Enhanced Tally Scheme for the “DEMOS” End-2-End Verifiable E-voting

Thomas Souliotis

Table of Contents

Background: Public Key Cryptography

- **Key generation**:
  $(pk, sk) \leftarrow Gen(1^\lambda)$

  $x \overset{r}{\leftarrow} \mathbb{Z}_q$

  $h = g^x$

  $pk = ((p, q, g), h)$

  $sk = x$

- **Encryption**:
  $m \rightarrow M \in G$

  $r \overset{r}{\leftarrow} \mathbb{Z}_q$

  $c = Enc(pk, M) = (c_1, c_2) = (g^r, h^r M)$

- **Decryption**:
  $Dec(sk, c) = \dfrac{c_2}{c_1^{sk}} = \dfrac{h^r M}{(g^r)^x} = M$

Background: Zero Knowledge Proofs

For voting:

1. 3-step (Σ-protocols)
2. Non-interactive
3. Completeness, Soundness, Zero Knowledge
4. OR-Proofs, Shuffle proofs
5. Example: Schnorr

Background: Zero Knowledge Proofs

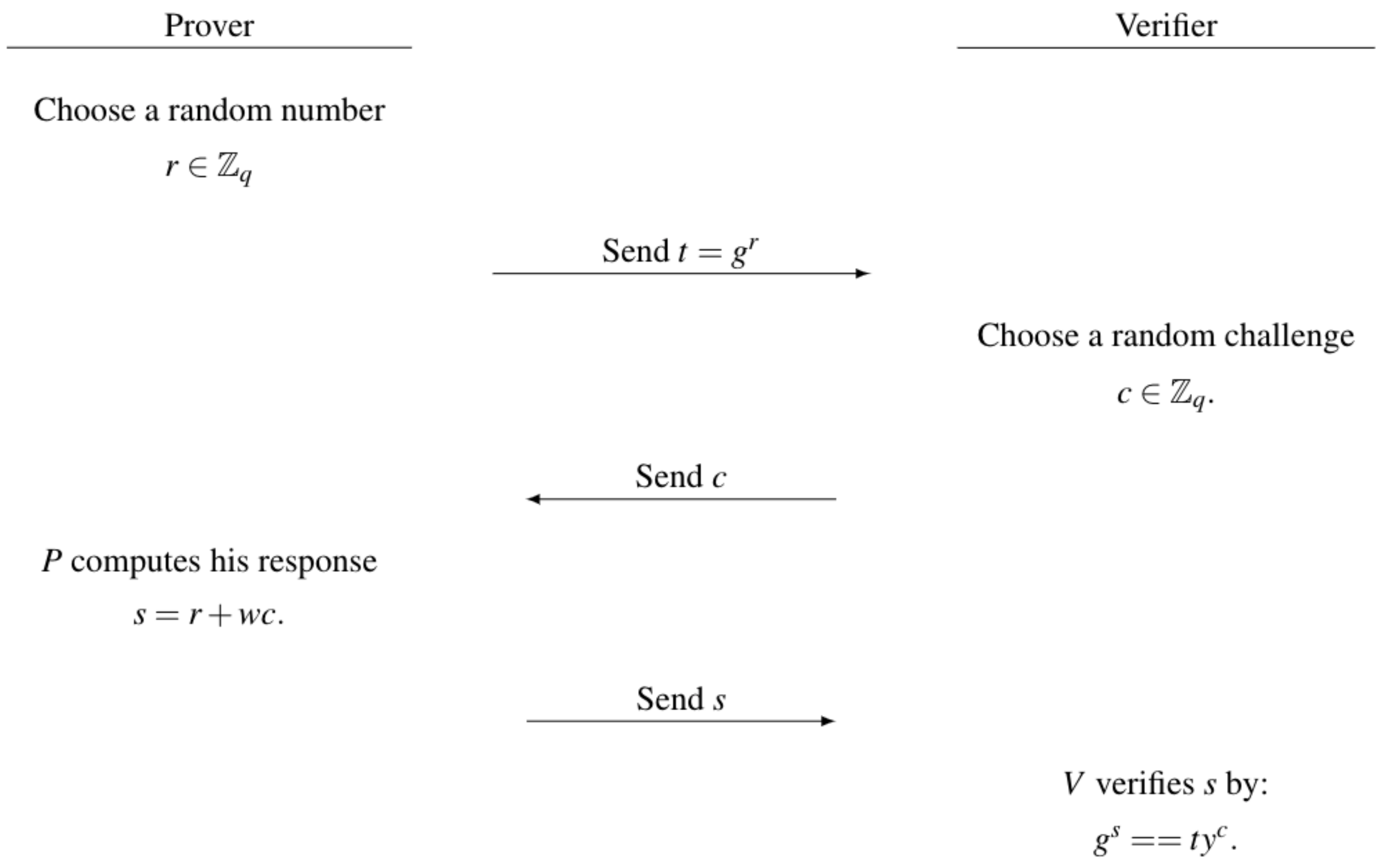


Figure 2.1: Schnorr Protocol

Background: Homomorphic Encryption

- Homomorphic Encryption with operation (·) :
  $c_1 \cdot c_2 = Enc(pk, M_1) \cdot Enc(pk, M_2) = Enc(pk, M_1 \cdot M_2),$

  $c_1 = Enc(pk, M_1), c_2 = Enc(pk, M_2)$

DEMOS: Introduction

Why DEMOS?

DEMOS: Introduction

1. E2E verifiable system in the standard model.

2. Does not depend on Random Oracles.

3. DEMOS is also a receipt free system, enhancing the privacy and its coercion resistance.

4. DEMOS is a very practical system, since the users/voters do not require to perform any complex operations, they just select their choices, and all the proofs, are handled by the EA.

5. DEMOS utilizes a technique by which the voters contribute ‘random bits’, in order to create a random challenge, for a sound ZKP.

DEMOS: Notation

We will talk about DEMOS-1 (referred as DEMOS for simplicity from now on).

- Commitment scheme is lifted ElGamal over elliptic curves: $Com_{ck}(m;r)\ (=(g^r, g^m h^r))$ which is additively homomorphic under multiplication: $c_1 \cdot c_2 = Com_{ck}(m_1;r_1) \cdot Com_{ck}(m_2;r_2) = Com_{ck}(m_1+m_2;r_1+r_2)$
- n voters denoted by $\mathbb{V} = \{V_1, \ldots, V_n\}$, m candidates denoted by $\mathbb{P} = \{P_1, \ldots, P_m\}$, a security parameter λ, and $m, n = poly(\lambda)$
- five main algorithms: **Setup(), Cast(), Tally(), Result(), Verify()**
- The **E2E Verifiability** is proven through an E2E Verifiability game that, as well as the **Voter Privacy** is proven through a voter privacy game, which is based on the receipt-freeness of the system.

DEMOS: Phases

1. **Setup()**
2. **Cast()**
3. **Tally()**
4. **Result()**
5. **Verify()**

DEMOS: Setup

- EA produces n double ballots, and assigns a (unique) tag to each of them.
- EA produces $2mn$ unique vote-codes, $2n$ unique permutations, $2mn$ random numbers for commitments.
- For candidate P_j the value that represents him is $(n+1)^{j-1}$ (value to be committed).
- EA permutes and encrypts each ballot side with the unique permutations (vote codes are also permuted with the same permutation).
- EA commits to the first phase of the Σ-protocol.
- The tags of each ballot, the committed vote codes of each ballot, alongside with the committed values and the commitments for the ZKP are all made public, by posting them to the BB

DEMOS: Cast

- V receives the decommitted personal ballot.
- V chooses one of the two sides, by which he will vote.
- He then finds the vote code that corresponds to the preferred candidate, and casts his vote, which consists of his **tag, his choice of the side of the ballot, and the vote code**.
- V keeps the not selected part of the ballot, as a receipt, which can be used to ensure that the values in this part (which are opened later) are what they should be.

DEMOS: Tally

- EA sends to the BB for each voter, the vote code chosen, alongside with the decommitted side of the ballot **not** chosen from the voter with all the randomnesses that were used for the commitment.
- The commitments corresponding to the vote codes chosen are placed into a tally.
- Based on each bit contributed by each voter from the random selection of the side of the ballot they voted, the challenge of the ZKP is extracted (second step of Σ-protocol).
- Third step of Σ-protocol for the selected parts of the ballots is produced and sent to BB.
- The sum of the randomnesses of the homomorphicaly multiplied ciphertexts is given, alongside with the actual decommitted value of the homomorphicaly multiplied tally, so as anyone to check the correctness of decryption of the tally.

DEMOS: Result

- Easily computable from the decommitted value then supposing that candidate P_j was chosen by x_j voters, then the total decommitted value will be equal to $\sum_{i=1}^{m} x_i(n+1)^{i-1}$. So, by repeatedly ‘modding' by $n+1$ and then dividing by the proper value, at the j-th repetition of the above we get $x_j = X \mod (n+1)$, and $X = \frac{X - x_j}{n+1}$

DEMOS: Verify

The verification process of the above data posted on the BB can take place from anyone.

DEMO: Σ - protocol

- 3-step ZKP.
- Proves that a value encrypted in a ballot, corresponds to a commitment to some value in $\{n+1\}_{i=0}^{m-1}$.
- Challenge, is extracted through the random bits that each V contributes when he chooses one of the two sides of the ballot (bits 0/1). Supposing that V are n,then we have n random bits.
- DEMOS uses a ZK amplification technique, where the whole challenge is segmented into k blocks, getting as a result k sub-challenges $\{a_i\}_{i=1}^{k}$
- Σ - protocol is run k times per commitment, and should produce k valid ZK-proofs for each commitment.
- Ensures that a commitment on the side of the ballot the voter chose to vote, belongs to a value in $\{(n+1)^i\}_{i=0}^{m-1}$.

Prover | **Verifier**

$\forall j \in \{0, \dots, \log m - 1\}$:

Define b_j: $i = \sum_{j=0}^{\log m - 1} b_j 2^j$

$t_j, z_j, y_j, r_j, w_j, f_j \xleftarrow{r} \mathbb{Z}_q$

$B_j = Com_{ck}(b_j; r_j)$

$T_j = Com_{ck}(t_j; z_j)$

$Y_j = Com_{ck}((1 - b_j)t_j; y_j)$

$W_j = Com_{ck}(w_j; f_j)$

Define A_j, a_j, r'_j :

$A_j = B_j^{N^{2^j} - 1} \cdot Com_{ck}(1; 0)$

$= Com_{ck}(a_j; r'_j)$

Define β_j, γ_j but for $j \in \{0, \dots, \log m\}$:

$\prod_{j=0}^{\log m - 1}(a_j X + w_j) = \sum_{j=0}^{\log m}(\beta_j X^j)$

$\prod_{j=0}^{\log m - 1}(r'_j X + f_j) = \sum_{j=0}^{\log m}(\gamma_j X^j)$

where $\gamma \log m = r$ (for efficiency)

$D_j = Com_{ck}(\beta_j; \gamma_j)$

$\phi_1 = \{B_j, T_j, Y_j, W_j, D_j\}_{j=0}^{\log m - 1}$

$\xrightarrow{\text{Send } \phi_1}$

challenge ρ produced

$\xleftarrow{\text{Send } \rho}$

P computes his response:

$\forall j \in \{0, \dots, \log m - 1\}$:

$t'_j = b_j \rho + t_j$, $z'_j = r_j \rho + z_j$

$y'_j = -y_j - r_j t'_j$, $w'_j = a_j \rho + w_j$

$f'_j = r'_j \rho + f_j$

$\phi_2 = \{t'_j, z'_j, y'_j, w'_j, f'_j\}_{j=0}^{\log m - 1}$

$\xrightarrow{\text{Send } \phi_2}$

V verifies by:

$E^{\rho^{\log m}} \prod D_j^{\rho^j} \stackrel{?}{=} Com_{ck}(\prod w'_j; \prod f'_j)$

$\forall j \in \{0, \dots, \log m - 1\}$:

$\frac{(Com_{ck}(1;0)/B_j)^{t'_j}}{Y_j} \stackrel{?}{=} Com_{ck}(0; y'_j)$

$A_j^{\rho} \cdot W_j \stackrel{?}{=} Com_{ck}(w'_j, f'_j)$

$B_j^{\rho} \cdot T_j \stackrel{?}{=} Com_{ck}(t'_j, z'_j)$

DEMOS: Security

If EA tries to cheat and guess for a specific voter the right ballot side, the probability of such an event is equal $\frac{1}{2}$. But even in this case, as it is proven in DEMOS, **the difference will be just one vote**, while the EA will be caught with probability $\frac{1}{2}$. Thus any significant variation of at least d votes will be caught with probability $1 - (\frac{1}{2})^d$.

DEMOS: Problems and Open Questions

A. **Efficiency problems**, because of all the heavy computations performed by a single EA. This is improved in the later publications of d-DEMOS and DEMOS-2, but it is not solved.

B. For a small number of voters, the system will not work correctly, as the challenges could be brute forced or guessed with non-negligible probability (min-entropy of the challenge).

C. **Limited max-number of voters** $n \cdot (n+1)^{m-1} \leq |M|$ e.g. $n = 10^6, m = 40 \Rightarrow n * (n+1)^{m-1} \approx 10^{240} = (10^3)^{80} \approx 2^{800}$

D. **Only approval voting** elections are supported from the system.

New Enhanced Demos Protocol: Modifications

1. Instead of just one tally we have m different tallies, one for each candidate. Furthermore, in the new system when a voter makes a choice, he does not choose the candidate, but he chooses the value (ranking) this candidate will receive.

2. V will not just cast one vote code but m.

3. **Two new ZKP approaches**, where in the first one, we use the same Σ- protocol used in DEMOS, but by providing a completely new proof concept, while in the second approach we use a ZKP of a shuffle.

4. The values encrypted are also changed($\{0,\ldots,m-1\}$ or $\{x^0,\ldots,x^{m-1}\}$ (l will be defined later)).

New Enhanced Demos Protocol: The ZKP Approaches

1. Shuffle Proof
2. Transforming the Current Σ - Protocol

New Enhanced Demos Protocol: Shuffle Proof

- Commitment to $Com_{ck}(j-1; r_j)$ (for simple borda case) instead of $Com_{ck}((n+1)^{j-1}; r_j)$. A 3-step shuffle proof is then provided.

- **Solves the problem with the maximum values of** m, n, since the current system can support elections with $n, m : n(m-1) < q$.

- **Problems with the challenge space**. Current 3-step shuffle proofs require at least m random challenges.

New Enhanced Demos Protocol: Transforming the Current Σ- Protocol

- We transform the current ZKP protocol into a working shuffle proof.

- We use the same protocol we only add some additional constraints.

- Helps with the security analysis.

New Enhanced Demos Protocol: Transforming the Current Σ- Protocol

- It is proven that given the current Σ - protocol (proves that each ciphertext is of the format x^i) and the 2 following values:

$\left(\prod_{i=0}^{m-1} c_i = Com_{ck}\left(\sum_{i=0}^{m-1} x^i ; \sum_{i=0}^{m-1} r_i\right), \sum_{i=0}^{m-1} r_i\right)$ **we have a valid shuffle proof** of the set: $\{x^0, \ldots, x^{m-1}\}$

- For the proof we start from the following (a_i represents how many times the value x^i may be chosen):

$$a_i \in \mathbb{Z}^{\geq}, \ (1)$$

$$\sum_{i=0}^{m-1} a_i = m, \ (2)$$

$$x \in \mathbb{Z}^{\geq}, \ (3)$$

$$\sum_{i=0}^{m-1} a_i x^i = \sum_{i=0}^{m} x^i, \ (4)$$

$$(1), (2), (3), (4) \Rightarrow a_i = 1, \ \forall i \in \{0,...,m-1\}$$

And we prove that this holds iff: $x \geq 2(m-1)+1$

New Enhanced Demos Protocol: Description of the New System

We present the 5 protocols (**Setup(), Cast(), Tally(), Result(), Verify()**) again, in more details under the new system.

New Enhanced Demos Protocol: Setup

1. $ck \leftarrow Gen(Param, 1^{\lambda})$

2. **EA** selects permutations $\pi_l^{(0)}, \pi_l^{(1)}$ over $\{1,...,m\}$, so as to shuffle the order of the vote-code and the choices, in the two parts of the ballot, following partially the existing protocol. The difference now is the new ZKP, and the form of the ballots.

3. **EA** selects unique vote-codes $C_{l,j}^{(0)} \leftarrow \mathbb{Z}_q$ (resp. $C_{l,j}^{(1)}$) with $j \in \{1,...,m\}$. The m different codes are associated with the m different possible position/rank each of the candidate might get. As, it is shown in the DEMOS system the values are not necessarily randomly chosen from $\mathbb{Z}_q$, but they might belong to a (much) smaller subset of it, so as to be more user friendly. **More precisely, in our practical implementation and in the example section we provide a way of getting meaningful vote codes in a systematic way.**

4. **EA** generates the ballot s_l consisting of two parts $s_l^{(0)}, s_l^{(1)}$ with each part consisting of: $s_l^{(a)} = \{(P_j, C_{l,j}^{(a)})\}, a \in \{0,1\}$, and $s_l = (tag_l, s_l^{(0)}, s_l^{(0)})$

New Enhanced Demos Protocol: Setup

4. **EA** computes $j' = \pi_l^{(a)}(j), \forall j \in \{1,...,m\}$, as the new indexes of the ciphertexts.

5. **EA** chooses randomnesses $t_{l,j'}^{(a)}$. These randomnesses will be used to commit in permuted form, to the vote-codes we have previously generated as: $U_{l,j'}^{(a)} = Com_{ck}(C_{l,j'}^{(a)}; t_{l,j'}^{(a)})$

6. **EA** chooses randomnesses $r_{l,j'}^{(a)}$, that will be used to actually encode the position/rank commitment ($\{x^0, \ldots, x^{m-1}\}$) . The commitments now will be: $E_{l,j'}^{(a)} = Com_{ck}(x^{j'-1}; r_{l,j'}^{(a)})$

7. **EA** prepares $\phi_{1,l,j'}^{(a)}$ for the first step of the ZKP

8. **EA** publishes the public information (very similar to the current DEMOS protocol):
   $Pub = (\mathbb{P}, \mathbb{U}, \{Pub_l\}_{l \in \{1,...,n\}})$, with
   $Pub_l = (tag_l, \{(U_{l,j'}^{(a)}, E_{l,j'}^{(a)}, \phi_{1,l,j'}^{(a)})\}_{j' \in \{1,...,m\}}^{a \in 0,1})$.
   The secret key of EA will be:
   $msk = \{Pub_l, s_l, msk_l, state_{\phi,l}\}_{l \in [n]}$ with
   $msk_l = \{(C_{l,j}^{(a)}, t_{l,j}^{(a)}, \pi_l^{(a)}(j) = j')\}_{j \in \{1,...,m\}}^{a \in 0,1}$

New Enhanced Demos Protocol: Cast

- Cast protocol is pretty similar, Only now, instead of just one vote-code, V_l sends m.
- V_l flips the coin and chooses the $a_l \leftarrow \{0,1\}$, and selects the $s_l^{a_l}$ to vote and the $s_l^{1-a_l}$ for audit.
- Suppose now that V_l has an order of preference ω_l over the m candidates, meaning that he considers the candidate P_1 as his $\omega_l(1)$ favorite choice, P_2 as his $\omega_l(2)$ favorite choice…
- V_l arranges all vote codes in an order of preference and then casts them as: $\psi_l = (tag_l, a_l, \{C_{l,j''}^{(a_l)}\}_{j''=1}^{m})$,
  where $\omega_l(j) = j''$

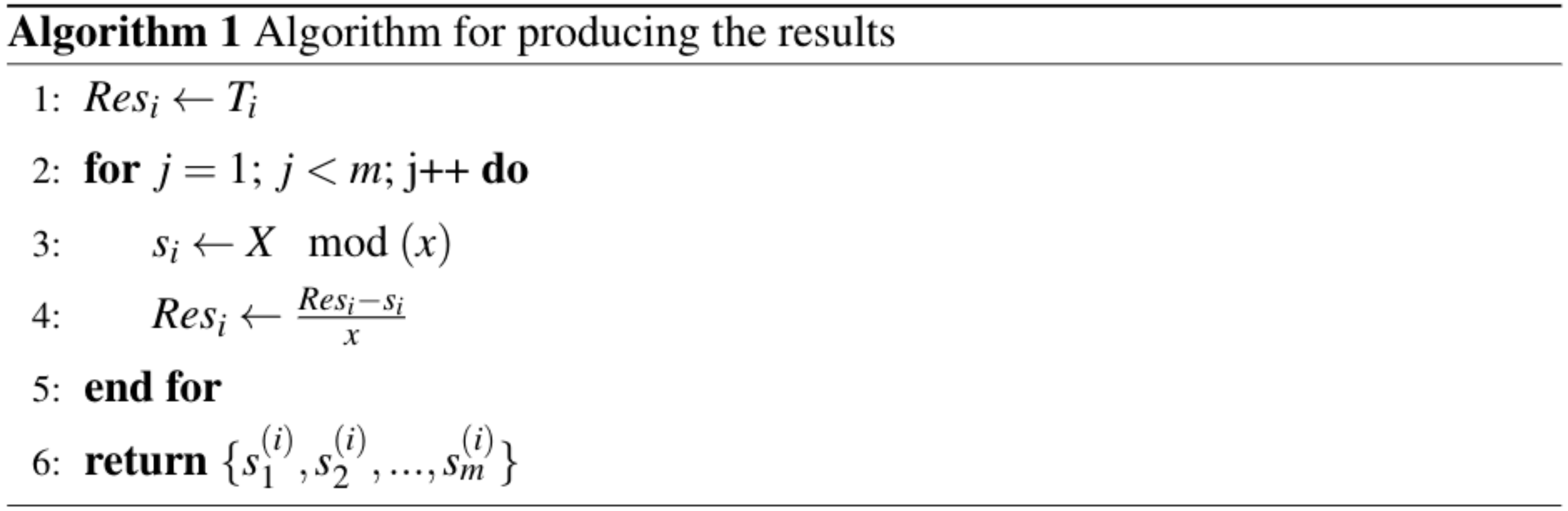
New Enhanced Demos Protocol: Tally

Similar but with m different tallies.

1. EA uses (tag_l, a_l) and finds $s_l^{1-a_l}$, sending it alongside with ψ_l to the BB for each voter in $\bar{\mathbb{V}}$ ($\bar{\mathbb{V}} \subseteq \mathbb{V}$).
2. The BB is also updated by opening all the vote-code commitments, and sending all the pairs $\{(C_{l,j}^{(a)}, t_{l,j}^{(a)})\}$ to the BB.
3. For each vote ψ_l, the **EA** does the following:
   i. For each of the m vote codes $C_{l,j}$, **EA** finds the cast vote-code that matches the $C_{l,j''}^{(a_l)}$, and finds and adds the corresponding commitment $E_{l,\pi_l^{(a_l)}(j'')}^{(a_l)}$ to the $\mathbf{E}_{tally}^{i}$ set. There are m - tally sets and each $\mathbf{E}_{tally}^{i}$ set corresponds to the values that the i-th candidate will get.

   ii. **EA** places all the $\{E_{l,j}^{(1-a_l)}\}_{j\in\{1,\ldots,m\}}$ to the $\mathbf{E}_{open}$ for the audit part.
4. Verifier's challenge for the ZKP is produced from the random bits, and then the third step of the protocol. Also, the sum of randomnesses are provided as $\{Q_i\}_{i=1}^{n}$, with $Q_i = \sum_{j=1}^{m} r_j^{(a_l)}$
5. **EA** combines $\mathbf{E}_{sum}^{i} = \prod_{E\in\mathbf{E}_{tally}^{i}} E$, and produces m results T_j with total randomness R_j with $j \in \{1,\ldots,m\}$
6. **EA** sends to the BB the previous results alongside with $\mathbf{E}_{open}$ and all the decommitted information of the not-used parts.

New Enhanced Demos Protocol: Result



Algorithm 1 Algorithm for producing the results

```
1: $Res_i \leftarrow T_i$
2: for $j = 1$; $j < m$; j++ do
3:     $s_i \leftarrow X \mod (x)$
4:     $Res_i \leftarrow \frac{Res_i - s_i}{x}$
5: end for
6: return $\{s_1^{(i)}, s_2^{(i)}, \ldots, s_m^{(i)}\}$
```

An $s_j^{(i)}$ contains the value of how many times the candidate P_i was voted as the j-th choice in total.

New Enhanced Demos Protocol: Verify

1. n distinct ballots, n distinct tags, $2nm$ distinct vote codes

2. All the not selected parts of the ballots are opened. No selected part is opened (each cast vote code should not be opened)

3. All the Σ - protocols are valid.

4. The homomorphic combination of the commitments of each selected ballot is a commitment to $1 + x + \ldots x^{m-1}$

5. All the openings of the not selected commitments are correct.

6. The final $\mathbf{E}^{i}_{sum}$ of each tally is indeed the product of everything it contains.

7. Each vote ψ_l contains m exactly vote codes, the one after the other, representing the order of preference for each of the candidates.

8. All the vote codes that are cast in a single vote ψ_l, are part of the ballot with $tag = tag_l$.

New Enhanced Demos Protocol: Correctness, Security and Verifiability

- **Correctness** & **Verifiability** proofs follow the same logic.

- For **Security** the following is defined:
  <u>Definition</u>:
  We consider that our system is secure under security parameters k, d, s.t. if a malicious EA 'alters' k ballots the result will not change iff
  $d = \min(|R(P_i) - R(P_j)|)_{i \neq j} > k(m-1)$.
  We state that for lower expected values of d the system will not be safe. We also claim that if **EA** tries to corrupt more than k ballots, then she will be caught with high probability $p > 1 - 2^{-k}$.

Limitations & Future Work

- Shuffle proofs need to be fixed.
- Issues with Security proofs.
- Σ - protocol still has efficiency issues.
- Practical implementation.

Thank you!