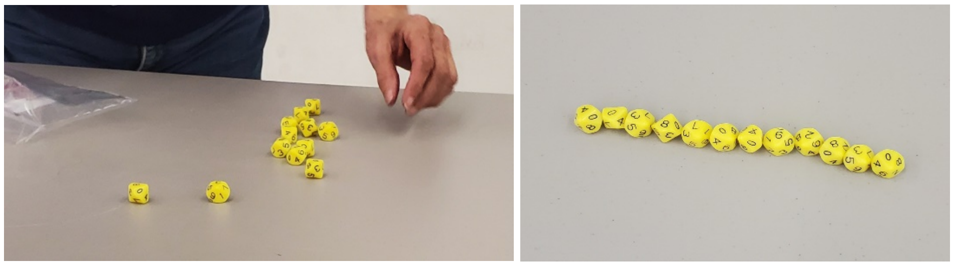
   a. Use the percentage of total batches scanned at each ICC in Step C.3., x the number of batches targeted to be verified in Step C.1.

   - For example, ICC 1 = 60% of total batches x 25 batches = 15 batches would be selected for verification from ICC 1.

   b. Use Pseudo-Random Number Generator (site link above) and follow instructions to generate a random selection of batches to pull for the audit. (See attached example copy of Pseudo-Random Number Generator page.)

   c. Conduct the following steps for each ICC:

      (1) Roll the ten-sided dice 10 times, and then 10 times again and input this information into the "Seed".



(2) Enter the "Number of objects from which to sample" = the number for this ICC from Step C.2, the total number of batches scanned by the ICC.

(3) Enter the "Draw this many objects" = the number for this ICC from Step C.4.a, the number of batches needed from this ICC. (The computer generates the "Current sample number." Do not enter the number in that field.)

(4) Hit the "draw sample" button.

(a) Count to confirm the number of "Items Selected" if the same as the "Draw this many objects." If not, hit the "reset" button and run again.

(b) Print a copy of the results of the Pseudo-Random Number Generator report for each ICC. (You cannot save it.)

d. Repeat the process for other ICCs.

e. Tips for the Pseudo-Random Number Generator:

- Number of Objects to Sample = C.2. above, the total number of batches scanned by the ICC.
- Draw this Many Objects = C.4.a., above, the random number of batches needed from each ICC.
- Count to make sure the number of batches equals C.4.a. and is the same as the request to "Draw this Many Objects." If not, press reset and reenter the numbers and redo process.
- Reset the Pseudo-Random Number Generator and repeat the steps to generate batches to be pulled for the other ICCs.

D. Pull the batch boxes from the vault.

Pull the boxes from the vault containing the batch numbers determined from the results of the Pseudo-Random Number Generator as detailed in the printed report in Step C.4.c.(4)(b). Keep the boxes from each ICC separated on their own carts.

E. Tally Each Batch by Hand Count, then Sort

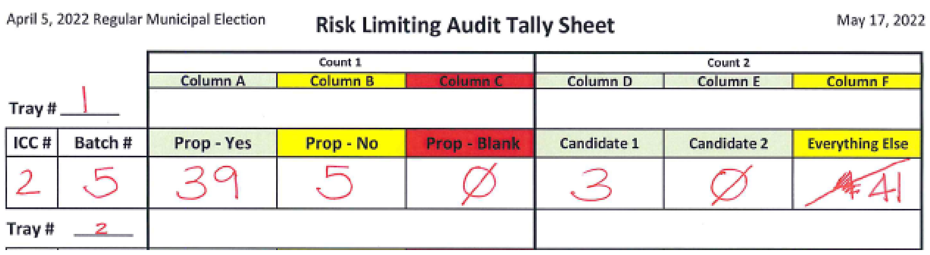
1. Working in teams of two, starting with the lowest batch number, pull the first randomly selected batch by removing the batch cover sheet and all of the batch's ballots from the batch box.

2. Complete the RLA Batch Cover Sheet (see attached sample). (Keep the original Batch Cover Sheet and both the new RLA Batch Cover Sheet and the original will be retained.)

3. For each batch, there will be two counts for each batch, (1) the "yes" votes and "no" votes for the proposition, and (2) the votes for the top two candidates.

A. Start with the proposition. Sort the ballots from the batch into three trays categories – "Yes," "No," and other (undervote, overvote).

Once sorted, count the "Yes," "No," and "Other" and write the numbers on the Risk Limiting Audit Tally Sheet.



April 5, 2022 Regular Municipal Election

Risk Limiting Audit Tally Sheet

May 17, 2022

Tray # 1

		Count 1			Count 2		
		Column A	Column B	Column C	Column D	Column E	Column F
ICC #	Batch #	Prop - Yes	Prop - No	Prop - Blank	Candidate 1	Candidate 2	Everything Else
2	5	39	5	Ø	3	Ø	~~4~~ 41

Tray # 2

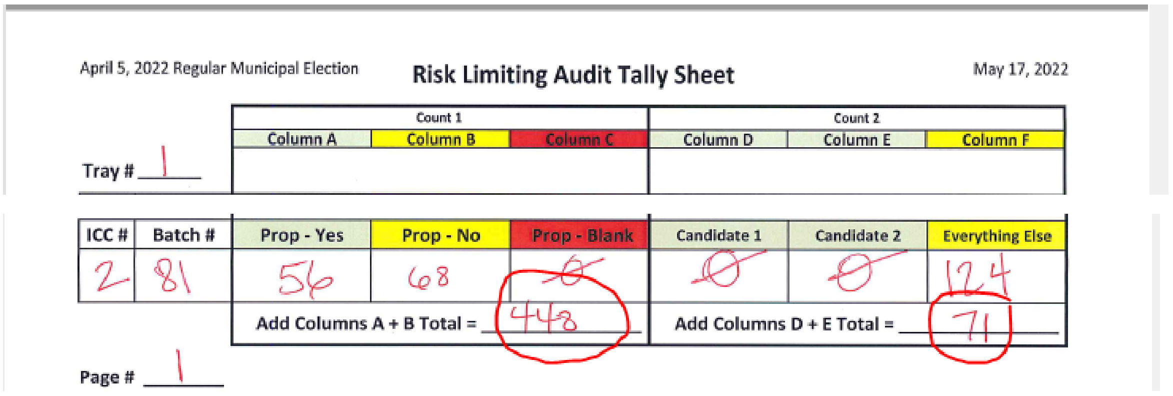
B. Continue with the candidate race. Sort the ballots in the batch into three trays appropriately – Candidate 1, Candidate 2, and other (undervote, overvote, other candidate).

Once sorted, count the Candidate 1 votes, the Candidate 2 votes, and the "other" votes, and write these numbers on the RLA tally sheet. Write the totals on the RLA Batch Cover Sheet.

4. Transfer to Adjudication. Place the Candidate 1 and Candidate 2 ballots crosswise in 1 tray; place the other category crosswise or in another tray. Document the batch number on the top of the ballots. Place the RLA batch cover sheet on top of the tray(s). Place on the ICC rack in the cage for the adjudication verification.

F. **Repeat step E for each randomly selected batch. Continue to tally up the ballot count from each randomly selected batch on the RLA Tally Sheet until 3% or 1.5% of ballots is reached.** (See Risk Limiting Audit Tally Sheet and Risk Limiting Audit Tally Adding Sheet.)

1. Calculate the page total votes marked YES, NO, and Other and add these numbers together at the bottom of the page on the Risk Limiting Audit Tally Sheet. Repeat and calculate the page total votes marked for Candidate 1, Candidate 2, and Other, and add to the tally sheet.



April 5, 2022 Regular Municipal Election **Risk Limiting Audit Tally Sheet** May 17, 2022

Tray # 1

	Count 1			Count 2		
	Column A	Column B	Column C	Column D	Column E	Column F

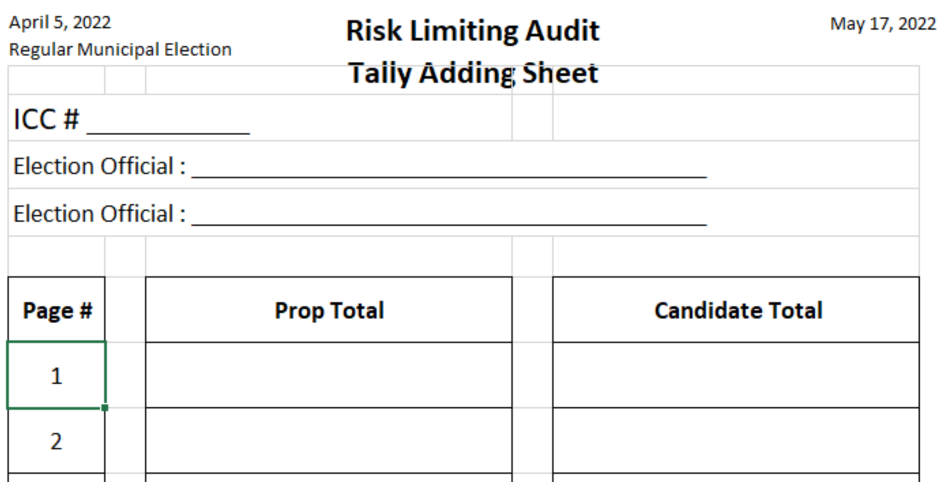
ICC #	Batch #	Prop - Yes	Prop - No	Prop - Blank	Candidate 1	Candidate 2	Everything Else
2	81	56	68	0	0	0	124

Add Columns A + B Total = 448

Add Columns D + E Total = 71

Page # 1

2. Transfer the page totals from the Risk Limiting Audit Tally Sheet to the Risk Limiting Audit Tally Adding Sheet, for each ICCs.



April 5, 2022
Regular Municipal Election

Risk Limiting Audit Tally Adding Sheet

May 17, 2022

ICC # ____________

Election Official : ______________________________________

Election Official : ______________________________________

Page #	Prop Total	Candidate Total
1		
2		

3. When the totals for both ICCs equal the total number of ballots sought to be verified in the election, stop tallying.

a. If, after all batches and ballots selected, the totals for both ICCs does NOT equal the total number of ballots sought to verified in the election, randomly select additional batches if needed.

G. Adjudication Verification

With one person at each adjudication computer reviewing separate batches, open the batch and compare the votes marked on the actual ballot to the candidate and propositions marked in the image of the ballot on the screen and then verify the ballot was adjudicated correctly on the audit mark page of the image. For any anomalies, first confirm you are reviewing the correct ballot number. If you are, and still see an anomaly, communicate with the Supervising Election Official\Deputy Municipal Clerk –

Elections for further review and resolution. Note any unresolved anomalies on the Anomaly Log.

When finished with each batch, wrap that batch in the RLA cover sheet over the original batch cover sheet indicating that the batch is out of numeric order due to the Risk Limiting Audit and place the batch back in the box it originated from.

REFERENCE: Number of total estimated ballots.

Assuming an audit of one proposition each year, and one major race (either mayor or a randomly selected assembly seat), the smallest number of total ballots to hand count each year could look like this, based on a sampling of 2020 and 2021 election results:

<u>Year 1</u>

Assembly seat: 20,000 x .015 = 300
Prop: 71,000 x .03 = 2,130*
Total ballots audited = greater of 300 or 2,130: 2,130

<u>Year 2</u>

Mayor: 75,000 x .03 = 2,250
Prop: 71,000 x .03 = 2,130*
Total ballots audited = greater of 2,250 or 2,130: 2,250

*This is based on the assumption that the proposition selected will be an areawide question. Since current practice is to place areawide questions ahead of other questions on the ballot, the random selection of a question using a 6-sided dice should provide this outcome more often than not.

ICC 1

Pseudo-Random Number Generator using SHA-256

Input a random seed with at least 20 digits (generated by rolling a 10-sided die, for instance), the number of objects from which you want a sample, and the number of objects you want in the sample.

Pseudo-Random Sample Using SHA-256

Seed: 1,8,4,3,8,3,3,3,0,8,6,9,9,4,1,1,7,1,3,3

Number of objects from which to sample: 8

Current sample number: 2 Draw this many objects: 2 draw sample

reset

Hashed value (for testing):

3b6509cbbf6b72bc4ffe9167e8550c7071cc8a4ca0dc49d8c868f2aa780e567a

Randomly selected item: 3

Items selected:

```
8,3
```

Sorted items selected:

```
3,8
```

Sorted items selected, duplicates removed:

```
3,8
```

What this does

The "seed," concatenated with a comma and the "Sample number," is passed through the SHA-256 hash function. The result is displayed as "Hashed value (for testing)". The hashed value, interpreted as a hexadecimal number, is divided by "Number of objects from which to sample." One is added to the remainder of that division to get "Randomly selected item," which will be a number between 1 and "Number of objects from which to sample," inclusive. Clicking "draw sample" successively adds one to "Sample number" and recomputes "Hashed value" and "Randomly selected item" "Draw this many objects" times. Selected items accumulate in "Items selected" (and "Sorted items selected"), which reset if the seed or the number of objects changes. The same item might be selected more than once. Duplicates are removed in the "Sorted items selected, duplicates removed." Clicking the "reset" button clears the history but leaves the seed.

I learned about this method of generating pseudo-random numbers from Ronald L. Rivest. It is related to a method described in https://tools.ietf.org/html/rfc3797. The SHA-256 hash algorithm produces hash values that are hard to predict from the input. They are also roughly equidistributed as the input varies. The advantages of this approach for election auditing and some other applications include the following:

- The SH-256 algorithm is public and many implementations are available in many languages. (The Javascript implementation used by this page was written by Brian Turek; The JavaScript routines for arithmetic long integers—the SHA-256 hashed values—were written by Leemon Baird).
- Given the seed, anyone can verify that the sequence of numbers generated was correct—that it indeed comes from applying SHA-256.
- Unless the seed is known, the sequence of values generated is unpredictable (so the result is hard to "game"). It is very hard to distinguish the output from independent, uniformly distributed samples.

For comparison, a reference implementation of this approach in Python written by Ronald L. Rivest is available at https://people.csail.mit.edu/rivest/sampler.py.

P.B. Stark, statistics.berkeley.edu/~stark. https://statistics.berkeley.edu/~stark/Java/Html/auditRand.htm Last modified 11 January 2017.

ICC 2

Pseudo-Random Number Generator using SHA-256

Input a random seed with at least 20 digits (generated by rolling a 10-sided die, for instance), the number of objects from which you want a sample, and the number of objects you want in the sample.

Pseudo-Random Sample Using SHA-256

Seed: 2,3,5,0,3,8,5,1,9,0,2,6,0,6,3,8,0,9,9,3

Number of objects from which to sample: 50

Current sample number: 8 Draw this many objects: 8 draw sample

reset

Hashed value (for testing):

31c0cd5a18cbe34df3094e0c6b691d20e3bcdf02743294cb99cb027bab0fb204

Randomly selected item: 39

Items selected:

```
48,43,30,48,38,13,1,19,39
```

Sorted items selected:

```
1,13,19,30,38,39,43,48,48
```

Sorted items selected, duplicates removed:

```
1,13,19,30,38,39,43,48
```

What this does

The "seed," concatenated with a comma and the "Sample number," is passed through the SHA-256 hash function. The result is displayed as "Hashed value (for testing)". The hashed value, interpreted as a hexadecimal number, is divided by "Number of objects from which to sample." One is added to the remainder of that division to get "Randomly selected item," which will be a number between 1 and "Number of objects from which to sample," inclusive. Clicking "draw sample" successively adds one to "Sample number" and recomputes "Hashed value" and "Randomly selected item" "Draw this many objects" times. Selected items accumulate in "Items selected" (and "Sorted items selected"), which reset if the seed or the number of objects changes. The same item might be selected more than once. Duplicates are removed in the "Sorted items selected, duplicates removed." Clicking the "reset" button clears the history but leaves the seed.

I learned about this method of generating pseudo-random numbers from Ronald L. Rivest. It is related to a method described in https://tools.ietf.org/html/rfc3797. The SHA-256 hash algorithm produces hash values that are hard to predict from the input. They are also roughly equidistributed as the input varies. The advantages of this approach for election auditing and some other applications include the following:

- The SH-256 algorithm is public and many implementations are available in many languages. (The Javascript implementation used by this page was written by Brian Turek; The JavaScript routines for arithmetic long integers—the SHA-256 hashed values—were written by Leemon Baird).
- Given the seed, anyone can verify that the sequence of numbers generated was correct—that it indeed comes from applying SHA-256.
- Unless the seed is known, the sequence of values generated is unpredictable (so the result is hard to "game"). It is very hard to distinguish the output from independent, uniformly distributed samples.

For comparison, a reference implementation of this approach in Python written by Ronald L. Rivest is available at https://people.csail.mit.edu/rivest/sampler.py.

P.B. Stark, statistics.berkeley.edu/~stark. https://statistics.berkeley.edu/~stark/Java/Html/auditRand.htm Last modified 11 January 2017.