Privacy Aware Data Aggregation with Application in Voting Systems

Arash Bayat

26 Jan 2025

Abstract

Turning raw data into informative insights is an essential procedure. When data is public, the aggregated information is trusted it can be reproduced. However, it is difficult to achieve trust when the data is private. An obvious example is the voting system where votes are kept in secret, yet everyone needs to trust the count. The problem is to achieve privacy and trust together.

Traditional paper-based voting satisfy privacy because the voter's identity is detached from the ballot. However, trust is not satisfied. It relies on the fact that a considerable change in the count requires a large number of election staff to be involved (which is difficult to achieve). Apart from being expensive to perform, post-election protests show that paper-based voting is not trusted as claimed. Paper-based voting remains the only effective solution as it is much more difficult to address privacy-trust problem in electronic voting. Advanced technologies in digital security (i.e. homomorphic encryption, block chain, etc.) have been widely used to address this problem [1,2,3,4,5,6,7,8,9,10,11].

This article introduces a novel approach that satisfies both trust and privacy in the aggregation of private data. Tow implementations of the proposed approach for the voting system are also described. In the proposed method, each party encrypts its data and publishes the encrypted data publicly. Aggregation on encrypted data results in decrypted information without the need for decryption process.

1 Simple aggregation

Although the voting system is the ideal example for the aggregation of private data, here a simpler example is used to better explain the proposed idea. In this example, there are five parties (u_1 to u_5) each producing a private number. They would like to find the sum of all private numbers without sharing them (e.g. to calculate the average). In Table 1, D is the list of private numbers (data) and K is the list of encryption keys. Each party encrypts the data by adding the encryption key to the data and publishes the encrypted data in E (i.e. $28 + 12 = 40$ for u_1). As shown in Equation 1, the sum of D is equal to the sum of E for any K if the sum of K is equal to 0 where $N = |U|$ is the number of parties (proven in Equation 2). **Encrypted data (E) can be shared with all parties and used to calculate the aggregated information (sum of D) without revealing the actual data (D).**

In the above example, each key (k_i) must be known only to the corresponding party (u_i). However, all parties need to ensure that the sum of K is equal to 0 and other parties perform encryption as expected. Implementing the key generation and encryption process is a challenging problem that is addressed later and in the context of voting systems (see Section 6).

Assuming that the encryption process is done fairly, the proposed method satisfies trust and privacy together in data aggregation. Trust is satisfied as all parties can compute the aggregated information by sharing E. Privacy is satisfied as E does not reveal any information about the data (D) or encryption keys (K).

U	u_1	u_2	u_3	u_4	u_5	Sum
D	28	-85	-18	64	49	38
K	12	-9	73	-44	-32	0
E	40	-94	55	20	17	38

Table 1: An example where D is encrypted to E by adding K ($e_i = d_i + k_i$). The sum of D and E are equal because the sum of K is 0

$$\sum_{i=1}^{N} d_i = \sum_{i=1}^{N} e_i \quad \forall \quad \sum_{i=1}^{N} k_i = 0 \tag{1}$$

$$\sum_{i=1}^{N} d_i = \sum_{i=1}^{N} e_i = \sum_{i=1}^{N} (d_i + k_i) = \sum_{i=1}^{N} d_i + \sum_{i=1}^{N} k_i = \sum_{i=1}^{N} d_i + 0 = \sum_{i=1}^{N} d_i \tag{2}$$

In the above example, if the sum of K is not equal to 0, the sum of D can be calculated as the sum of E subtracted by the sum of K. This is shown in Equation 3.

$$\sum_{i=1}^{N} d_i = \sum_{i=1}^{N} e_i - \sum_{i=1}^{N} k_i \tag{3}$$

The encryption key generation and encryption function depend on the aggregation function. Equation 4 shows another example in which the aggregation function is to find the product of all numbers. In this example, encryption is performed by multiplying d_i by k_i and the product of K is equal to 1. Equation 5 is the proof for Equation 4.

$$\prod_{i=1}^{N} d_i = \prod_{i=1}^{N} e_i \quad \forall \quad \prod_{i=1}^{N} k_i = 1 \tag{4}$$

$$\prod_{i=1}^{N} d_i = \prod_{i=1}^{N} e_i = \prod_{i=1}^{N} (d_i \times k_i) = \prod_{i=1}^{N} d_i \times \prod_{i=1}^{N} k_i = \prod_{i=1}^{N} d_i \times 1 = \prod_{i=1}^{N} d_i \tag{5}$$

In the second example, if the product of K is not equal to 1, the product of D can be calculated as the product of E divided by the product of K. This is shown in Equation 6.

$$\prod_{i=1}^{N} d_i = \prod_{i=1}^{N} e_i \div \prod_{i=1}^{N} k_i \tag{6}$$

2 Randomness as encryption key

This section describes how randomness is used as an encryption key. Assume that the aggregation function is sum and the key generation logic is running on a trusted machine. A uniform random number generator with the range $-R$ to R can be used to generate a key for each party. If the number of parties (N) is known, the last key can be adjusted to ensure that the sum K is equal to 0. Otherwise, the sum of K is expected to be close to 0 since the distribution of K should resemble a symmetrical distribution around 0 (uniform randomness), especially when the number of parties increases. Since sum K is approximately 0, sum E is approximately equal to sum D.

R impacts the privacy of the data and the accuracy of the aggregated information. Table 2 demonstrates two examples in which K_A and K_B are produced with $R = 1$ and $R = 10$ and used to calculate E_A and E_B, respectively. The smaller R (compare K_A and E_A with K_B and E_B) results in less privacy as the original data (D) can be predicted with greater precision given the encrypted data (E). The smaller R also results in K being closer to 0 (i.e., the sum of E closer to the sum of D) In this situation R impact privacy and accuracy (trust).

U	u_1	u_2	u_3	u_4	u_5	Sum
D	28	-85	-18	64	49	38
K_A	1	0	1	-1	1	2
E_A	29	-85	-17	63	50	40
K_B	-4	3	-5	1	-9	-14
E_B	24	-82	-23	65	40	24

Table 2: Example sum

In simple words, the idea is to mix data with randomness in such a way that the aggregation function eliminates randomness in the aggregated information. In other words, randomness is the encryption key and aggregation on the encrypted data results in the unencrypted information. The aggregation function determines how to use randomness to encrypt the data.

3 Voting system aggregation

This section explains how to utilize the proposed method in a simple voting system. Assume that there are two candidates (Alice and Bob) and each individual can vote for one of the candidates. Also, there are two imaginary candidates (Red and Blue). Votes (for Alice and Bob) are transformed into imaginary votes (for Red and Blue) randomly given the probabilities matrix shown in Equation 7. Table 3 shows the encryption process. The key is generated using uniform randomness ranging from 1 to 10 (integer and inclusive). Alice is transformed to Red if the key is less than 8 (70% probability), otherwise to Blue (30% probability). Bob is transformed to Red if the key is less than 4 (30% probability), otherwise Blue (70% probability).

In the right-most column 'Counts', the actual number of votes for Alice and Bob is assumed to be 600 and 400, respectively. The rest of the values in the count columns assume theoretical probability. Each number in key occurs exactly 100 times. Exactly 70% of Alice votes (420) and 30% Bobs votes (120) are turned into votes for Red. Also, exactly 30% of Alice votes (180) and 70% Bobs votes (280) are turned into votes for Blue.

$$P = \begin{bmatrix} & Red & Blue \\ Alice & 0.7 & 0.3 \\ Bob & 0.3 & 0.7 \end{bmatrix} \tag{7}$$

U	u_1	u_2	$\dots$	u_{999}	u_{1000}	Counts
D	Alice	Bob	$\dots$	Bob	Alice	Alice:600, Bob:400
K	4	8	$\dots$	2	5	1:100, 2:100, ... 10:100
E	Red	Blue	$\dots$	Red	Blue	Red:540, Blue:460

Table 3: of

The relation between the number of actual votes and imaginary votes forms a system of equations shown in Equation 8 that holds if theoretical probability is assumed (the name of the candidate or imaginary candidate stands for the number of votes). Equation 9 shows the same system of equations in the form of a matrix operation and explains how the number of votes for Alice and Bob can be computed using the number of votes for Red and Blue and the inversion of the probability matrix.

$$\begin{cases} 0.7 \times Alice + 0.3 \times Bob = Red \\ 0.3 \times Alice + 0.7 \times Bon = Blue \end{cases} \tag{8}$$

$$\begin{cases} \begin{bmatrix} Alice \\ Bob \end{bmatrix} \times \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix} = \begin{bmatrix} Red \\ Blue \end{bmatrix} \\ \begin{bmatrix} Alice \\ Bob \end{bmatrix} = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{bmatrix}^{-1} \times \begin{bmatrix} Red \\ Blue \end{bmatrix} \end{cases} \tag{9}$$

The imaginary vote (Red or Blue) is attached to voter's identity and is published publicly without revealing the actual candidate (Alice or Bob). The number of vote for the actual candidate can be calculated given the number of votes for Red and Blue which are publicly available. Thus, the voting system satisfies both privacy and trust. Experimental probability adds an approximation to the equality in Equation 9. The approximation is discussed in Section 4.

Equation 10 generalize the above example for n candidates and imaginary candidates where c_x and i_y are the number of votes for candidate x and imaginary candidate y, respectively. $p_{x,y}$ is the probability that candidate x is transformed to candidate y

$$\begin{cases} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{bmatrix} \times \begin{bmatrix} p_{c_1,i_1} & p_{c_1,i_2} & p_{c_1,i_3} & \cdots & p_{c_1,i_n} \\ p_{c_2,i_1} & p_{c_2,i_2} & p_{c_2,i_3} & \cdots & p_{c_2,i_n} \\ p_{c_3,i_1} & p_{c_3,i_2} & p_{c_3,i_3} & \cdots & p_{c_3,i_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{c_n,i_1} & p_{c_n,i_2} & p_{c_n,i_3} & \cdots & p_{c_n,i_n} \end{bmatrix} = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ \vdots \\ i_n \end{bmatrix} \\ Simplified : C \times P = I \rightarrow C = P^{-1} \times I \end{cases} \tag{10}$$

The probability matrix (P) should be invertible (that is, the determinant of P is greater than 0). Also, there should be less than $n - 1$ zero in each row or column (i.e. each candidate transforms to at least two imaginary candidates and at least two candidates to be transformed to each imaginary candidate). The second condition is set to maintain the privacy of the voters (i.e. votes for an imaginary candidate can not be linked to a specific candidate). Equation 11 shows a possible matrix that works for any number of candidates n where $\alpha > 1/n$ and $\beta = (1-\alpha)/(n-1)$. Equation 7 is an example of such probability matrix where $\alpha = 0.7$ and $\beta = 0.3$.

It is possible to have more imaginary candidates than actual candidates. In that case, there are more equations than variables. However, P must be chosen such that the system of equations is solvable (i.e. the number of votes for the candidate can be computed given the number of votes for imaginary candidates).

$$P = \begin{bmatrix} \alpha & \beta & \beta & \cdots & \beta \\ \beta & \alpha & \beta & \cdots & \beta \\ \beta & \beta & \alpha & \cdots & \beta \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta & \beta & \beta & \cdots & \alpha \end{bmatrix} \tag{11}$$

4 α controls a tunable trade-off between the privacy and trust

In the proposed matrix, each candidate is biased towards an imaginary candidate with probability α. For example, Alice is biased to Red (Equation 7), which means that if an imaginary vote is Red, it is more likely that the actual candidate is Alice.

Equation 12 shows two other probability matrix P_1 and P_2 with extreme α (high and low, respectively). When α is high (P_1) almost all Red votes are contributed by Alice. This violates privacy of the voters, as anyone voted for Red is most likely to vote for Alice. When α is low (P_2) privacy is strongly satisfied (Red votes contributed by Alice and Bob almost equal). However, a lower α results in a lack of precision. This is experimentally shown in Section 5. α controls a tunable trade-off between privacy and trust.

There might be a probability matrix that increases both trust and privacy together. Also, in case votes are transformed so that the probabilities in P are exactly implemented (experimental probability is equal to theoretical probability), there would be no error in the system and α can be reduced to maximize privacy. Depending on the actual implementation of the voting system, it may be achievable at the cost of some dependencies in the system. This article assumes that each individual submits the vote independently at any given point in time.

$$\begin{cases} P_1 = \begin{bmatrix} & Red & Blue \\ Alice & 0.99 & 0.01 \\ Bob & 0.01 & 0.99 \end{bmatrix} \\ P_2 = \begin{bmatrix} & Red & Blue \\ Alice & 0.51 & 0.49 \\ Bob & 0.49 & 0.51 \end{bmatrix} \end{cases} \tag{12}$$

5 Experimental analysis

In order to measure the accuracy of the system described in Section 3 a Python program is written and attached as a supplementary data file (see Section 7). The program takes the parameter listed below to perform a simulation and transforms each vote into an imaginary vote using a uniform random number generator. Finally, it compares the computed number of votes for each candidate with the real number of votes to see how much error is introduced due to randomness. The simulator calculates the percentage error rate for each candidate. For example, if the computed number of votes for Alice is 1100 and the actual number of votes is 1000, then the percentage error rate for Alice is $((1100 - 1000)/1000) * 100 = 10\%$. Since each simulation might be slightly different from the other (due to randomness), each simulation (for a given parameter) is repeated S times. Summary statistics are calculated for the percentage error rate of each candidate and also the percent error rate of all candidate combined. The Python program exports the full simulation details (see Section 7). However, this report only presents the mean and standard deviation (std) of the percent error rate of all candidates combined as a measure of the precision of the system.

Parameters for each simulation include the followings

- N: Number of Voters
- C: Number of candidate
- V: Number of votes per candidate

- α: As described in Section 3
- S: Number of reparations per simulation

Here, five experiments are performed, each of which demonstrates the effect of one of the above parameters by changing its value and performing the simulation. Table 4 summarise the value of the simulation parameter. The complete simulation results and exact simulation parameters are available in Section 7.

Experiment	Parameters				
	Number of Voters (N)	Number of Candidates (C)	Votes per Candidate (V)	α	Number of Simulation Repeats (S)
1	100,000	2	Equal	0.7	**5** **10** **50** **100**
2	100,000	2	**50%-50%** **60%-40%** **70%-30%** **80%-20%** **90%-10%**	0.7	100
3	1000,000	**2** **3** **4** **5** **6**	Equal	2β	100
4	**1000** **10,000** **100,000** **1000,000**	2	Equal	0.7	100
5	100,000	2	Equal	**0.55** **0.60** **0.65** **0.70** **0.75** **0.80** **0.85** **0.90** **0.95**	100

Table 4: Parameter used in each of five experiments. The parameter that is studied in each experiment is written in bold and as a list. The word 'Equal' in the column 'Votes per Candidate' means that total number of votes are equally distributed across candidate.

Experiment 1: Number of simulation repetitions

This experiment is to find out how many times the simulation needs to be repeated to ensure that the statistics on the percentage error rate are accurate. The number of repetitions (S) is varied between 5, 10, 50 and 100. The experiment is executed 10 times (T_1 to T_{10}). The standard deviation (STD) is calculated over the 10 experiments for both the mean and standard deviation (std) of the percentage error rate. Note that the capitol STD is used for the standard deviation over 10 executions of the experiment, and the lower case std is used for the standard deviation of the percentage error rate of S simulation repetitions. The results are shown in Table 5. Figure 1a shows the STD of both the mean and the std of the percentage error rate is reduced when the number of repetitions per simulation increases.

Statistic	S	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	**STD**
mean	5	0.764	0.739	0.968	0.492	0.529	0.368	0.377	0.593	0.476	0.573	0.188
	10	0.487	0.634	0.579	0.501	0.598	0.516	0.590	0.778	0.434	0.488	0.099
	50	0.548	0.582	0.612	0.420	0.494	0.535	0.542	0.583	0.572	0.590	0.056
	100	0.563	0.554	0.609	0.565	0.615	0.561	0.495	0.569	0.571	0.630	0.038
std	5	0.432	0.855	0.331	0.323	0.422	0.210	0.461	0.590	0.356	0.309	0.182
	10	0.305	0.459	0.287	0.280	0.263	0.434	0.304	0.527	0.299	0.388	0.091
	50	0.404	0.424	0.454	0.368	0.419	0.415	0.415	0.479	0.325	0.464	0.045
	100	0.392	0.431	0.517	0.419	0.419	0.393	0.409	0.463	0.477	0.420	0.040

Table 5: How many simulation repetitions is needed for a stable statistics of percentage error rate

Experiment 2: Number of votes per candidate

This experiment studies the effect of the percentage of votes for each candidate on the percentage error rate statistic (over all candidates). 100,000 votes are distributed between two candidates where the first candidate takes 90% to 50% (with 10% intervals) of the total votes. As shown in Figure 1b, the lower error occurs when the candidate takes an equal number of votes.

Experiment 3: Number of candidates

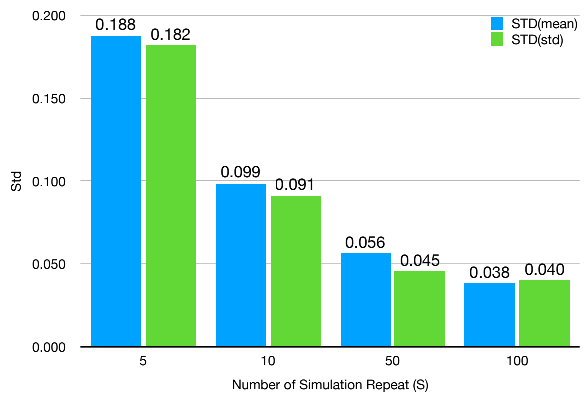
This experiment shows the effect of the number of candidates on accuracy. The number of candidates ranges from 2 to 6. In addition, the number of voters increased to 1 million to prevent the lack of precision due to the lack of votes per candidate. To be fair, α is also varied to be 2β (that is, $(1/(C+1)) * 2$). As shown in Figure 1c, the error rate was reduced as the number of candidates was reduced. The highest accuracy achieved with having only two candidates.

Experiment 4: Number of voters

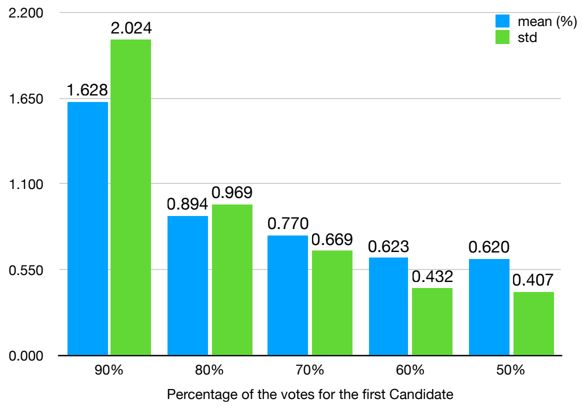
This experiment shows that the error rate reduced as the number of voters increased. Figure 1d demonsterate the accuracy when the number of voters varies.

Experiment 5: The effect of α

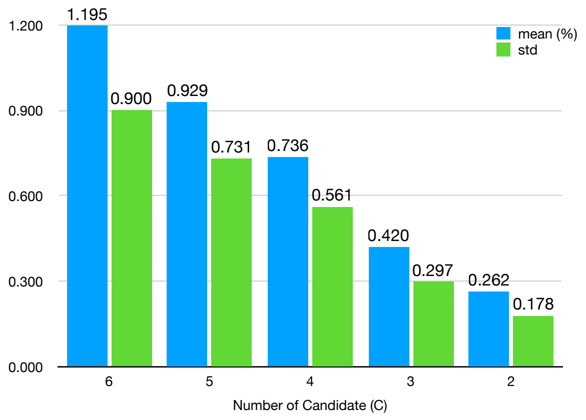
As mentioned in Section 4, the higher α results in higher accuracy (trust) but also reduces the privacy of the voter since each imaginary candidate strongly related to a specific candidate. Figure 1e shows the effect of α on the accuracy.



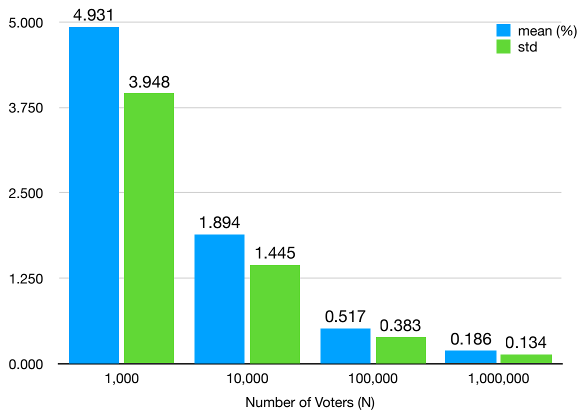
(a)



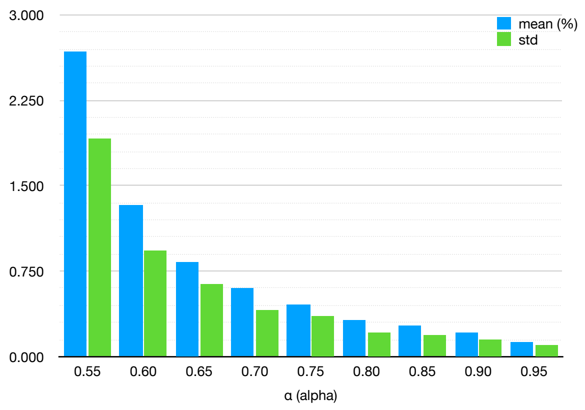
(b)



(c)



(d)



(e)

Figure 1: The effect of different simulation parameter on the accuracy

6 Implementation of voting system

This section introduces two implementations of the voting system described in Section 3 after clarifying some key topics in relation to the implementation. The following data are involved when an individual acts to vote.

- C: The name of the candidate (option) the individual selected
- ID: The identity of the individual
- R: The random number used to transform the selected candidate (C) to an imaginary candidate (I)
- I: The imaginary candidate after transformation.

The goal is to prevent exposure of C and ID together, as otherwise the privacy of the voter is violated. Instead, I and ID must be public so that everyone can count the votes. Also, consider that R, I and ID must not be exposed together as C can be computed given R and I.

Note that I should be generated based on the given probability matrix and uniform randomness, but not any other factor. However, C and R can be chosen in such a way that the resulting I is in favour of a specific candidate that is unfair. Since C is known to the voter, R must be generated by the system. The voter must expose C to the system without knowing R. However, the voter must know R is determined by the system before the voter exposes C to the system (i.e., the system provides a signature of R to the individual before C is provided).

6.1 Mechanical implementation

Imagine there are two candidates, Alice and Bob, and two fair dice, each corresponding to a candidate. Alice's dice has four faces red and two faces blue. Bob's dice has two faces red and four faces blue. There is no other mark on the dice. In the voting environment, voters throw the dice for the candidate of their choice (C) only once. The voting machine scans the color of the top surface of the dice (without knowing what dice is used) and publishes the colour (I) along with the voter's identity (ID). This will perform the transformation of the candidate into an imaginary candidate with the probability matrix shown in Equation 13. From when the voters pick the dice to when the dice is returned to its initial place, the actual candidate selected by the voters C is exposed along with his identity ID. Thus, during this time, the voter must not be observed.

$$P = \begin{bmatrix} & \mathit{Red} & \mathit{Blue} \\ \mathit{Alice} & 2/3 & 1/3 \\ \mathit{Bob} & 1/3 & 2/3 \end{bmatrix} \tag{13}$$

6.2 Digital implementation

Imagine there are two computers X and Y. X has limited input/output and does not have persistent memory (all data are wiped on restart). The voting actions are as follows and are shown in Figure 2:

1. The voter reset Y
2. Y generate new keypair (K_1, K_2) and share public key (K_1) with the voter.
3. The voter generate random salt A and share K_1 along with hash of $[C, A]$ (H_C) with X

4. X generate a random number R and share hash of $[R, H_C]$ (H_R) with the voter
5. The voter shares $[C, A]$ with X
6. X
   - validate $[C, A]$ using H_C
   - computes I given C, R and hardcoded probability matrix (known publicly)
   - produces a random salt (B)
   - encrypts $[I, B]$ using K_1 (E_I) and share it with the voter
7. The voter provide the E_I to Y along with voter's identity ID
8. Y decrypt I and publish it along ID publicly.
9. Y share B with the voter
10. The voter provide B to the X
11. when B is provided by the voter, X provides R and I to the voter.
12. The voter validates R using H_C and H_R and ensures that I is calculated correctly.
13. The voter resets X to wipe all its data.

Note that X observes the candidate selected by the voter (C) but not the identity of the voter ID. On the other hand, Y only observes I and ID. This ensures that C and ID are never exposed to the same computer. There should be no communication channel between X and Y that can be used secretly without notified by the voter.

7 Supplementary Data File: Python simulation and its results

The supplementary data is a zip file that contains the following files and directories:

- **README.md:** Explain how the simulator works and how to re-run all five experiments described in Section 5.
- **sim_logic.py:** Python Simulator
- **parse.py:** Read simulation results (yaml file) of multiple simulation repeat and produce a summary CSV used in this article
- **experiment:** A directory containing parameters and simulation results for all experiments

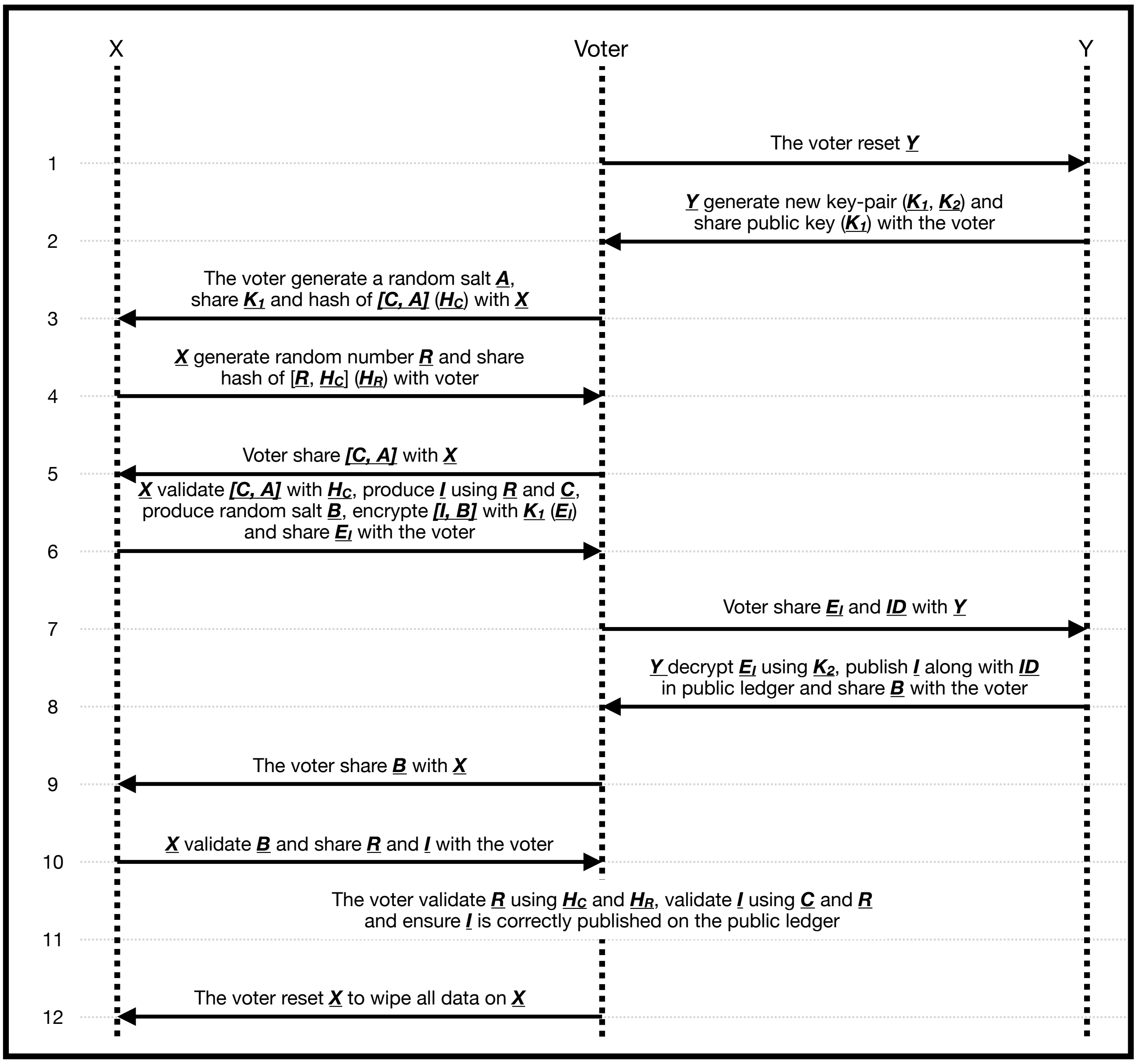

X
Voter
Y
1
The voter reset Y
2
Y generate new key-pair (K1, K2) and share public key (K1) with the voter
3
The voter generate a random salt A, share K1 and hash of [C, A] (HC) with X
4
X generate random number R and share hash of [R, HC] (HR) with voter
5
Voter share [C, A] with X
6
X validate [C, A] with HC, produce I using R and C, produce random salt B, encrypte [I, B] with K1 (EI) and share EI with the voter
7
Voter share EI and ID with Y
8
Y decrypt EI using K2, publish I along with ID in public ledger and share B with the voter
9
The voter share B with X
10
X validate B and share R and I with the voter
11
The voter validate R using HC and HR, validate I using C and R and ensure I is correctly published on the public ledger
12
The voter reset X to wipe all data on X

Figure 2: The steps in digital implementation of voting system

8 References

1. Jafar, Uzma, Mohd Juzaiddin Ab Aziz, and Zarina Shukur. ”Blockchain for electronic voting system—review and open research challenges.” Sensors 21.17 (2021): 5874.

2. Hajian Berenjestanaki, Mohammad, et al. ”Blockchain-based e-voting systems: a technology review.” Electronics 13.1 (2023): 17.

3. Adekunle, Salako E. ”A Review of Electronic Voting Systems: Strategy for a Novel.” International Journal of Information Engineering & Electronic Business 12.1 (2020).

4. Huang, Jun, et al. ”The application of the blockchain technology in voting systems: A review.” ACM Computing Surveys (CSUR) 54.3 (2021): 1-28.

5. Gibson, J. Paul, et al. ”A review of e-voting: the past, present and future.” Annals of Telecommunications 71 (2016): 279-286.

6. Bederson, Benjamin B., et al. ”Electronic voting system usability issues.” Proceedings of the SIGCHI conference on Human factors in computing systems. 2003.

7. Muyambo, Edmore, and Stacey Baror. ”Systematic Review to Propose a Blockchain-based Digital Forensic Ready Internet Voting System.” International Conference on Cyber Warfare and Security. Vol. 19. No. 1. 2024.

8. Kho, Yun-Xing, Swee-Huay Heng, and Ji-Jian Chin. ”A review of cryptographic electronic voting.” Symmetry 14.5 (2022): 858.

9. Ologunde, Ezekiel. ”Cryptographic Protocols for Electronic Voting System.” Available at SSRN 4823470 (2023).

10. Alharbi, Ayman, Haneen Zamzami, and Eman Samkri. ”Survey on homomorphic encryption and address of new trend.” International Journal of Advanced Computer Science and Applications 11.7 (2020).

11. Alloghani, Mohamed, et al. ”A systematic review on the status and progress of homomorphic encryption technologies.” Journal of Information Security and Applications 48 (2019): 102362.