Preventing Board Flooding Attacks in Coercion-Resistant Electronic Voting Schemes

Reto E. Koenig, Rolf Haenni, Stephan Fischli

University of Fribourg
&
Bern University of Applied Sciences

07.06.2011

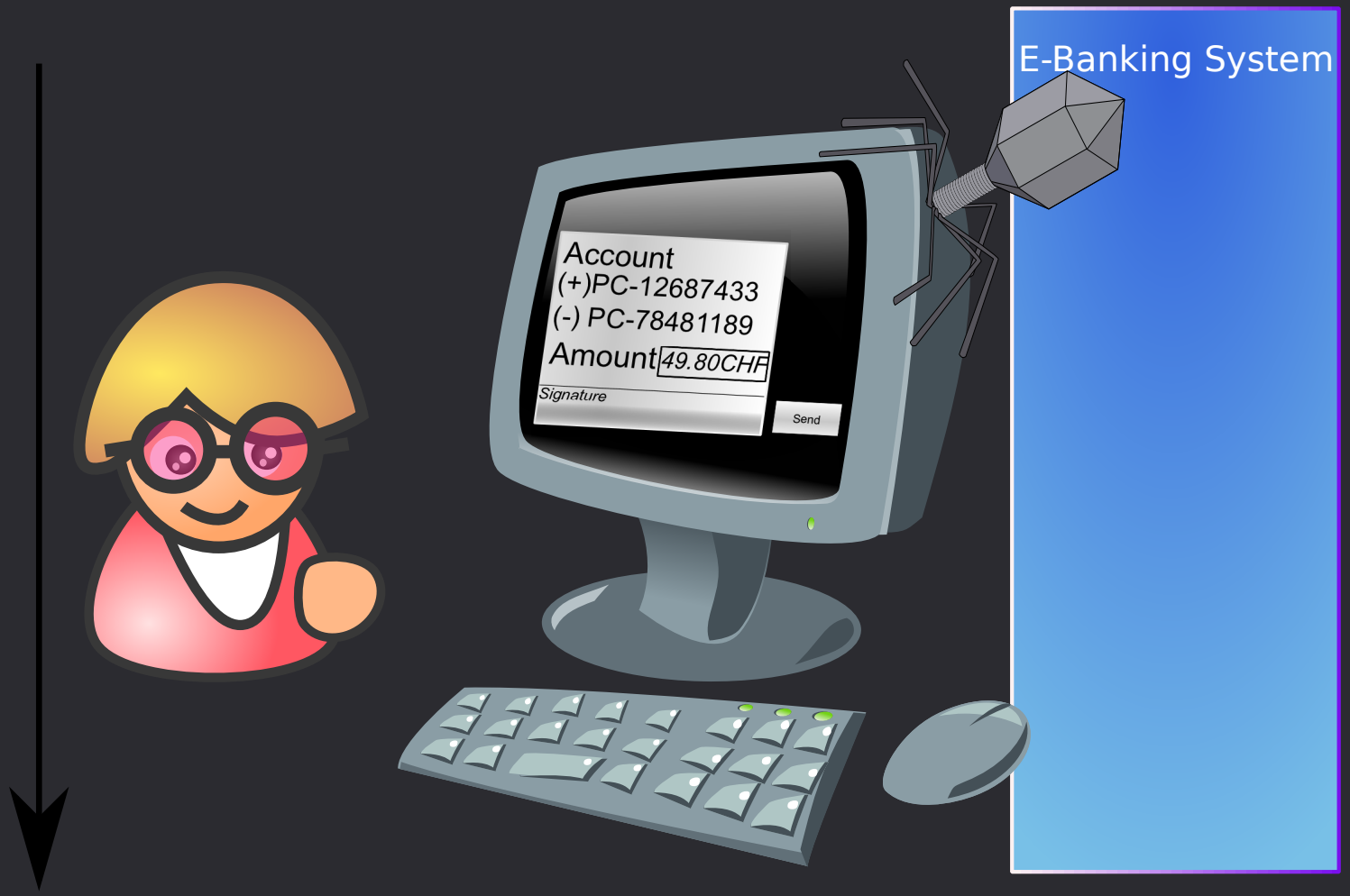
Question

Is e-voting like e-banking?

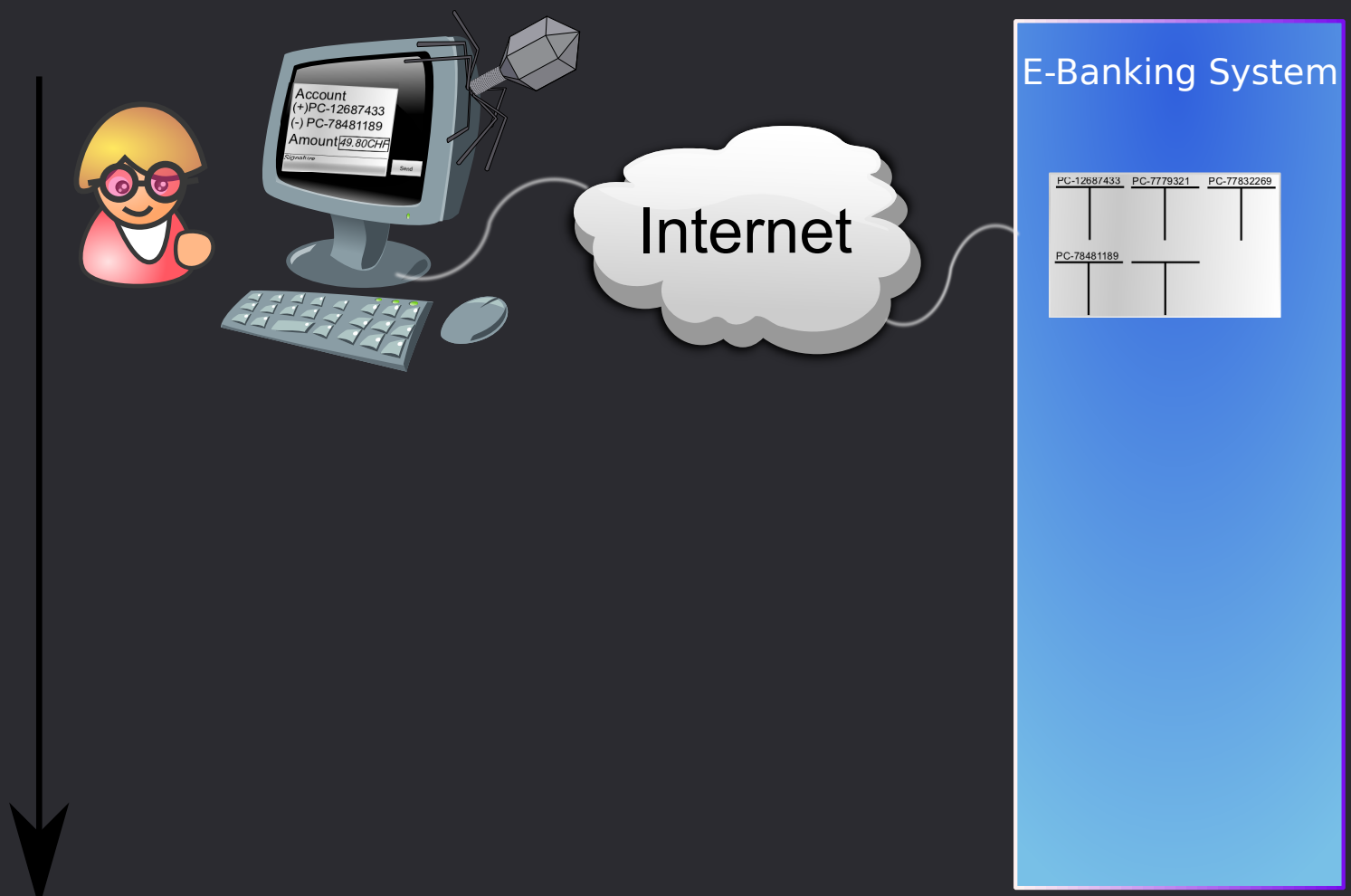
Alice wants to transfer money.

She wears the rose coloured glasses...

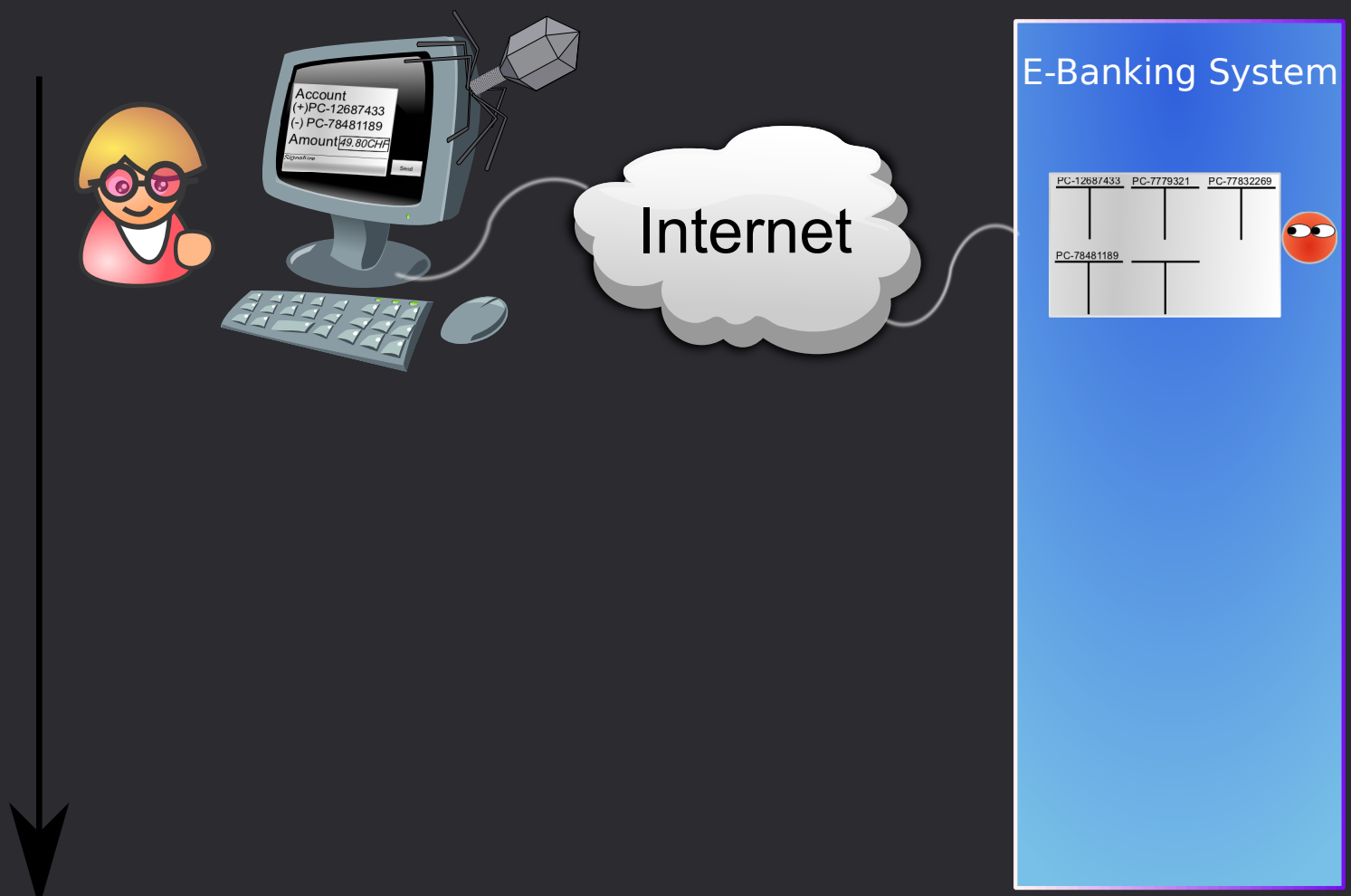
...due to the secure platform problem.



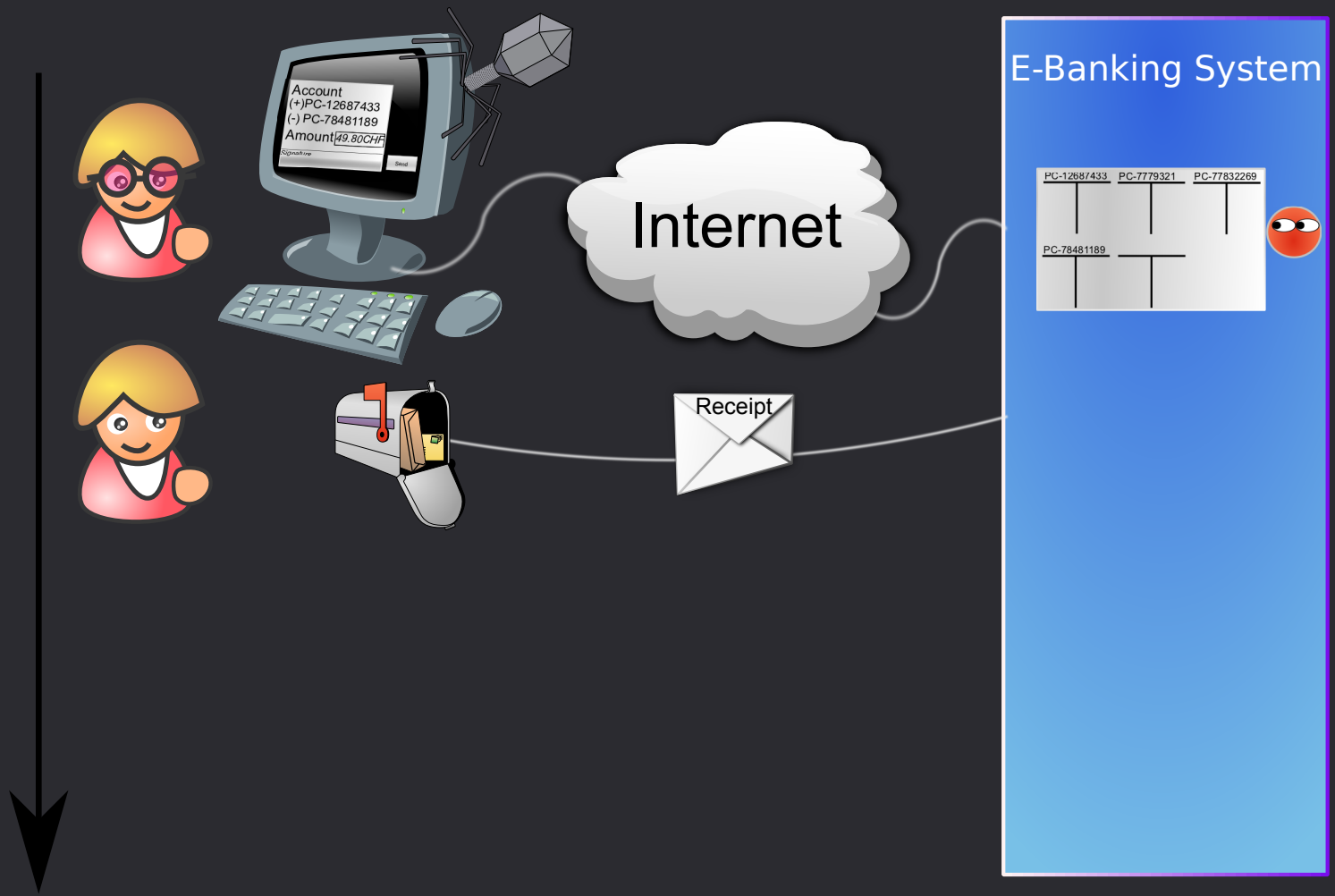
She transfers the transaction to the e-banking System...



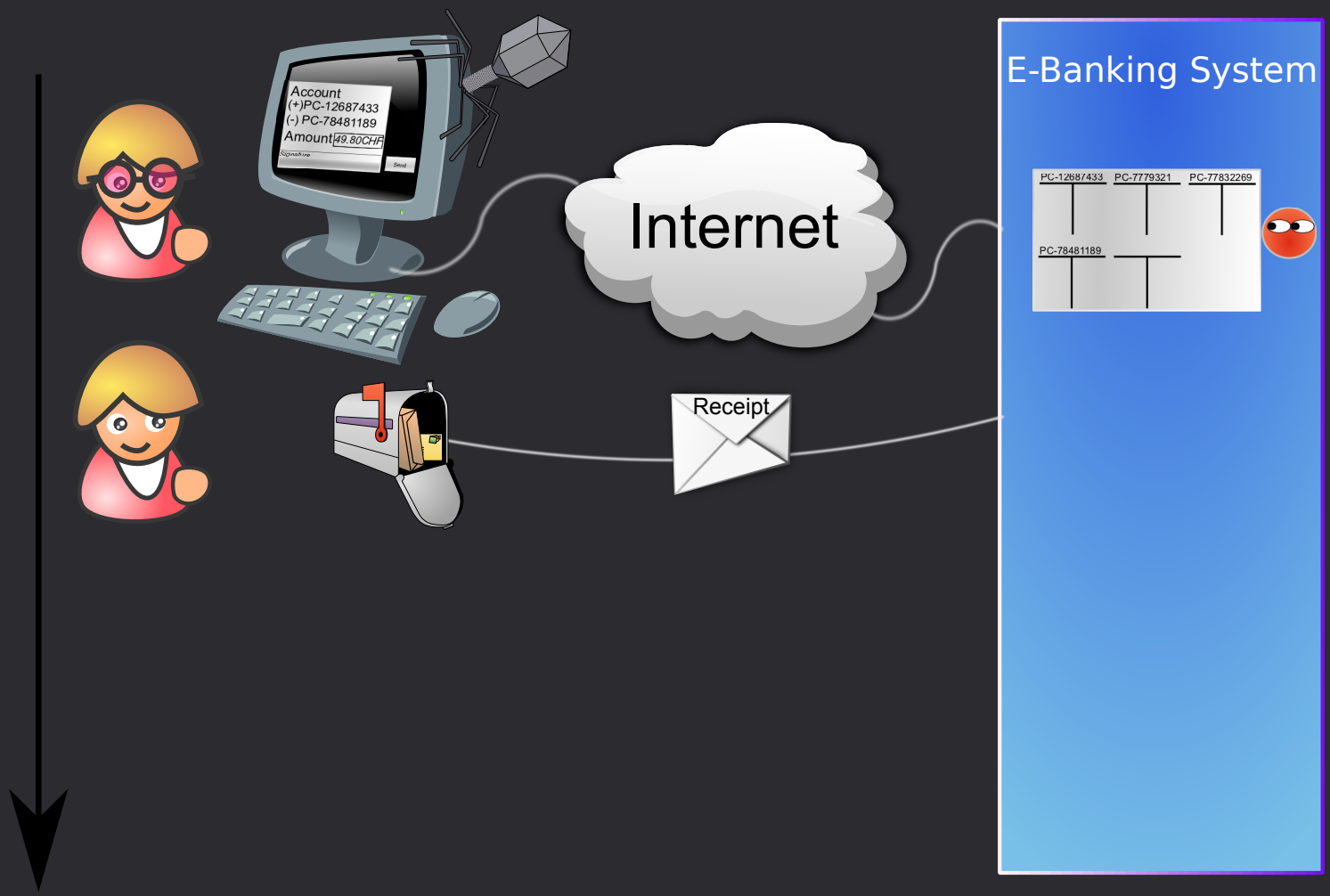
...which is fully observed by the bank.



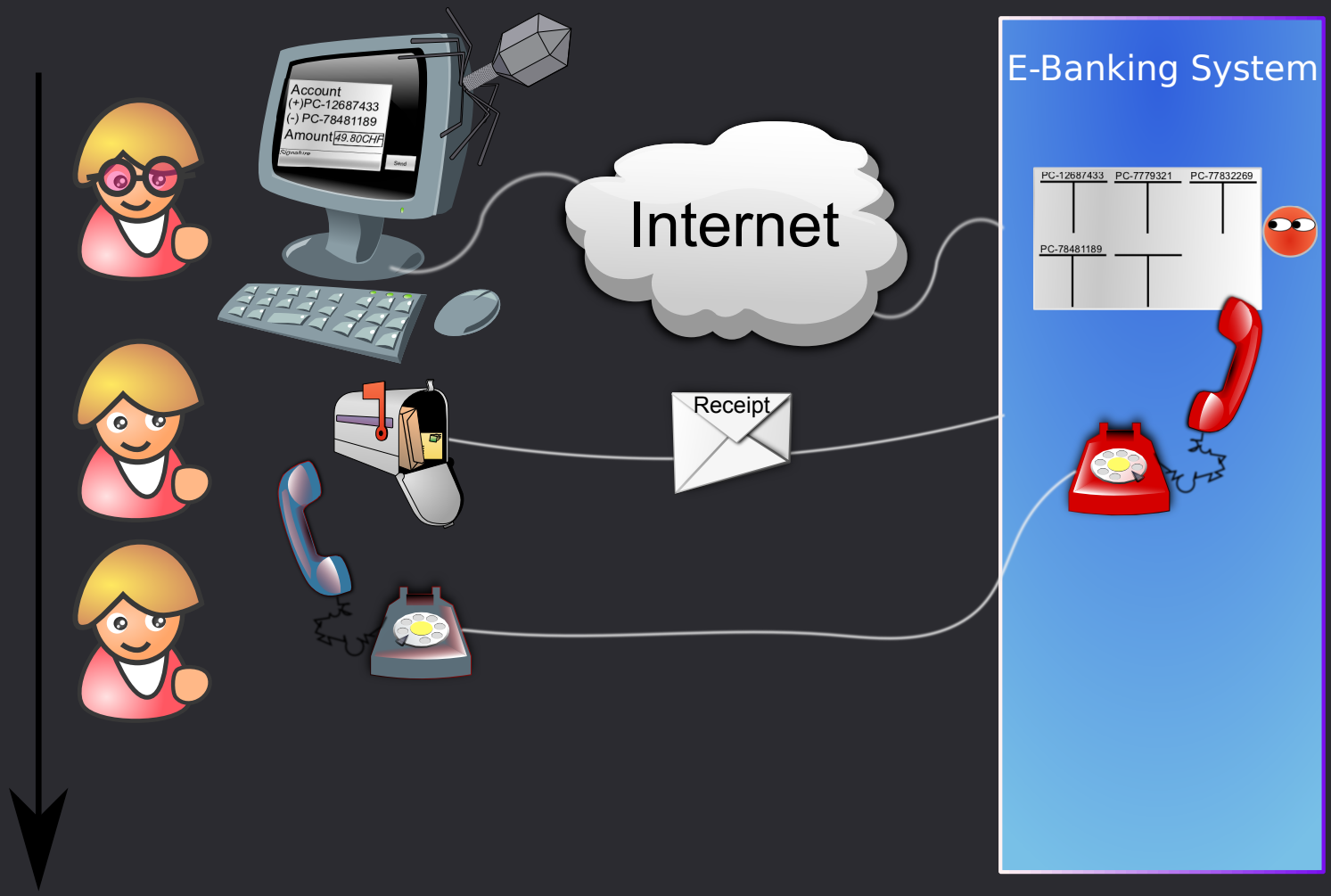
Alice gets a transaction receipt via an independent channel.



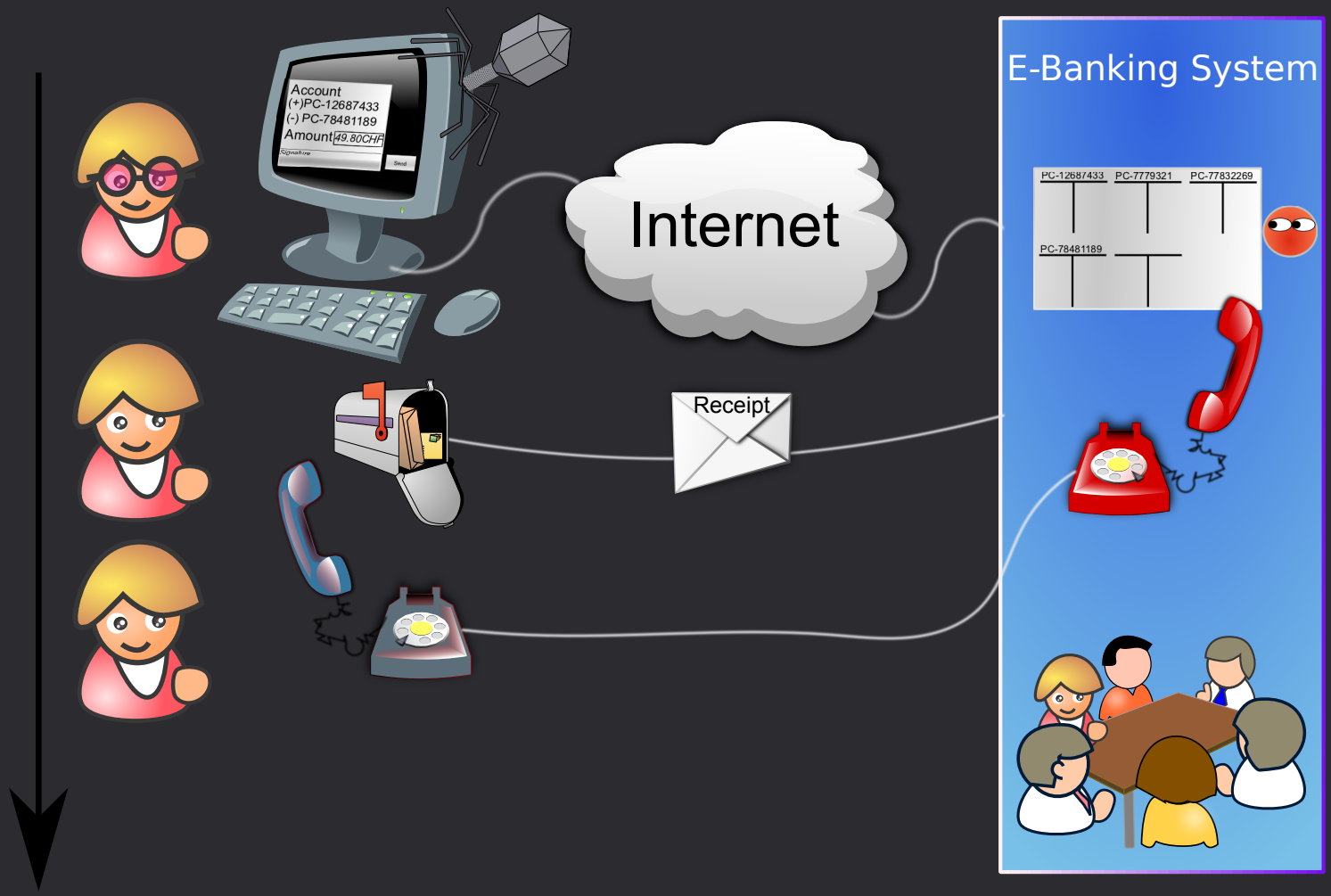
If either side has doubts about the transaction...



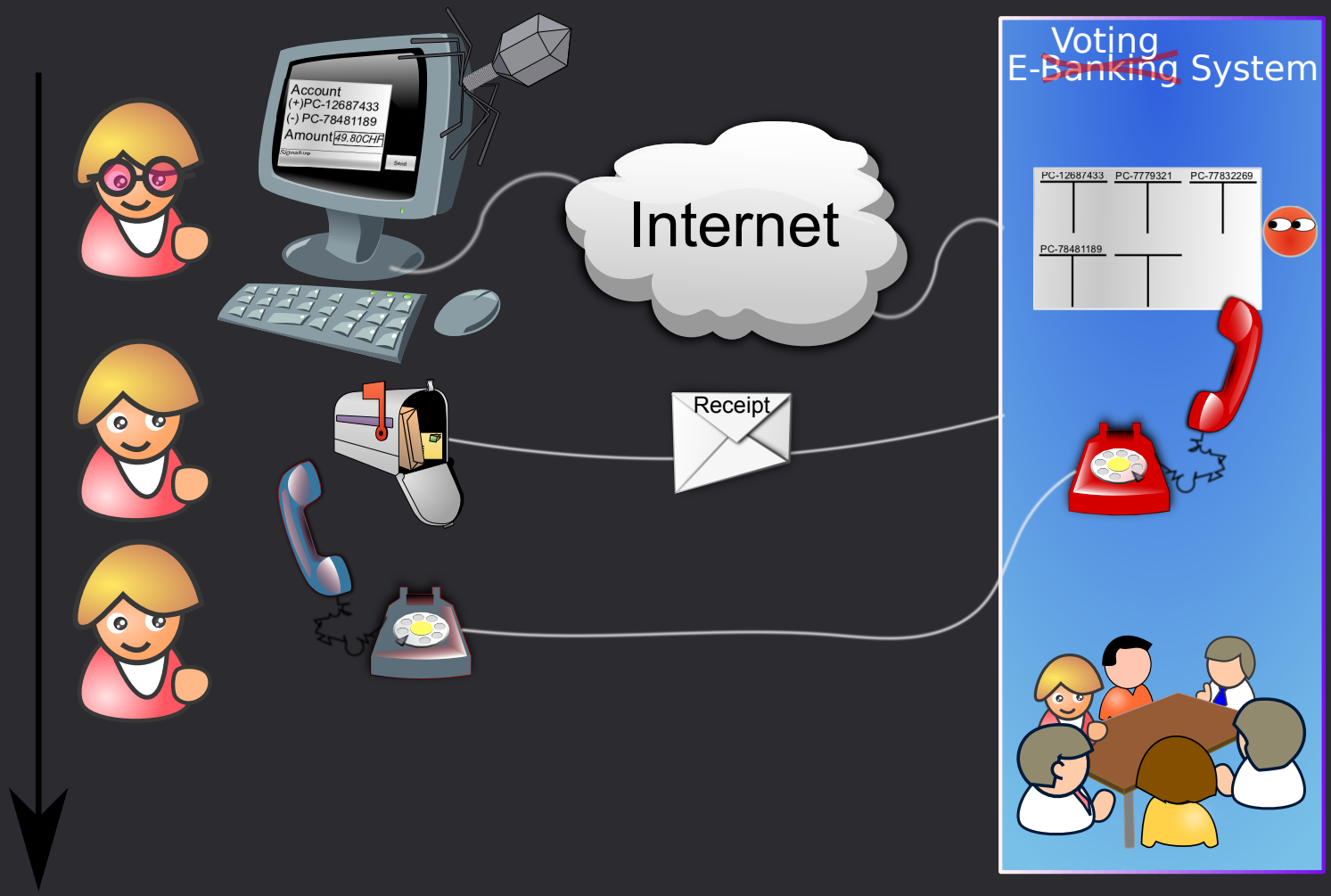
...the parties can get in touch...



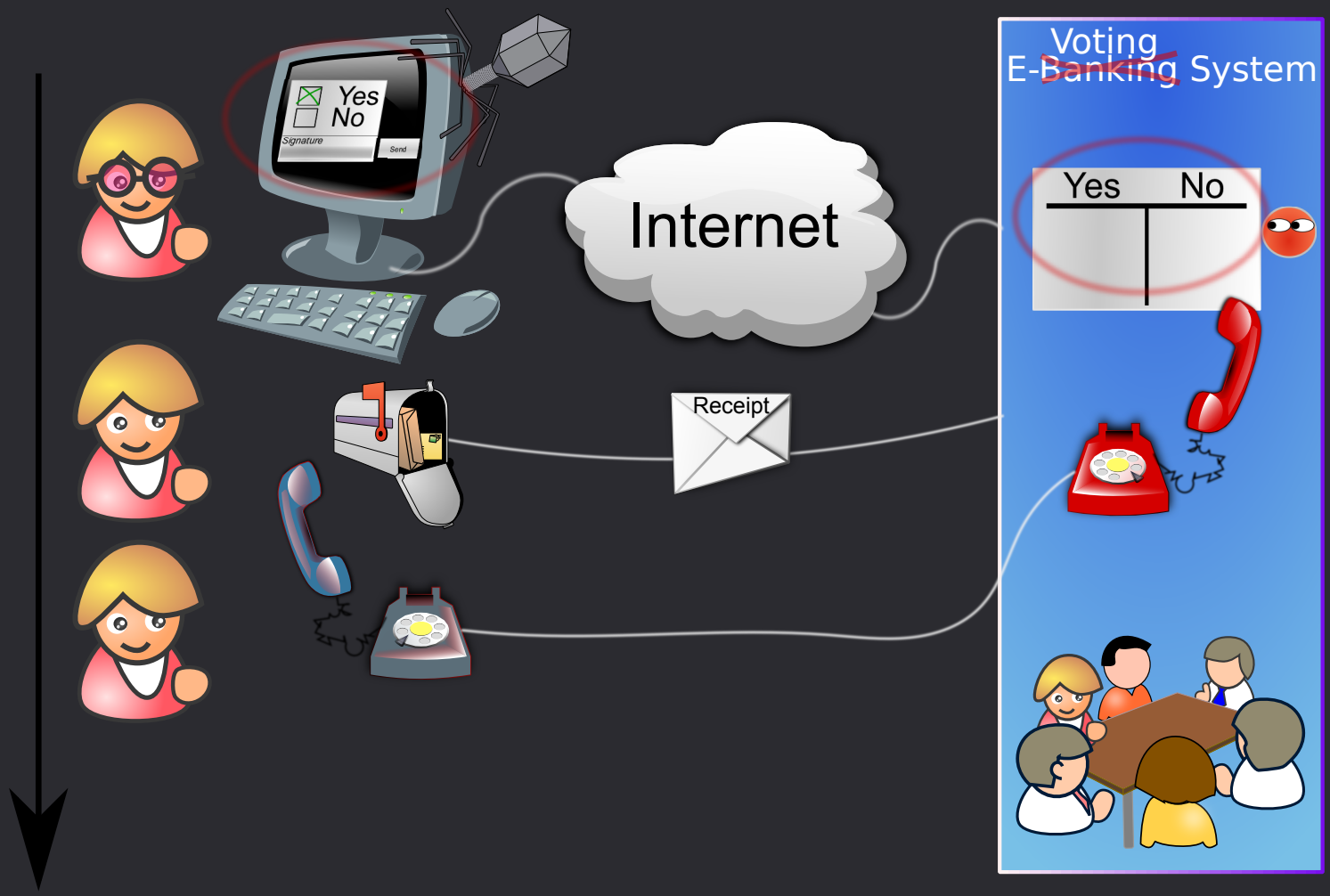
...the parties **can react**



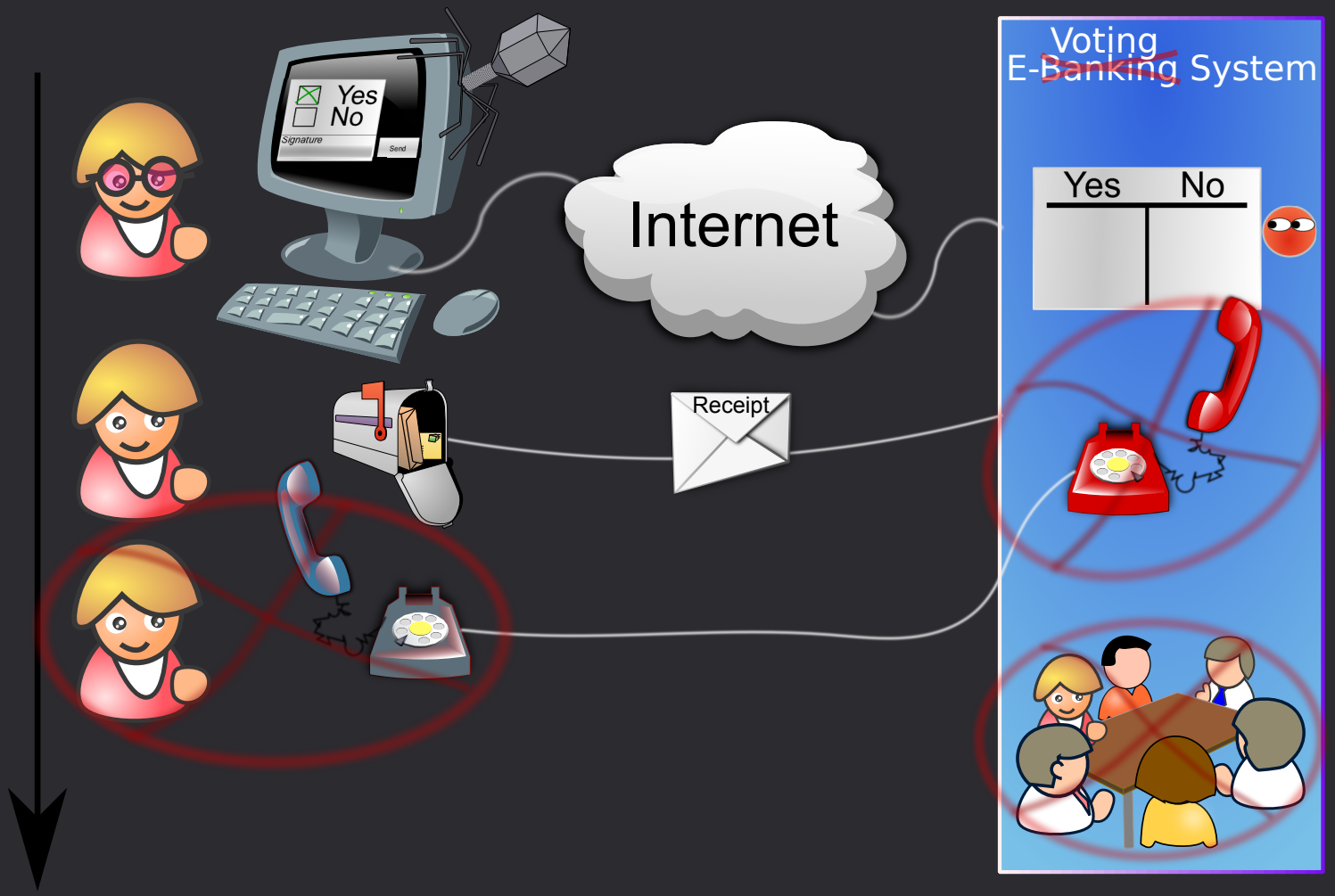
Start modifying the scheme by renaming it...



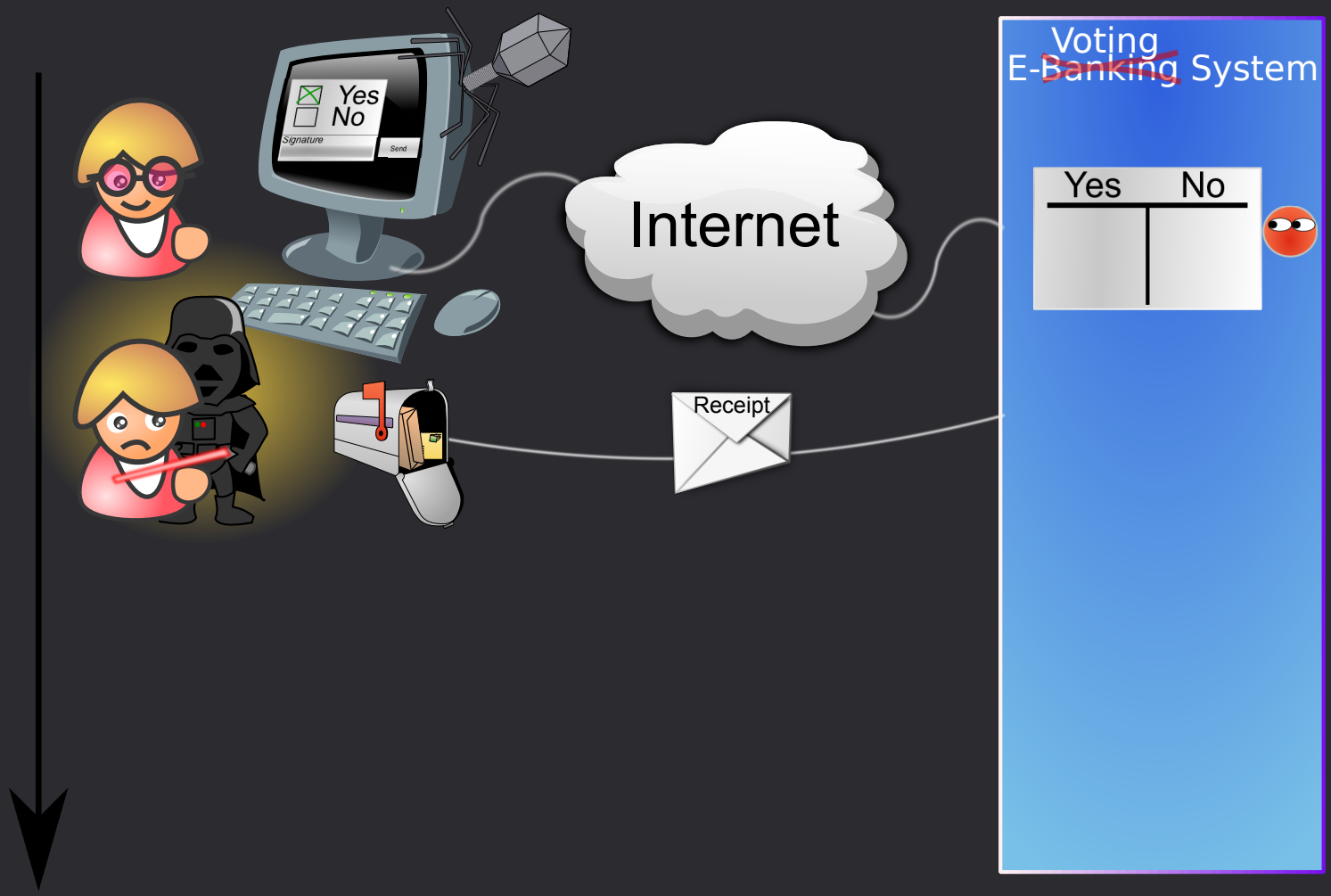
Account → ballot, reduce to one account for (Yes / No)



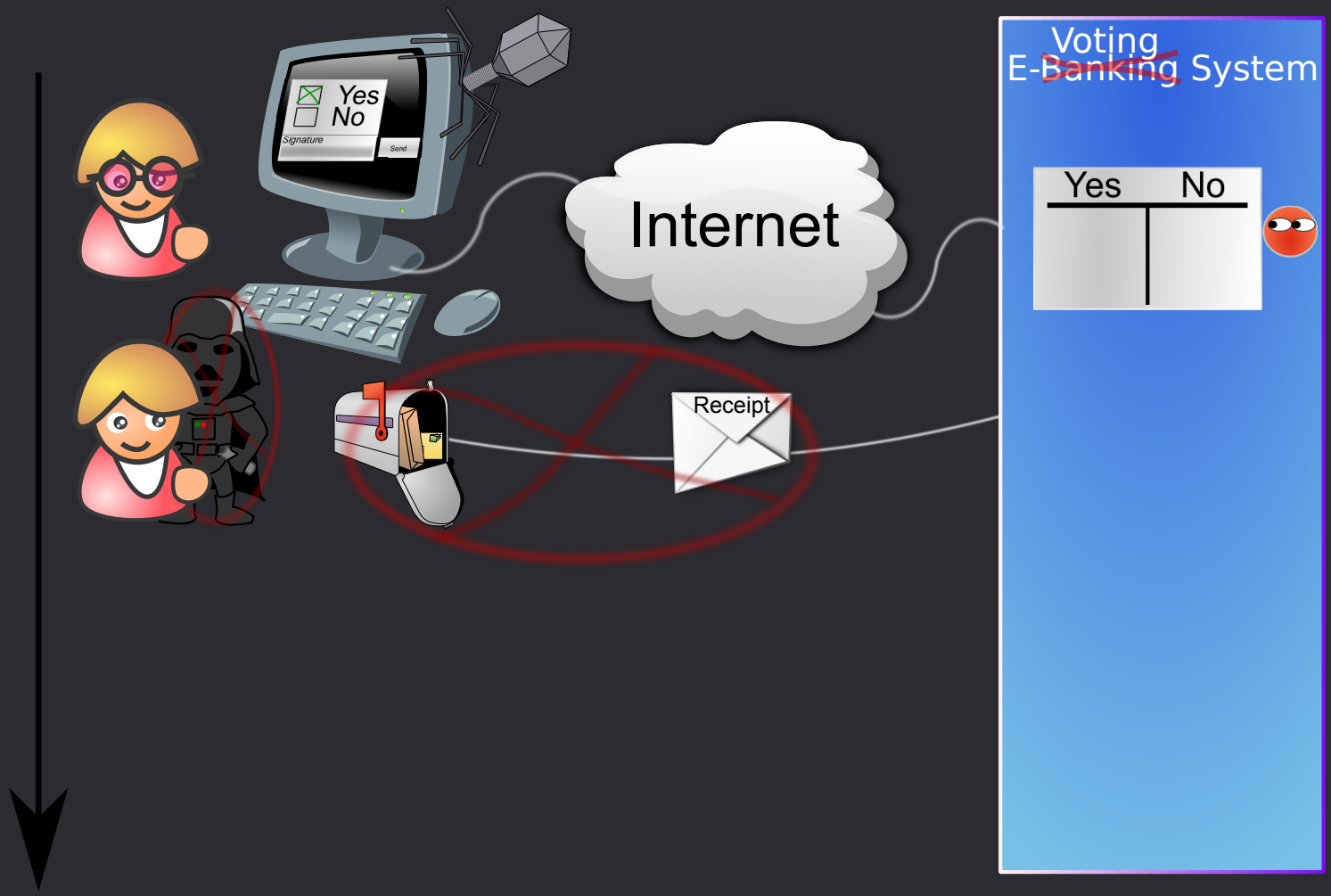
Disable the ability for voter interaction after the process



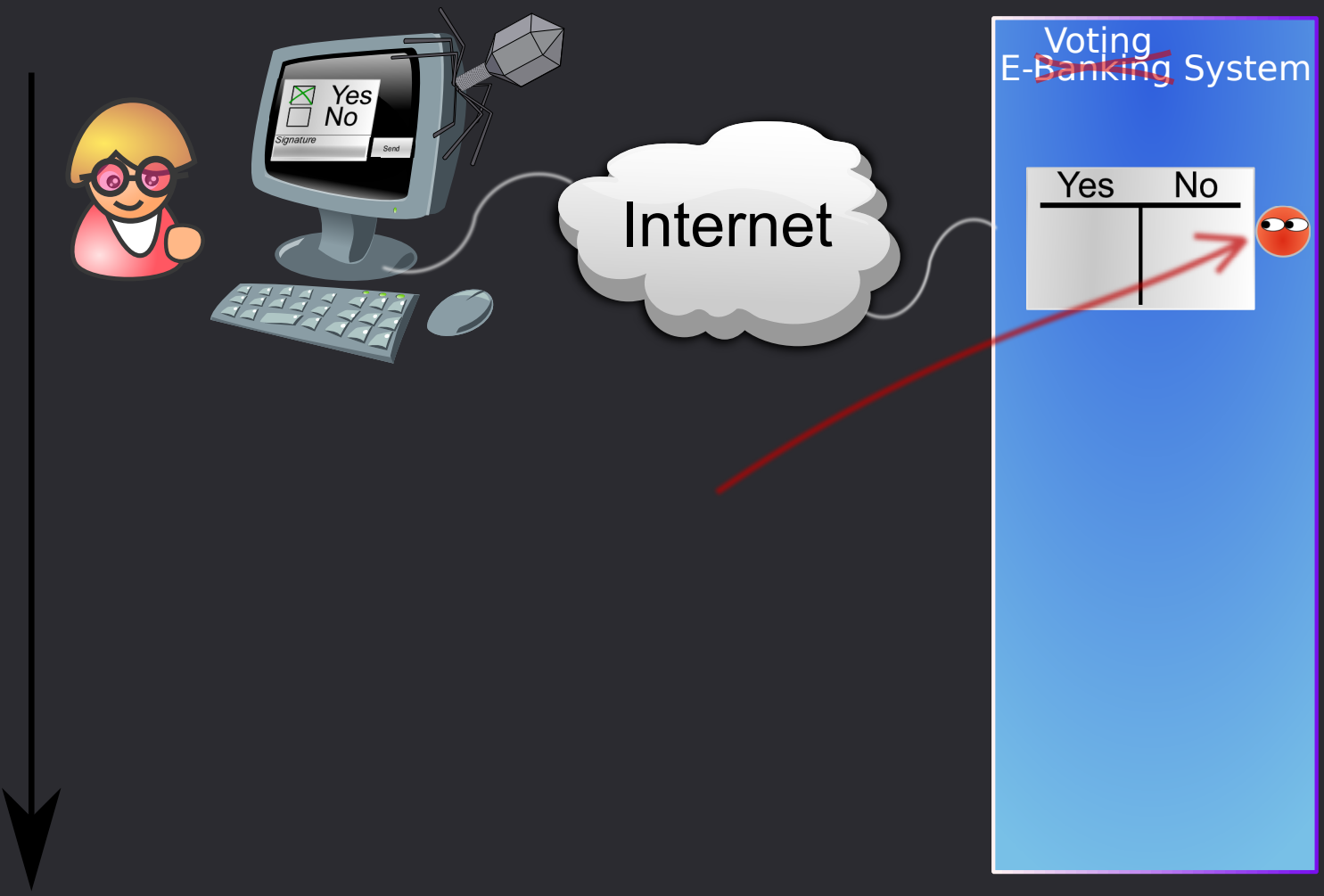
Disable the per-voter receipt of the counted vote...



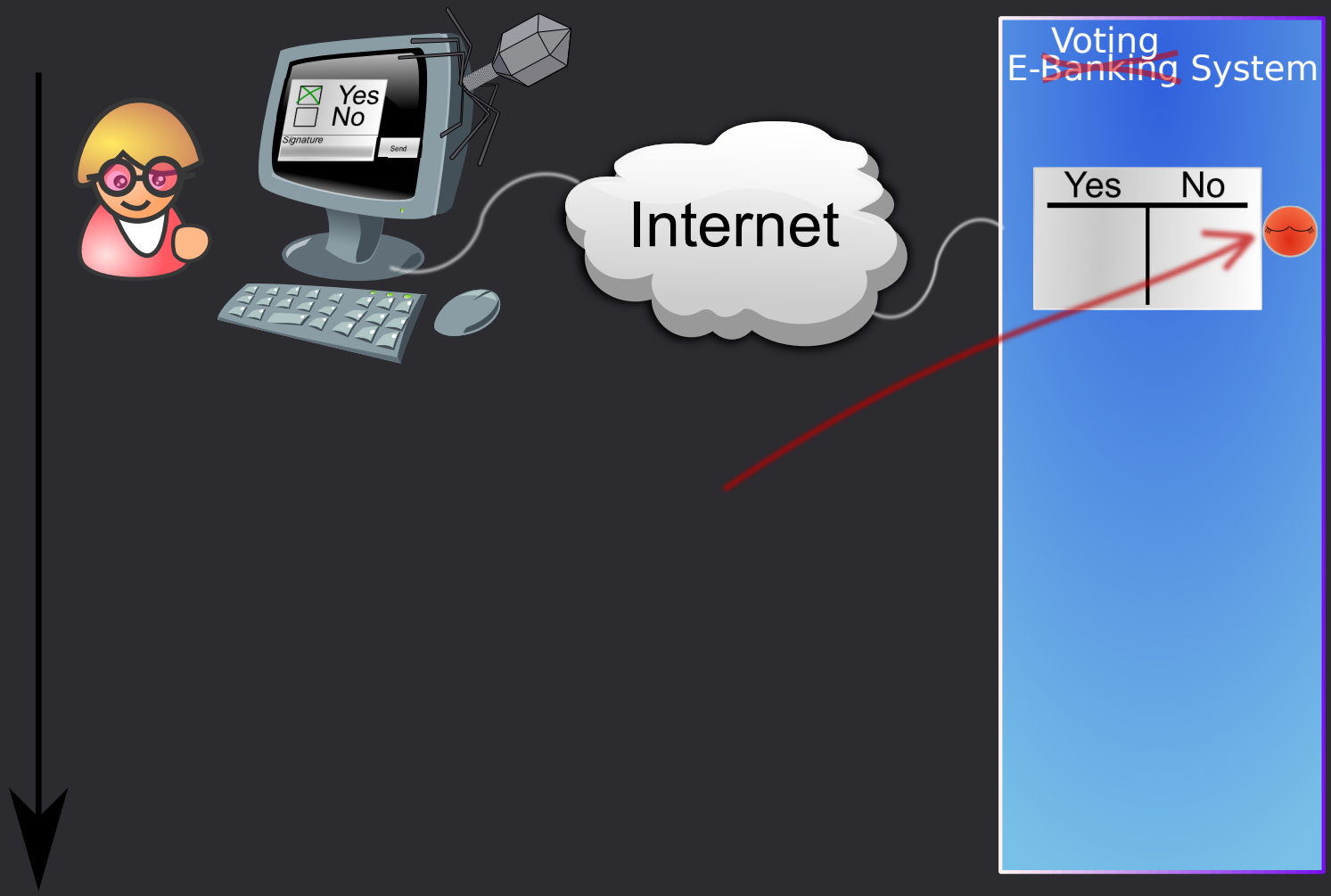
...in order to eliminate the possibility for bribary and coercion



The system has to become blind during the vote casting process...



...in order to guarantee privacy and fairness



Eh... Is that what we want? Questions arise on either side

Did my vote count?

- Encrypted as intended?
- Cast as encrypted?
- Recorded as cast?
- Decrypted as recorded?
- Counted as decrypted?

Is my privacy guaranteed?

- No linking back from my cleartext vote to me?

Which votes shall count?

- Only votes from eligible voters
- One vote per eligible voter

Will we make it in time?

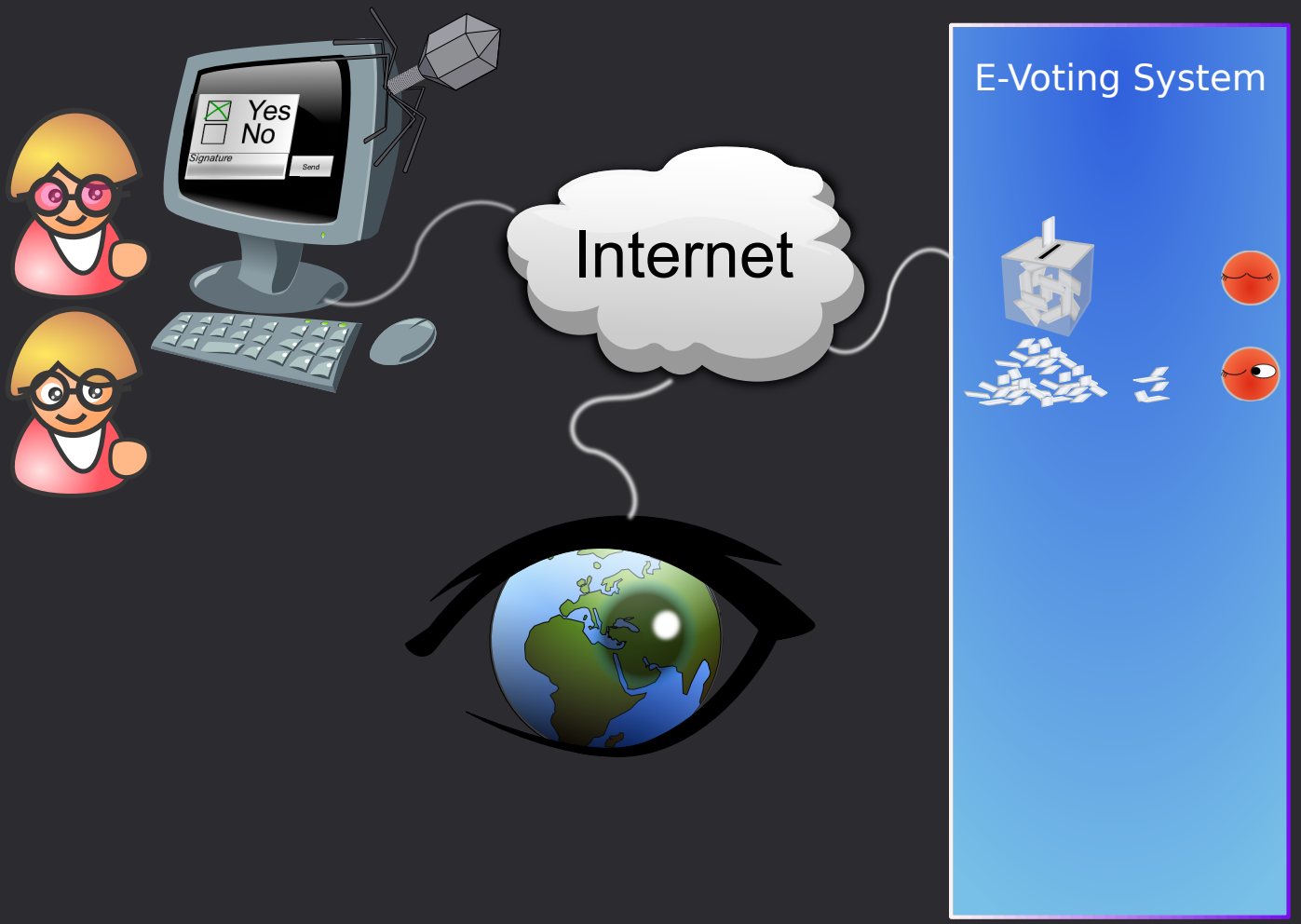
- Test eligibility
- Test duplicates
- Decrypt
- Count

Question

How should a true e-voting scheme look like?

What is already known... Blind the system during cast of ballots

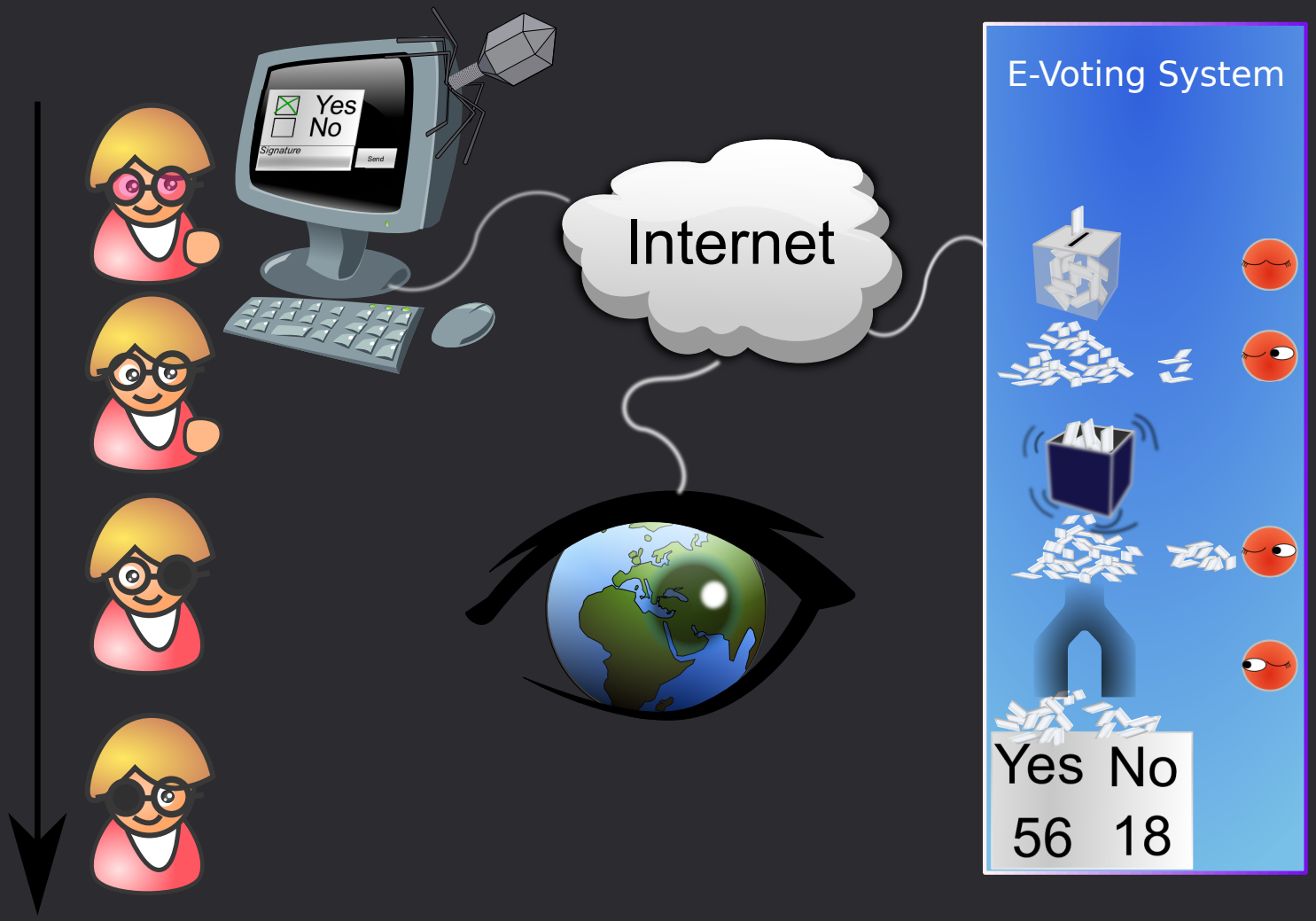
System publicly removes duplicate ballots, Alice can still see hers



System shuffles the remaining ballots, Alice knows its in there

System publicly removes non authorized ballots not seeing any vote

System publicly decrypts the votes and counts them



That is what we want! Just one question remains on the system side

My vote did count!

- Encrypted as intended!
- Cast as encrypted!
- Recorded as cast!
- Decrypted as recorded!
- Counted as decrypted!

Is my privacy guaranteed?

- No linking back from my cleartext vote to me!

The real votes count!

- Only votes from eligible voters
- One vote per eligible voter

Will we make it in time?

- Test eligibility
- Test duplicates
- Decrypt
- Count

It even allows stronger statements about universal verifiability!

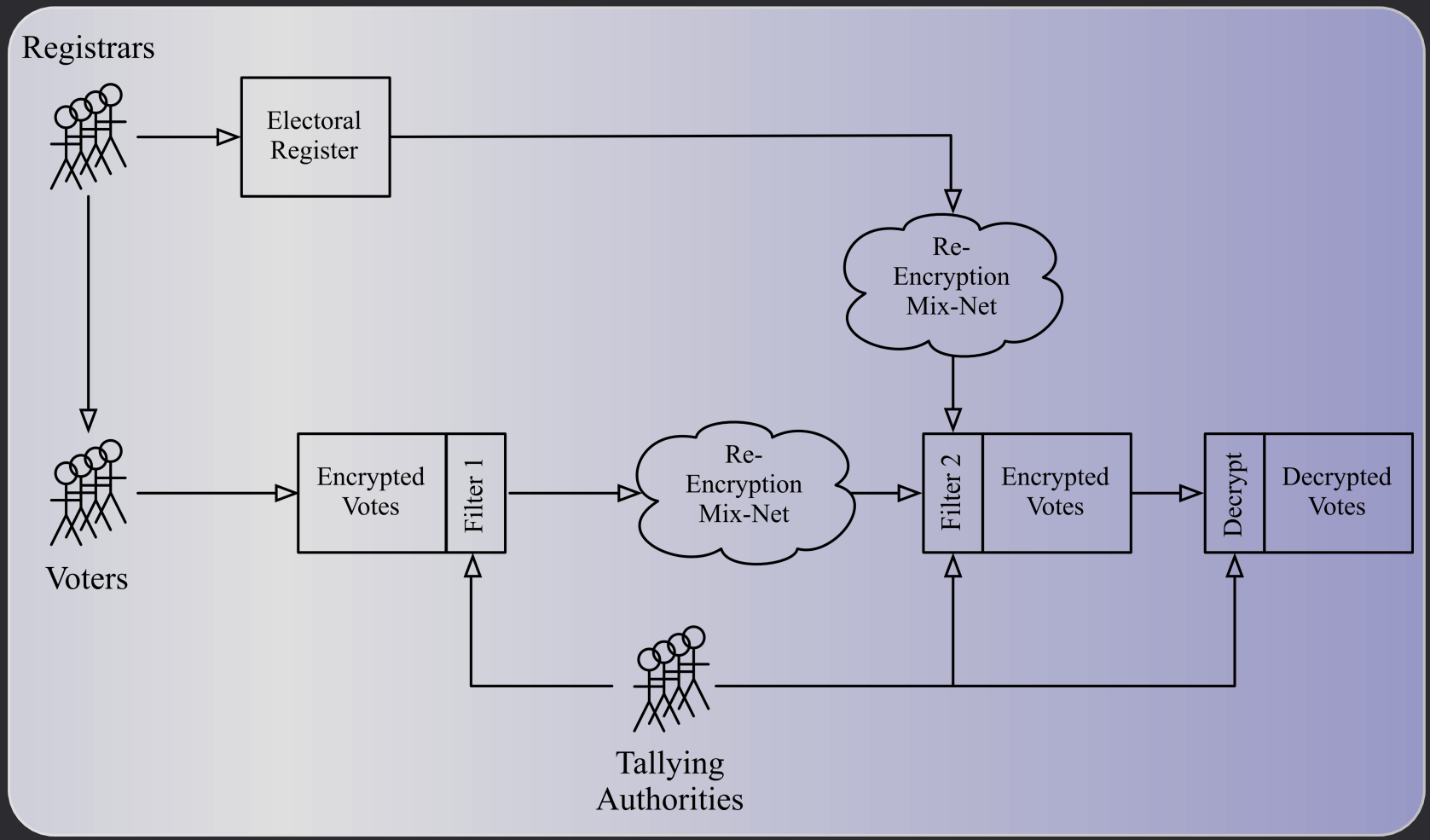
Fairness can be guaranteed!

- No results available prior the end of ballot casting

Democracy can be guaranteed!

- Every eligible voter can vote
- There are no duplicate votes
- No 'additional' votes have been introduced
- No authorized vote has been deleted

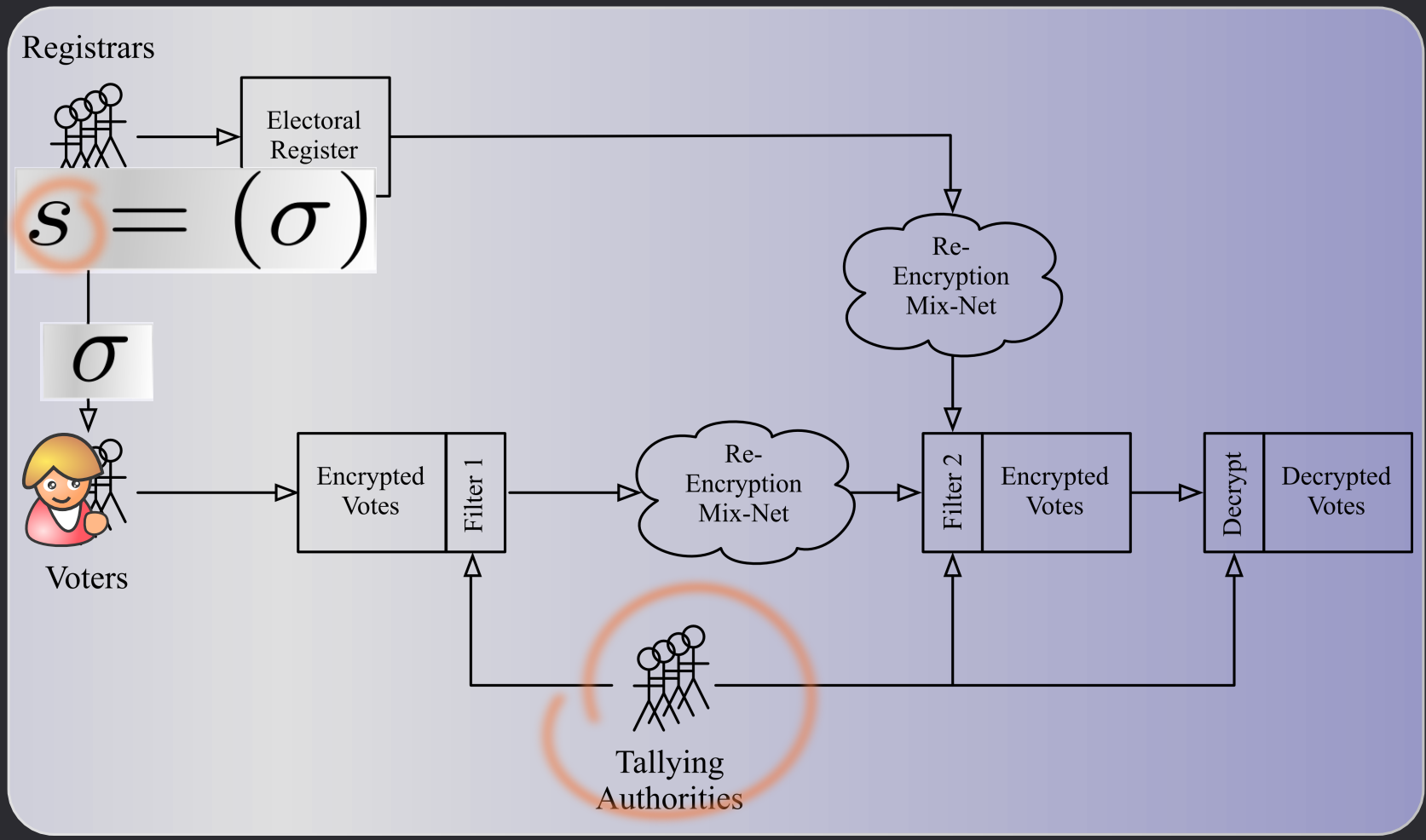
Overview


Registrars
Electoral Register
Re-Encryption Mix-Net
Voters
Encrypted Votes
Filter 1
Re-Encryption Mix-Net
Filter 2
Encrypted Votes
Decrypt
Decrypted Votes
Tallying Authorities

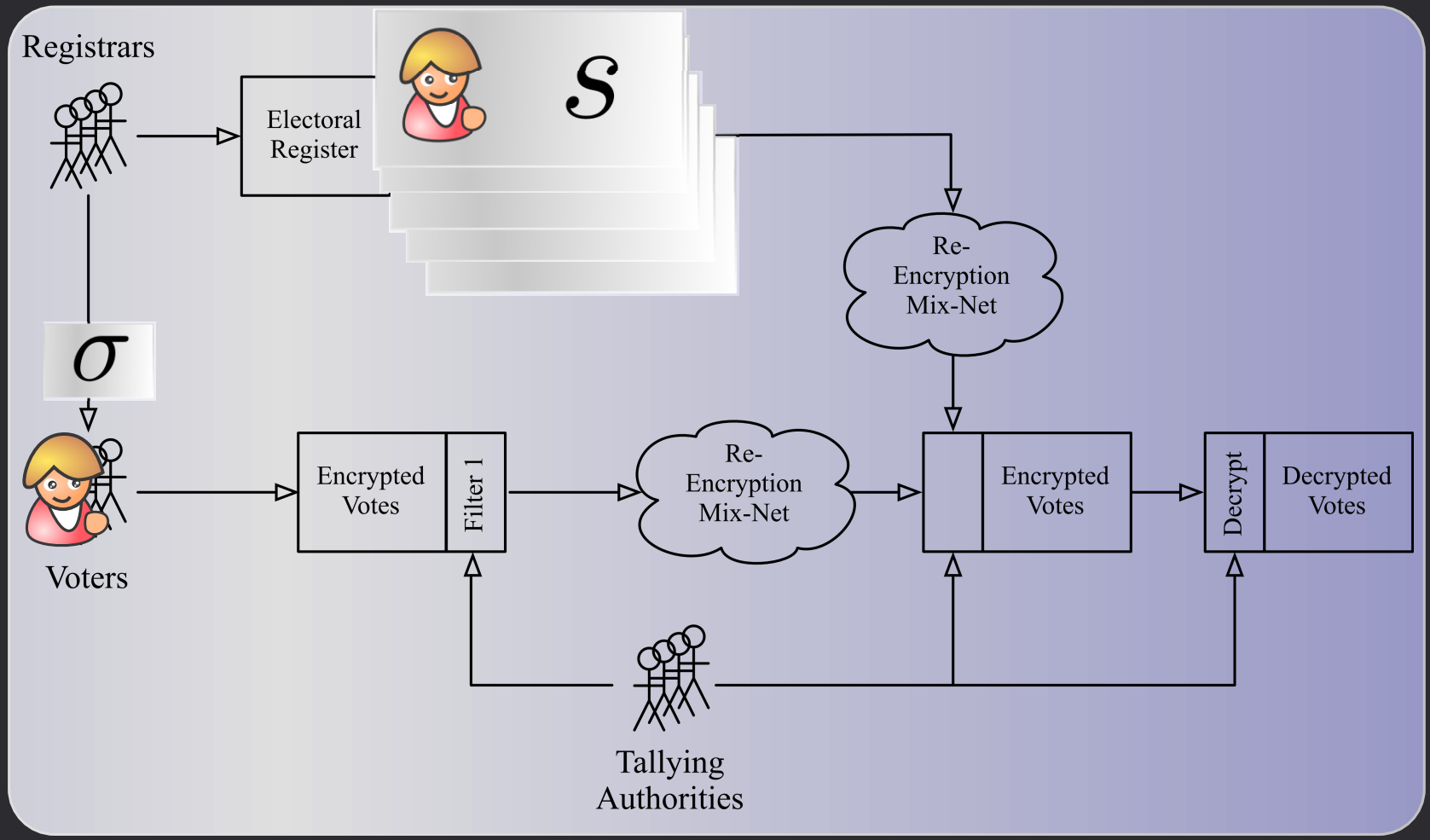
Question

What tricks are behind that scheme?

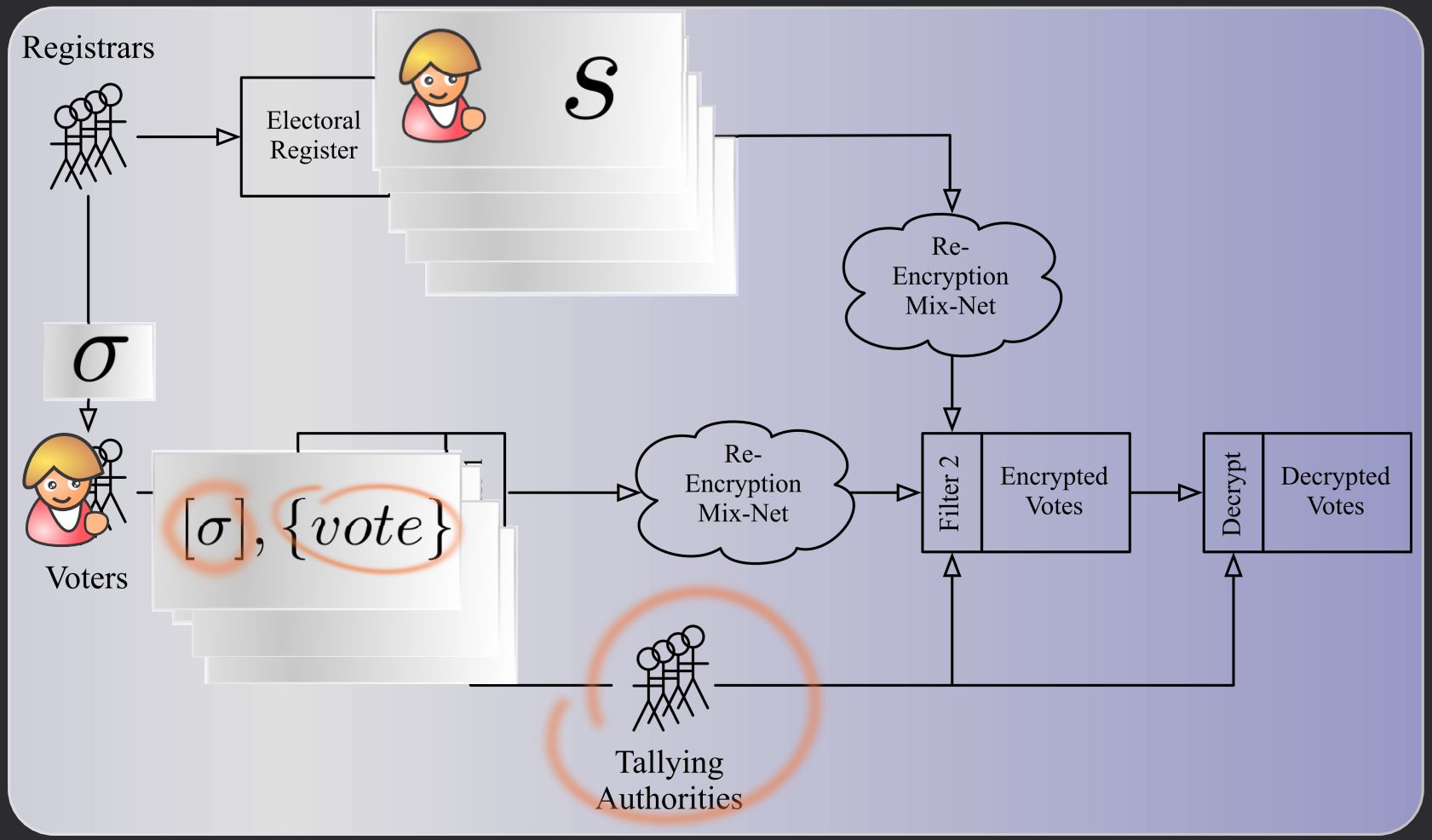
After setup, Alice knows a credential, system knows its encryption



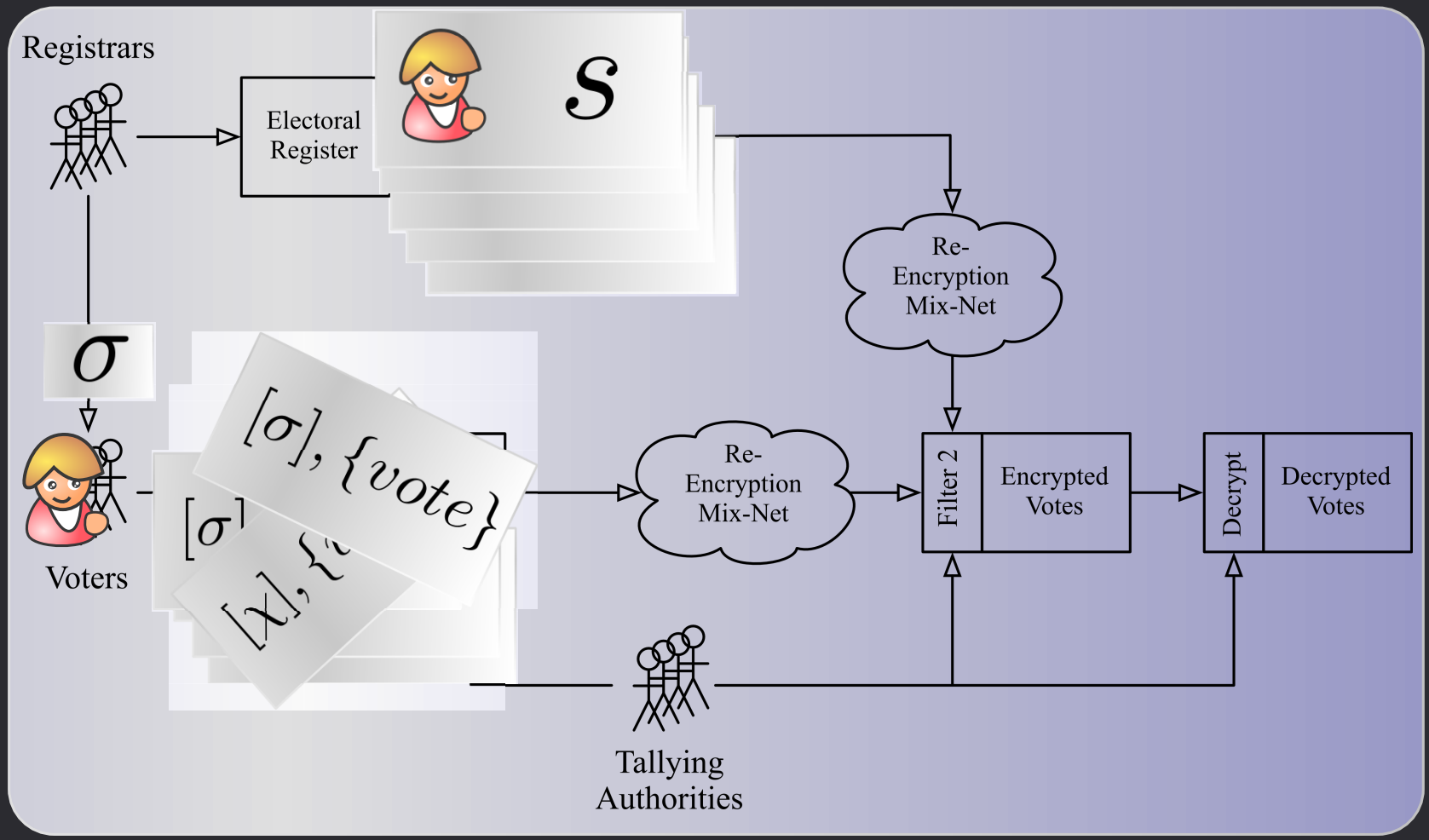
System publicly presents id & encrypted credential



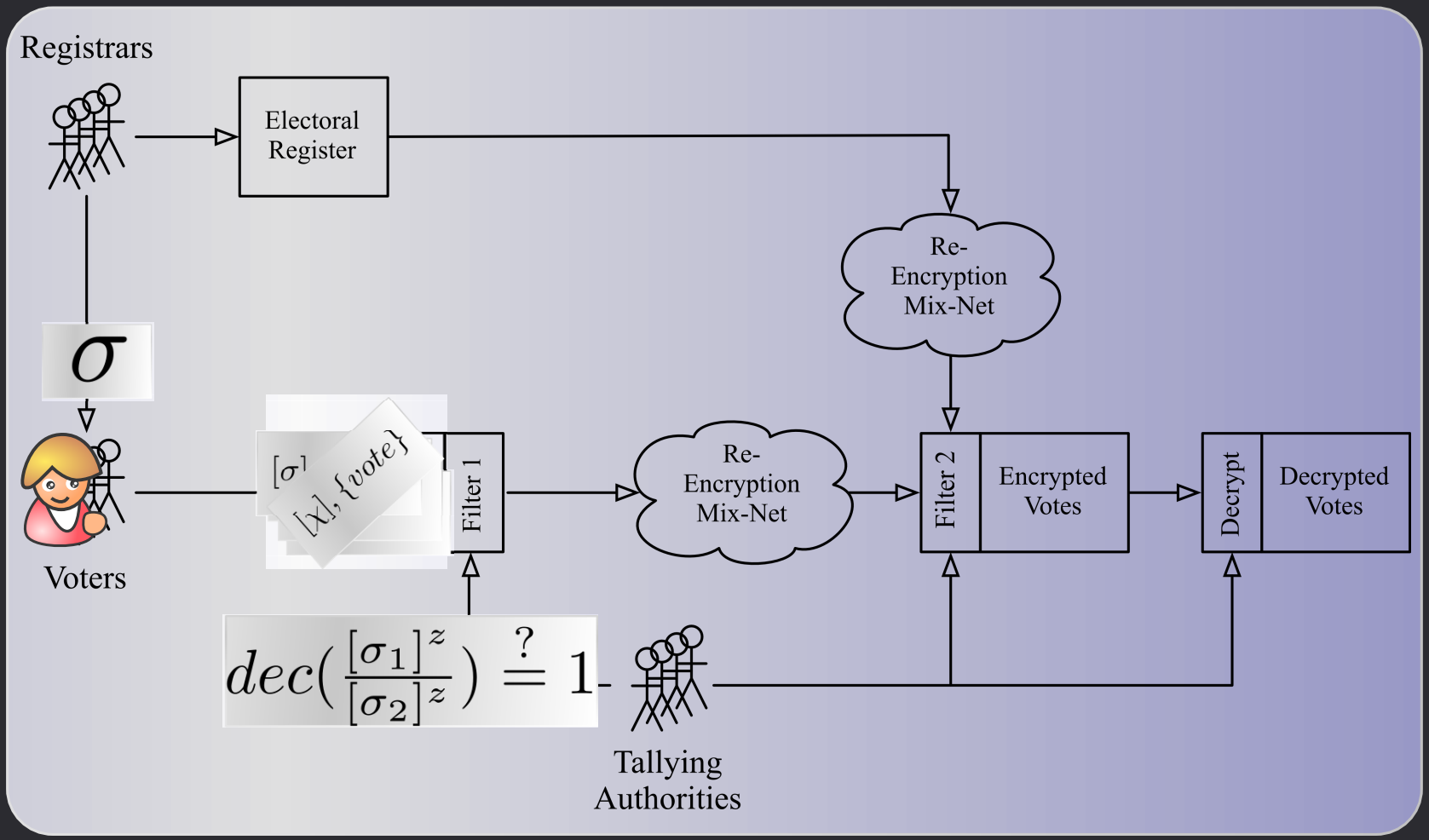
On vote-casting alice encrypts credential and vote



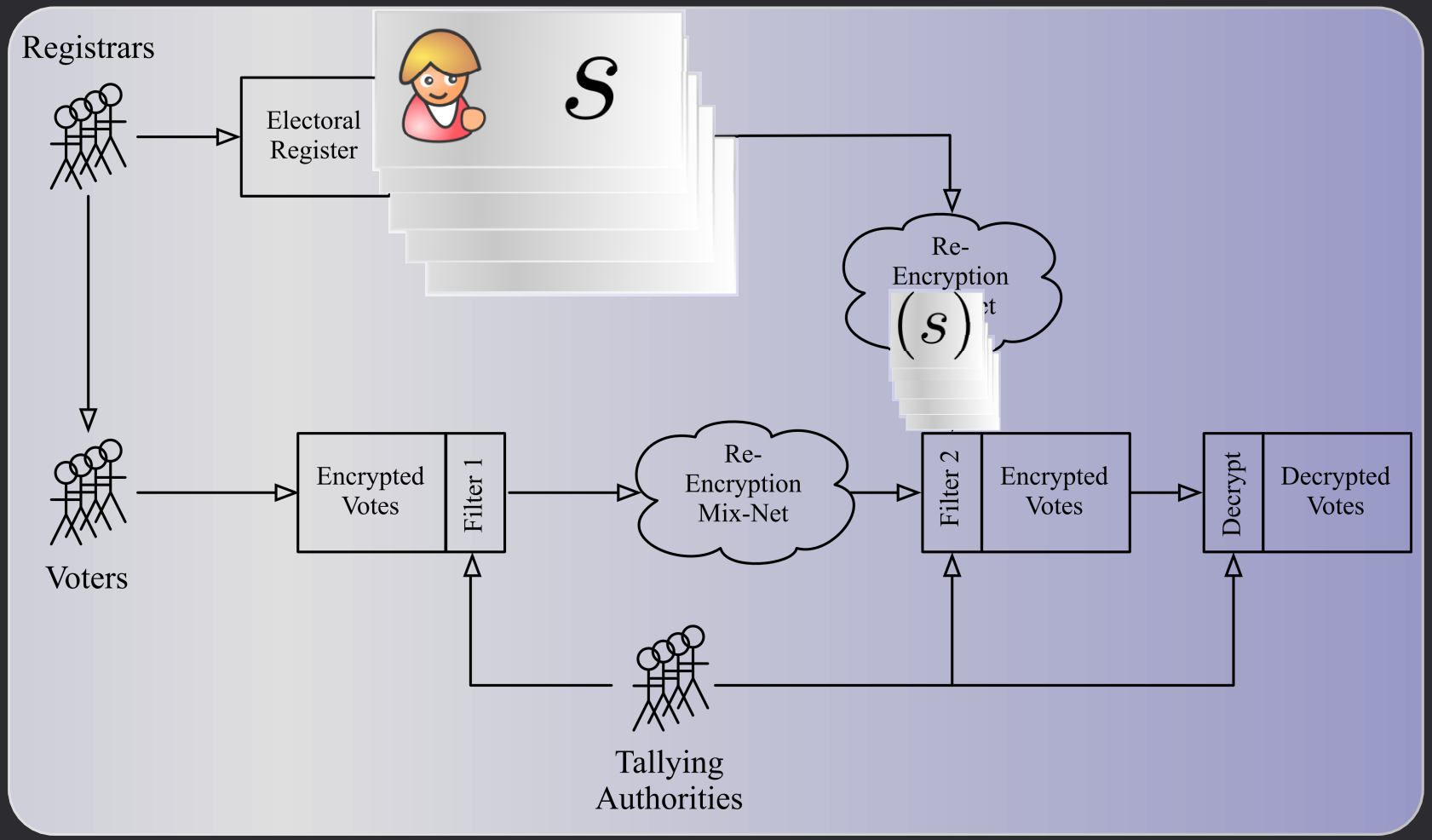
The system accepts ANY ballot



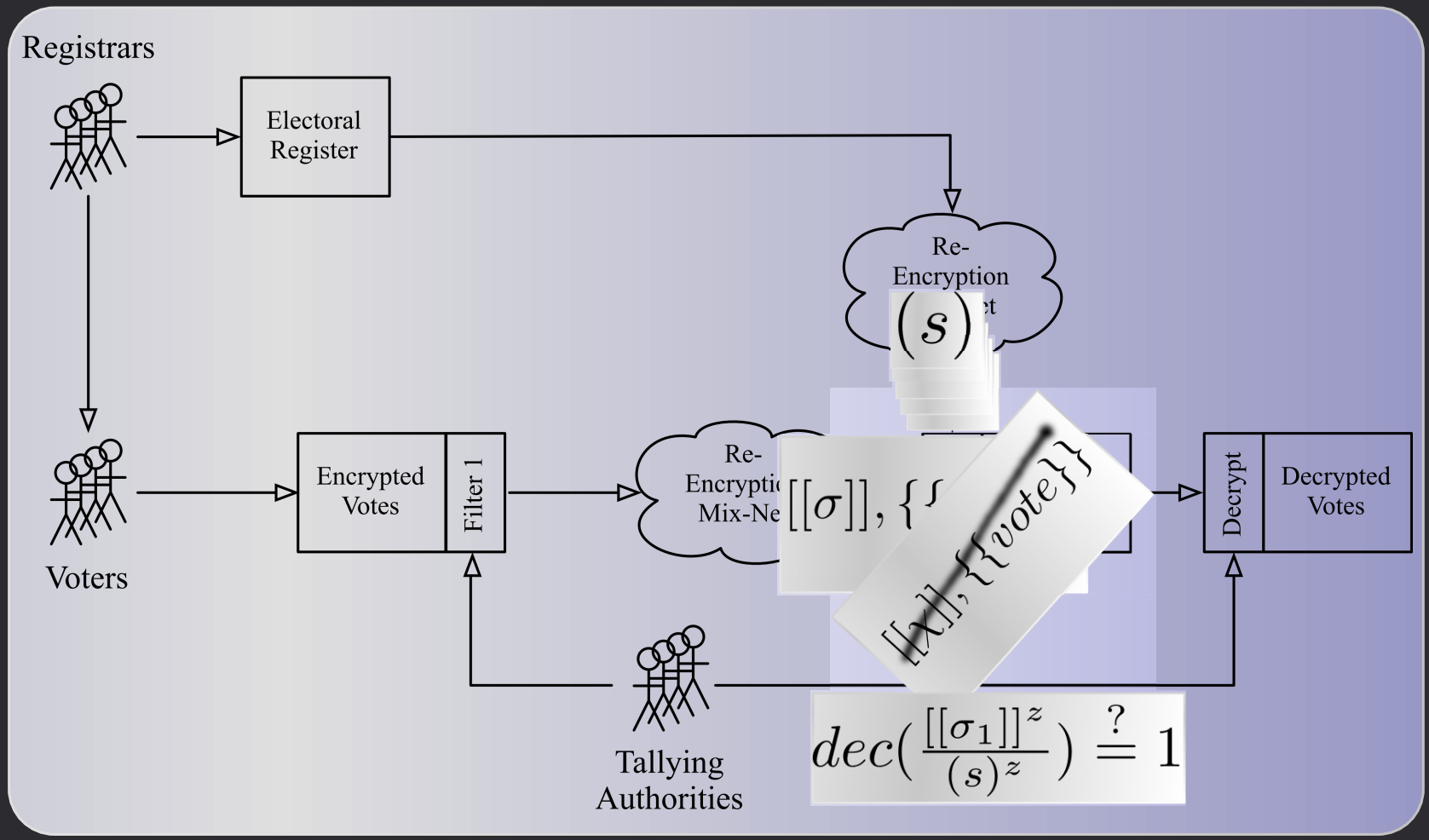
Pre-Tallying: System 'blindly' removes duplicate credentials



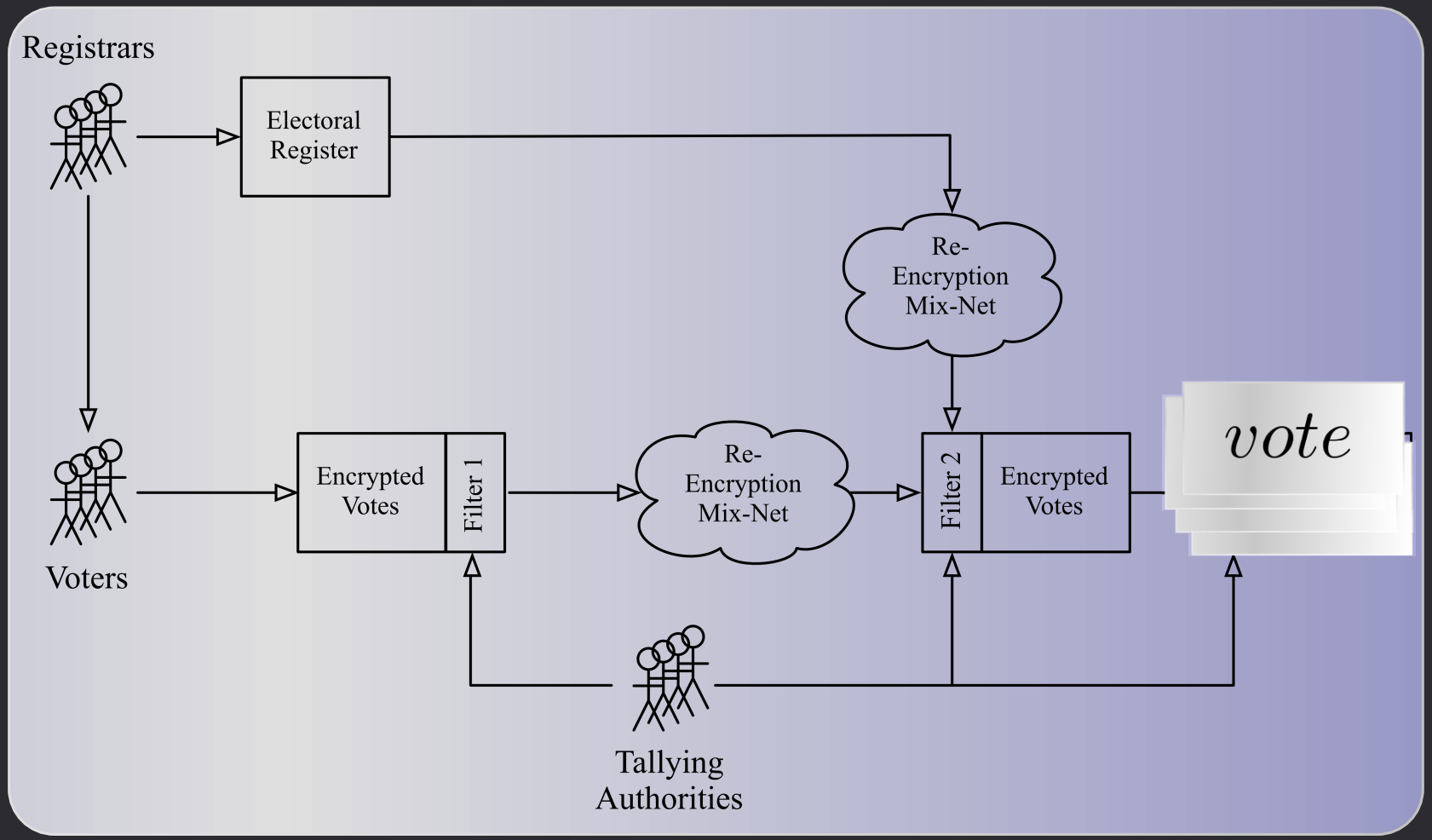
System creates a shuffled list of valid encrypted credentials



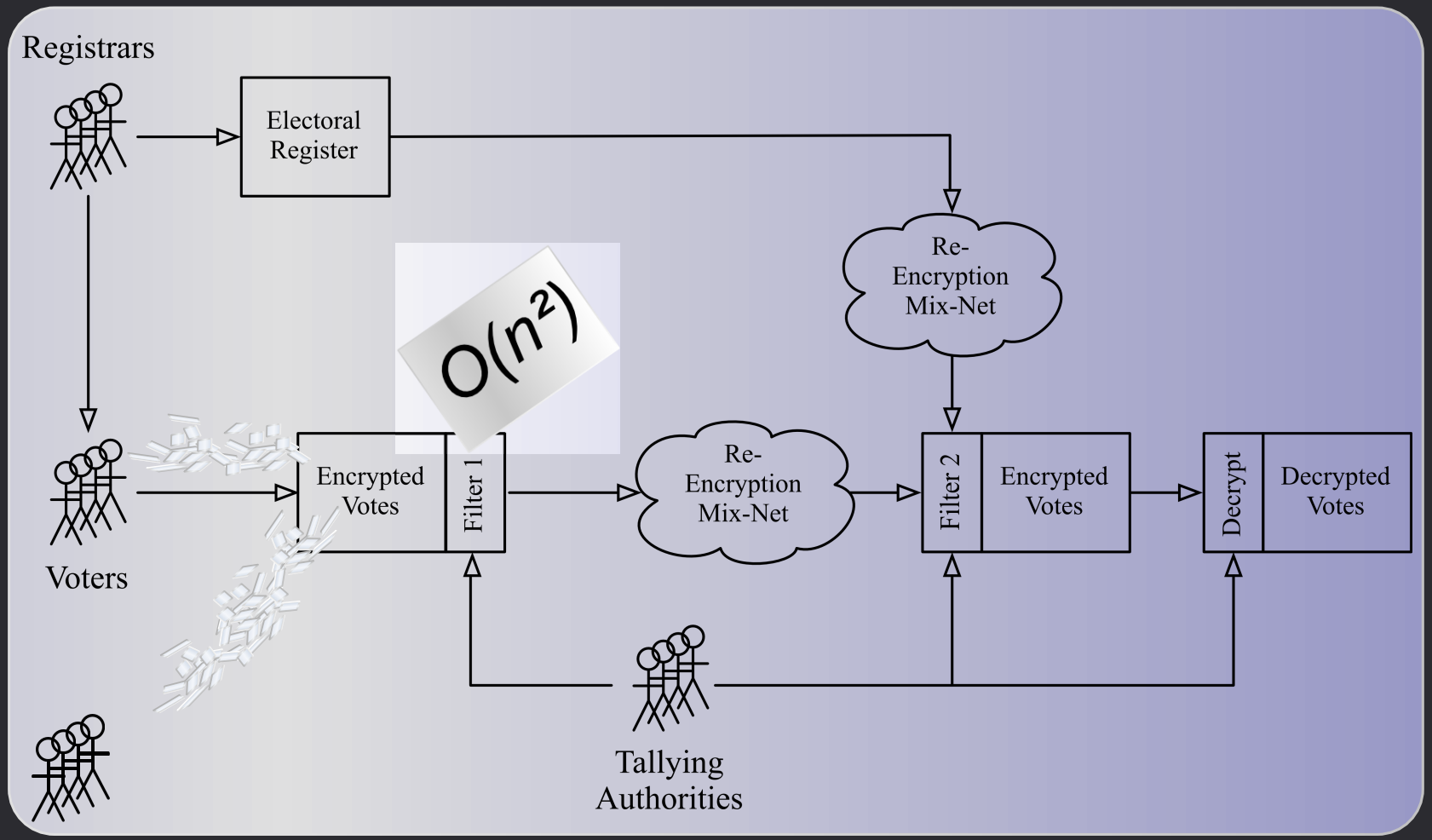
System 'blindly' removes votes with invalid credentials



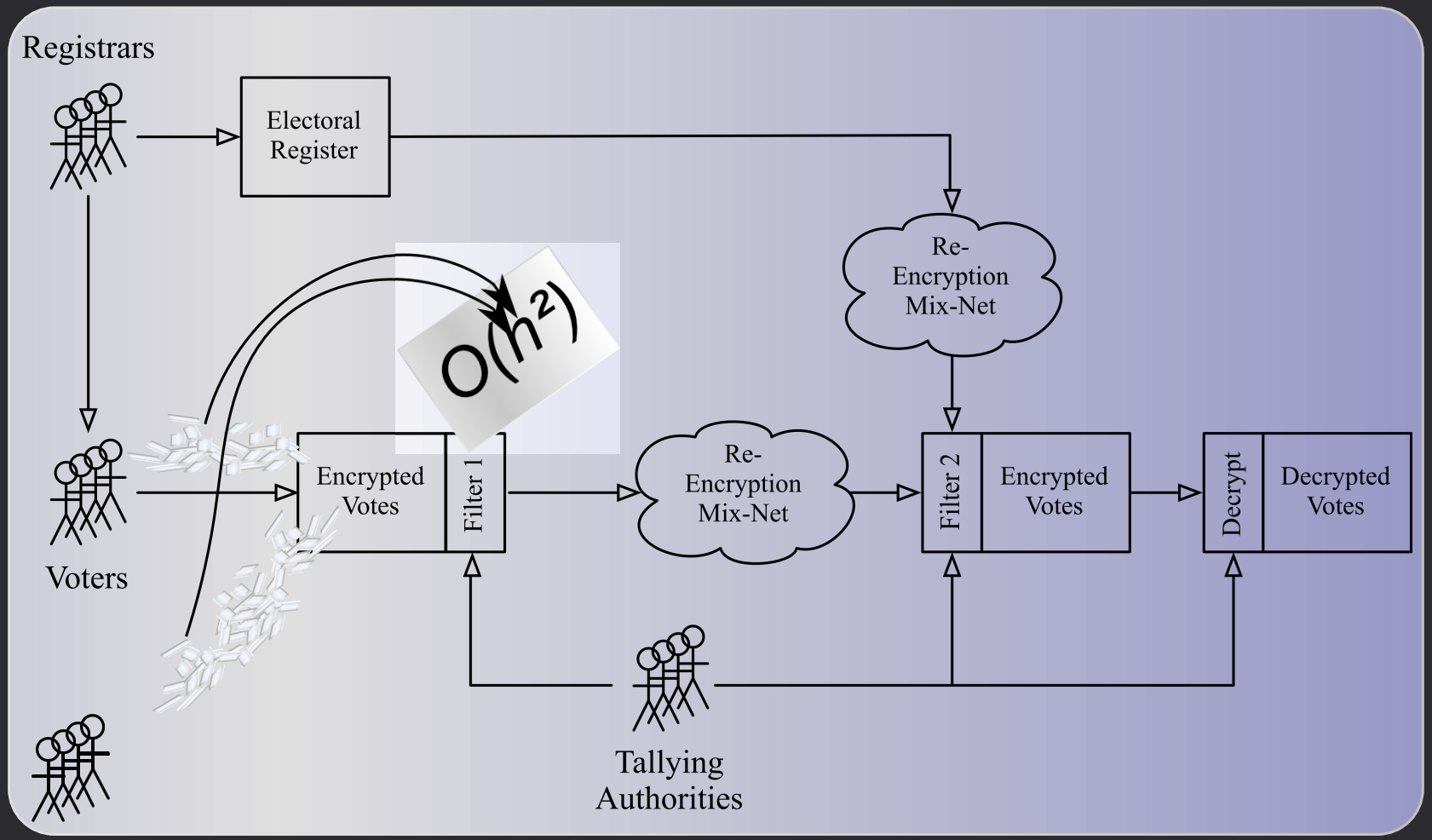
System decrypts remaining votes



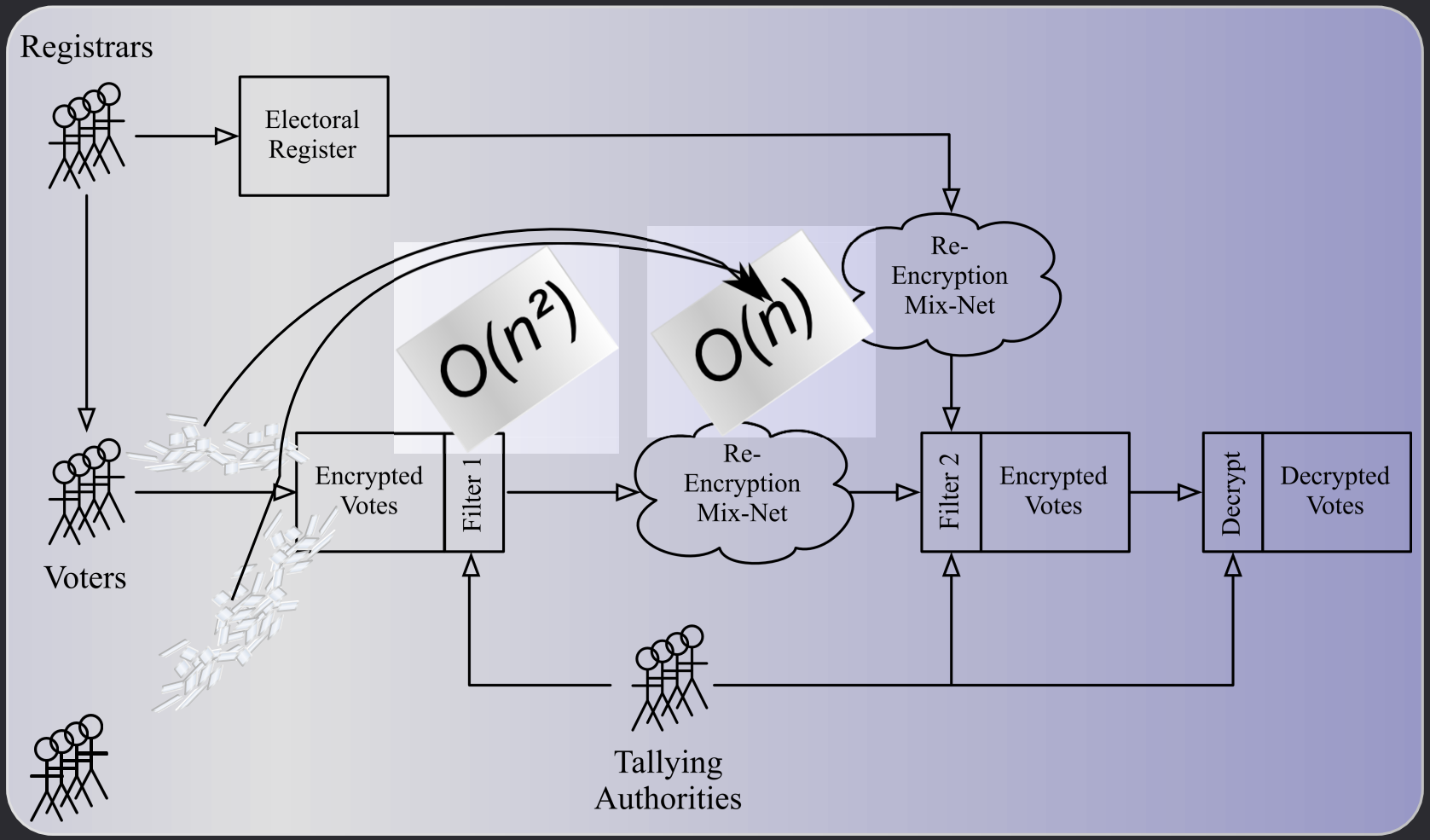
The time complexity of the scheme



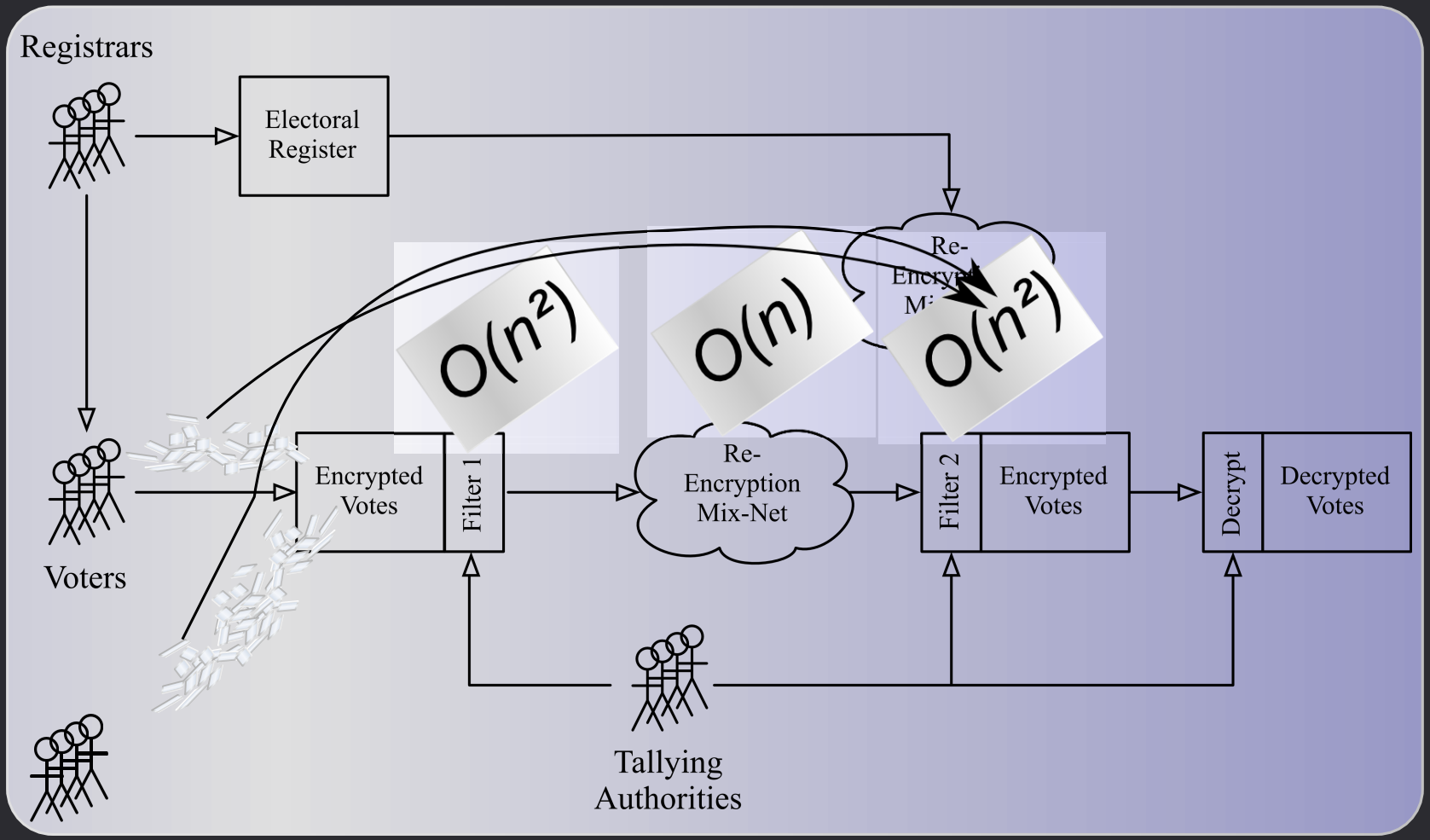
n depends on all the cast ballots



n depends on all the cast ballots



n depends on all the cast ballots



So the question remains...

Will we make it in time?

- Test eligibility (worst scenario depends on attacker!)
- Test duplicates (worst scenario depends on attacker!)
- Decrypt (worst scenario known in advance!)
- Count (worst scenario known in advance!)

The answer is given by a single anonymous attacker:

Never!

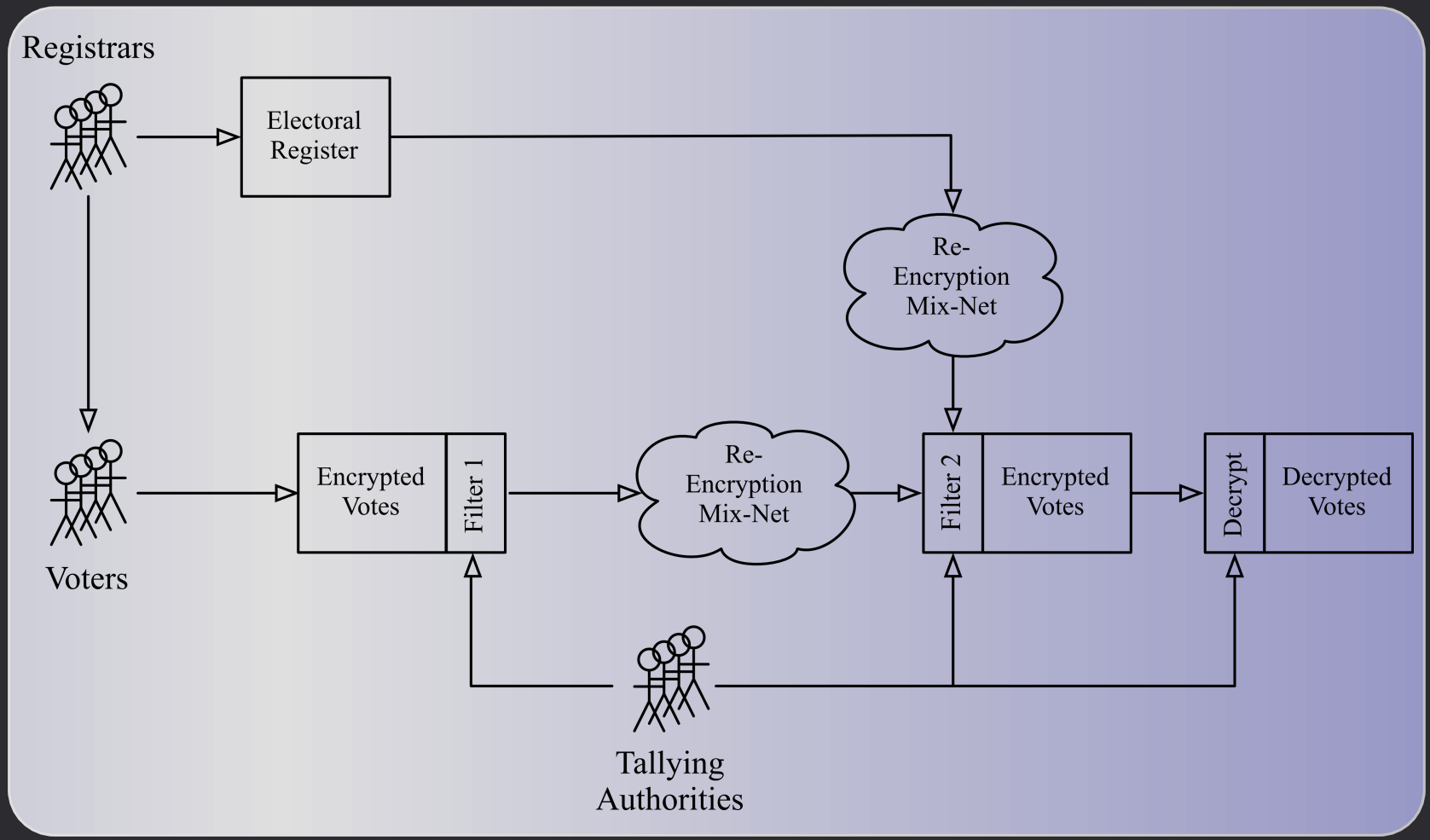
- Distributed attack with **a lot of** bogus votes

Question

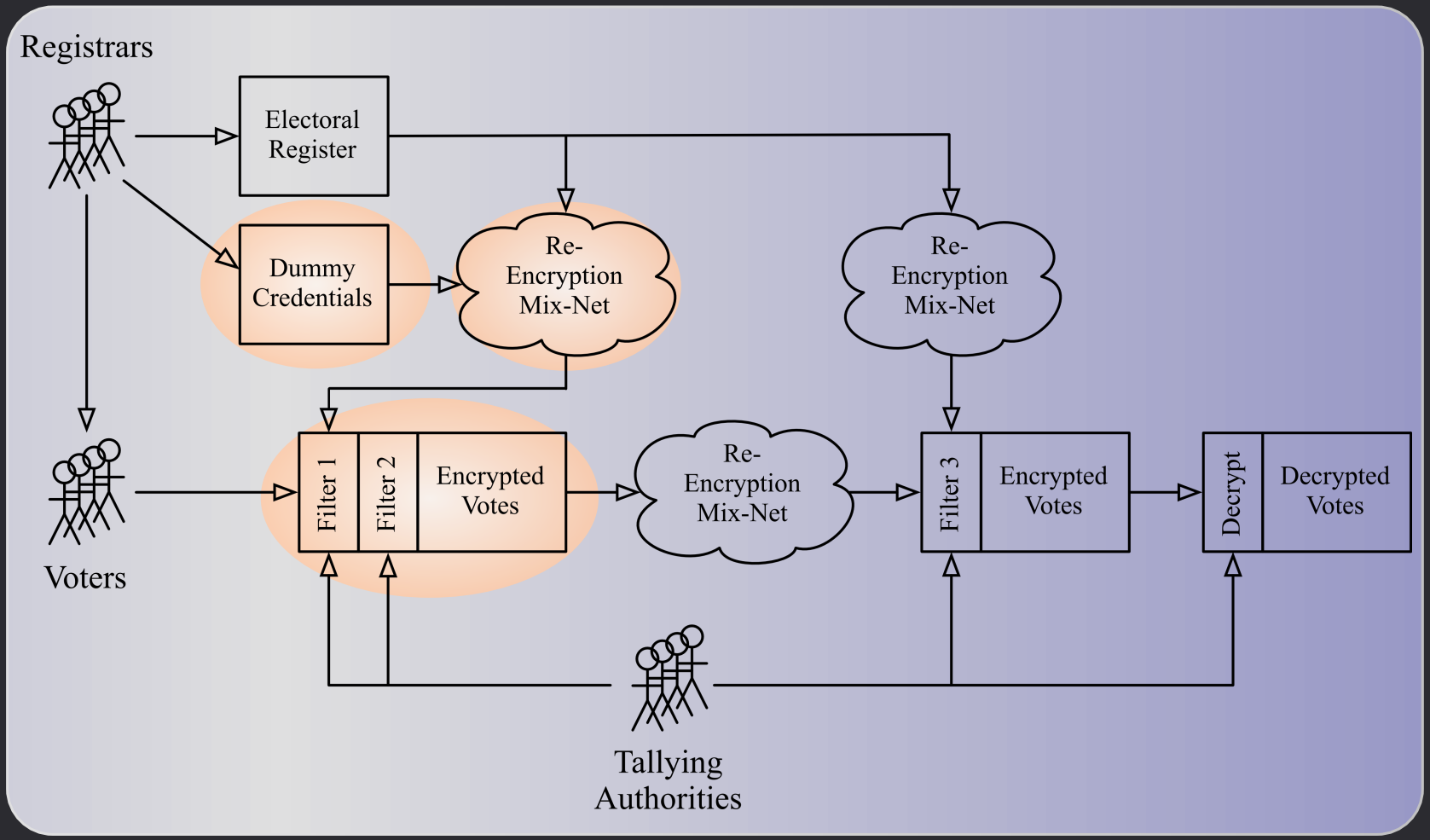
Do we really have to accept EVERY Ballot?

The modified scheme with time guarantee due to board flooding prevention

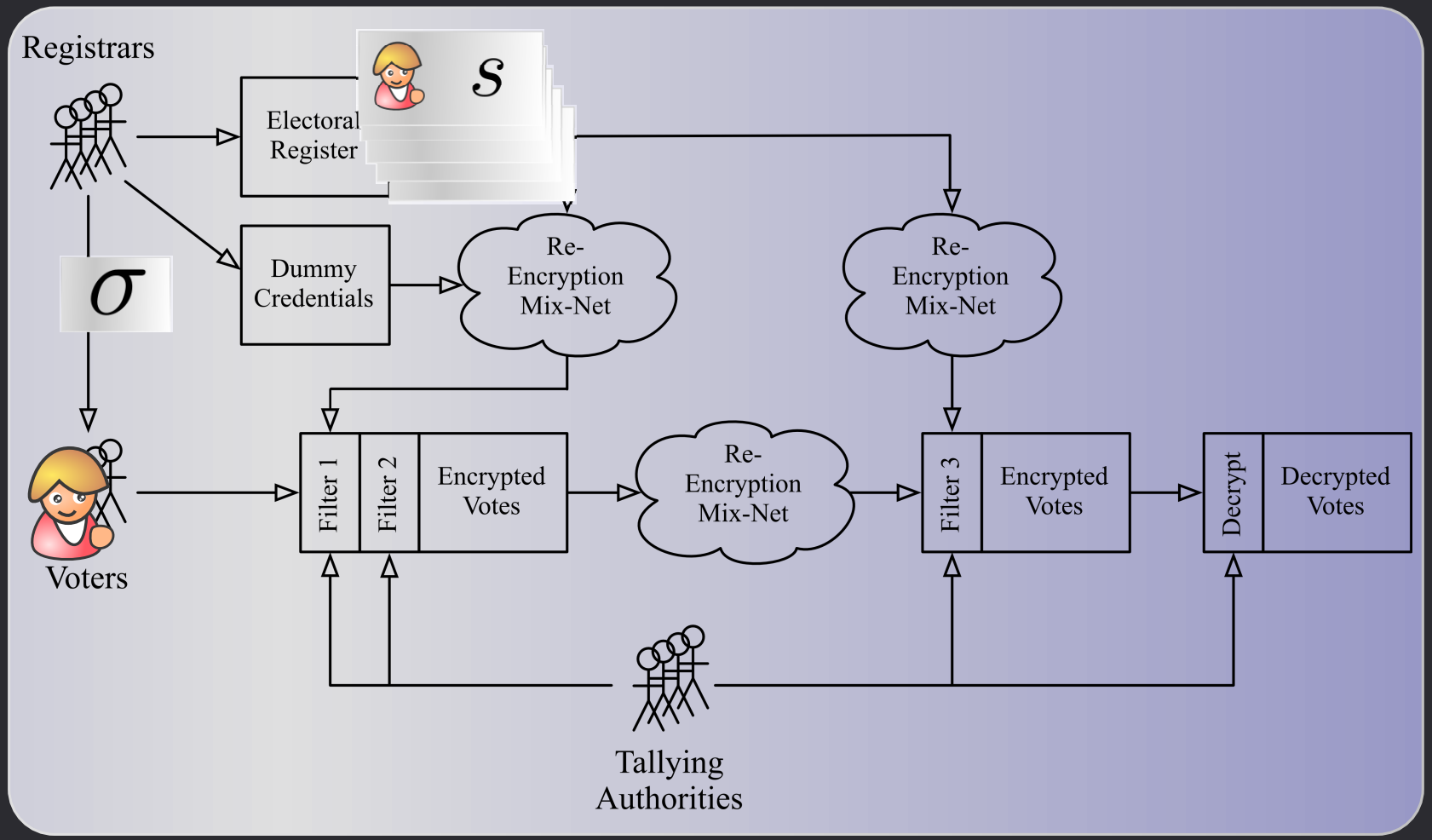
JCJ-05



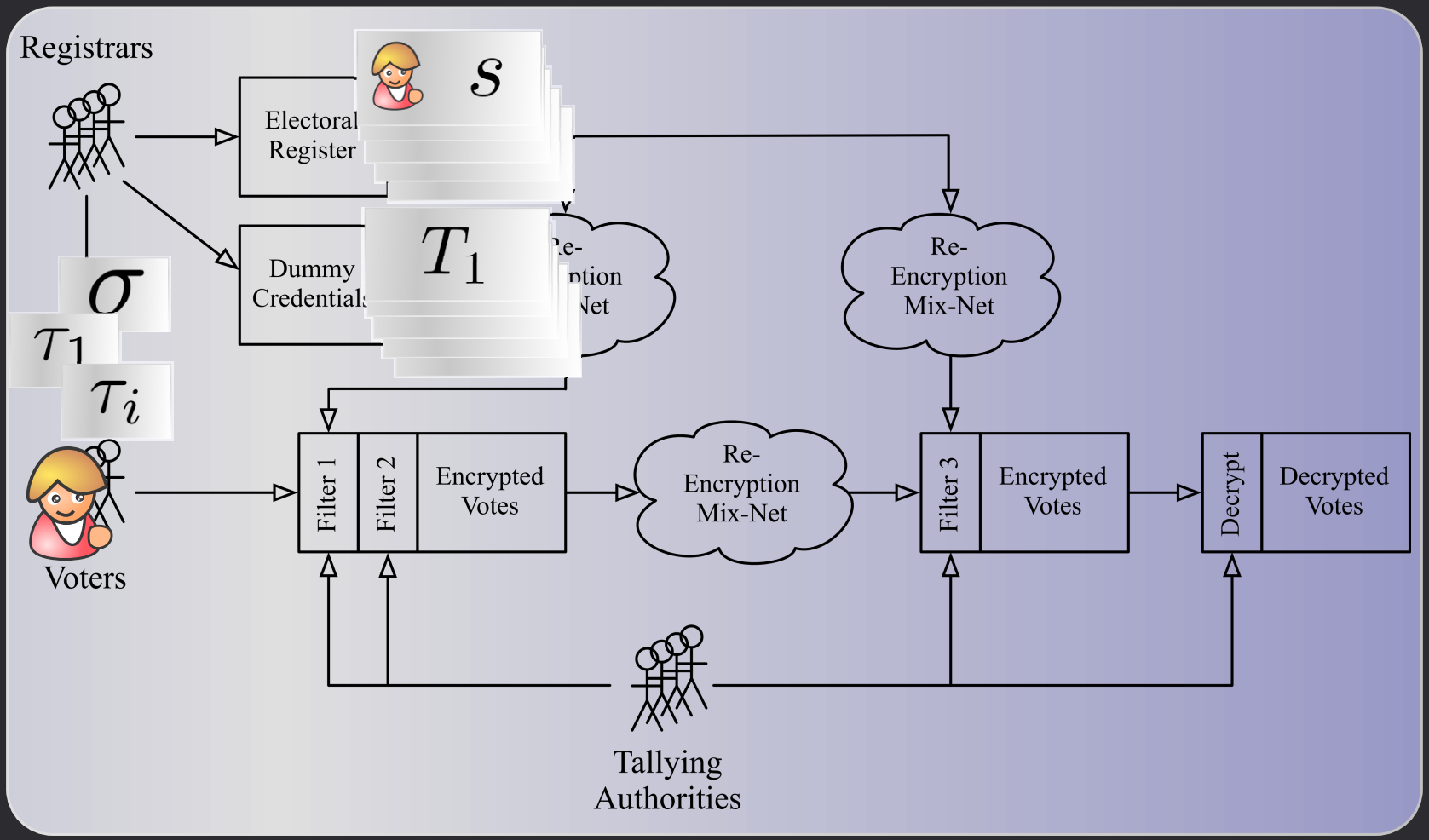
The modified board flooding resistant scheme



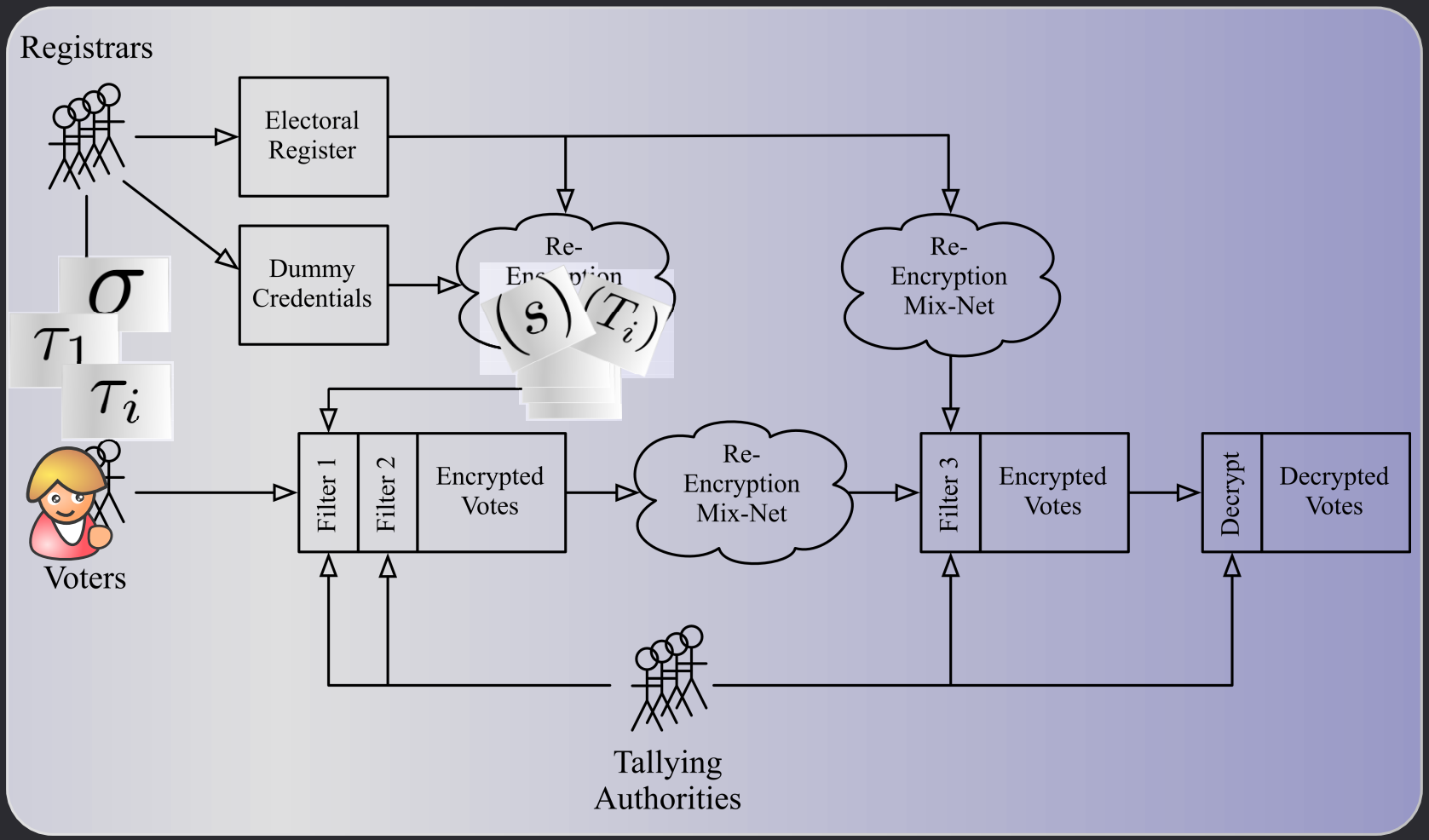
Alice knows a credential, system knows its encryption



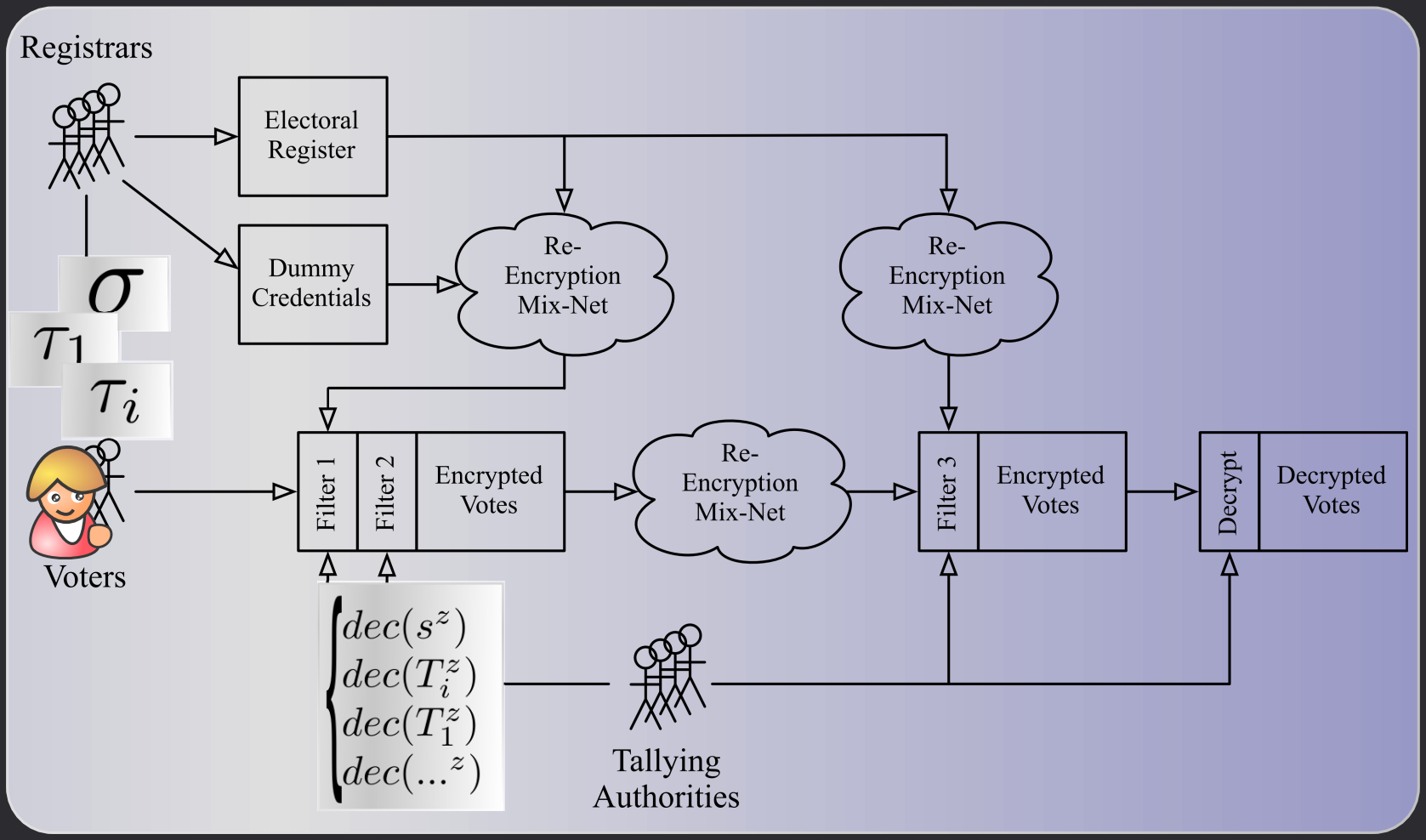
Alice gets dummy-credentials, system knows their encryption



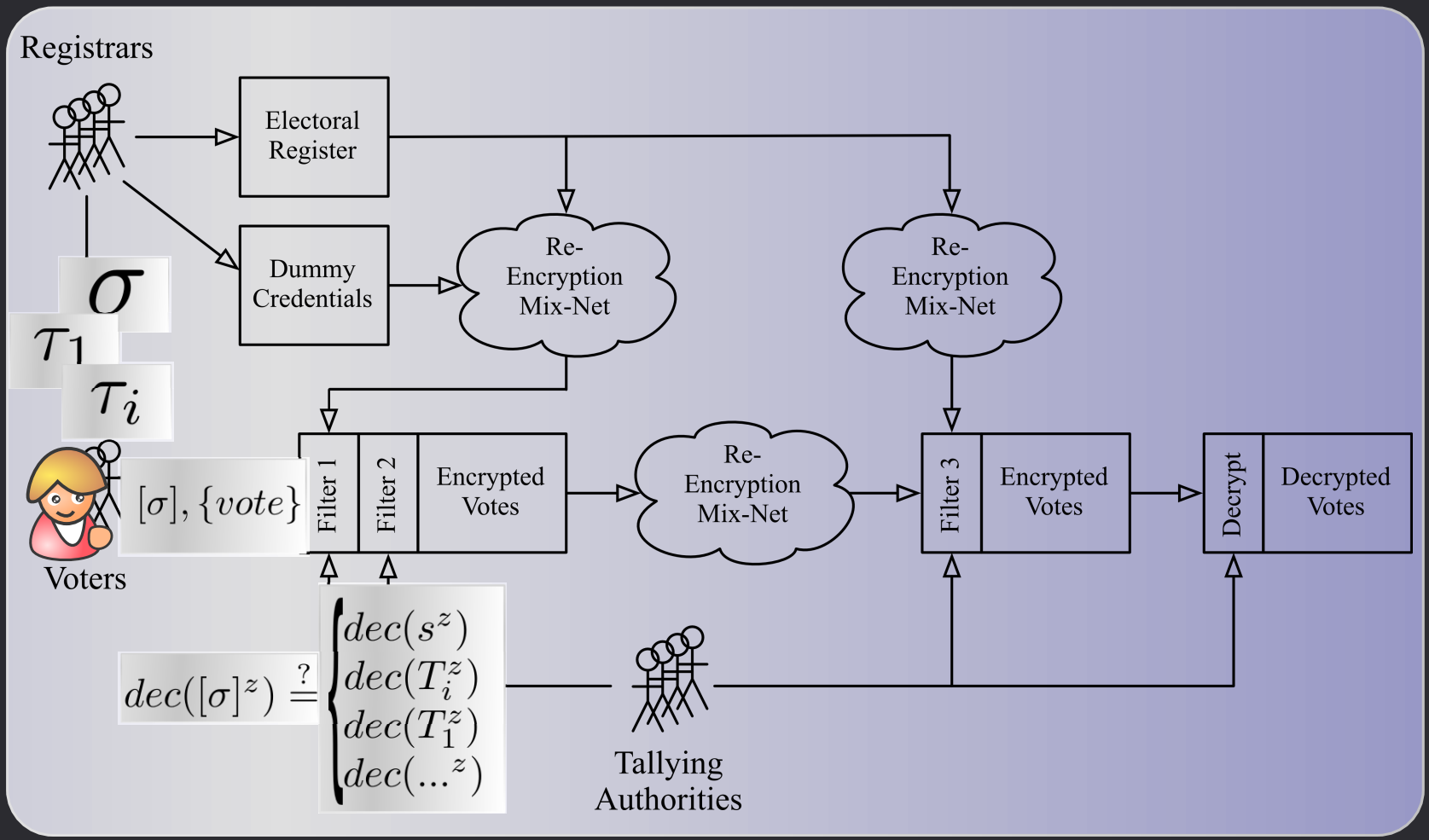
System creates a shuffled list of ALL encrypted credentials



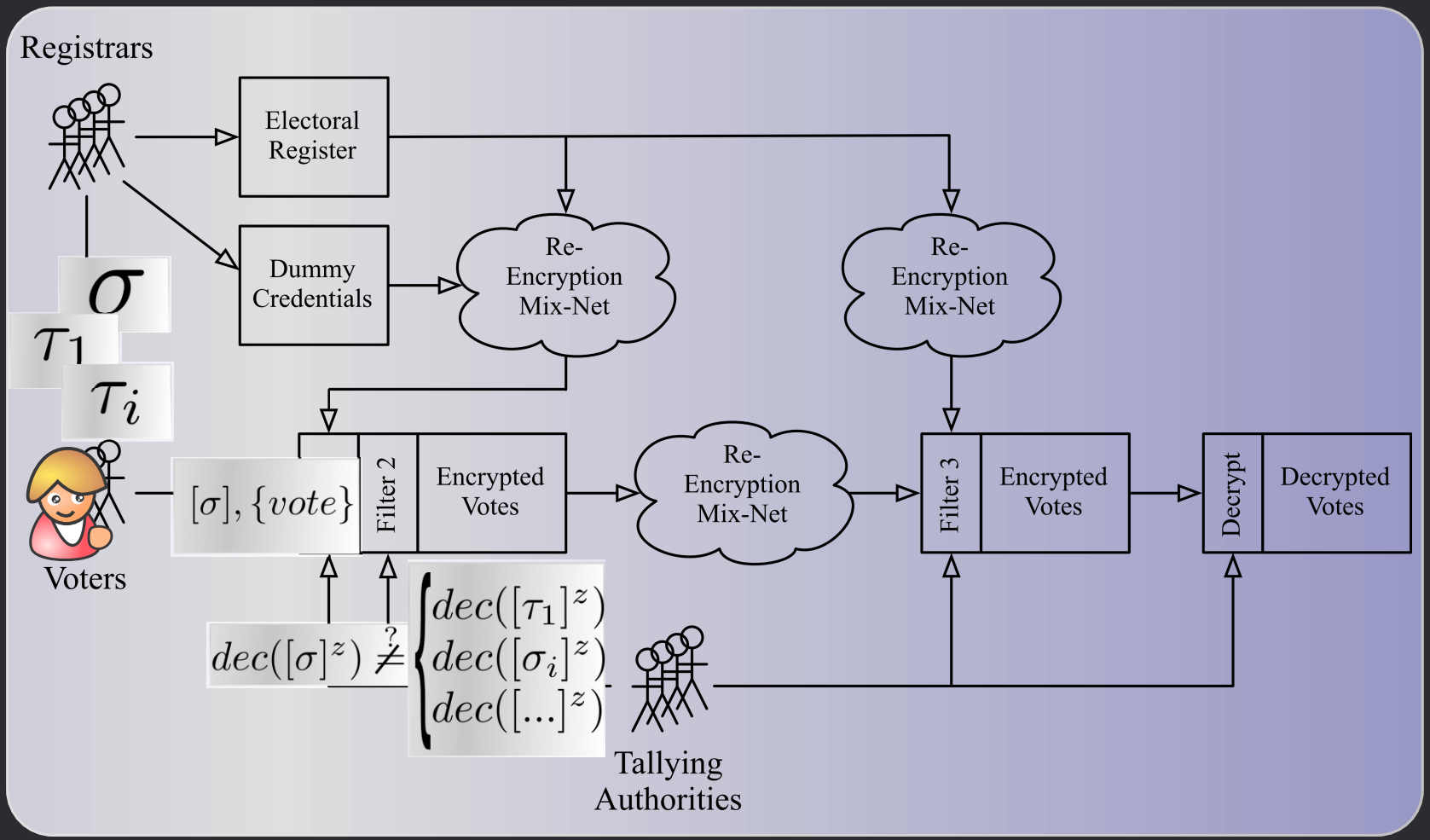
System creates a map with blinded decrypted credentials



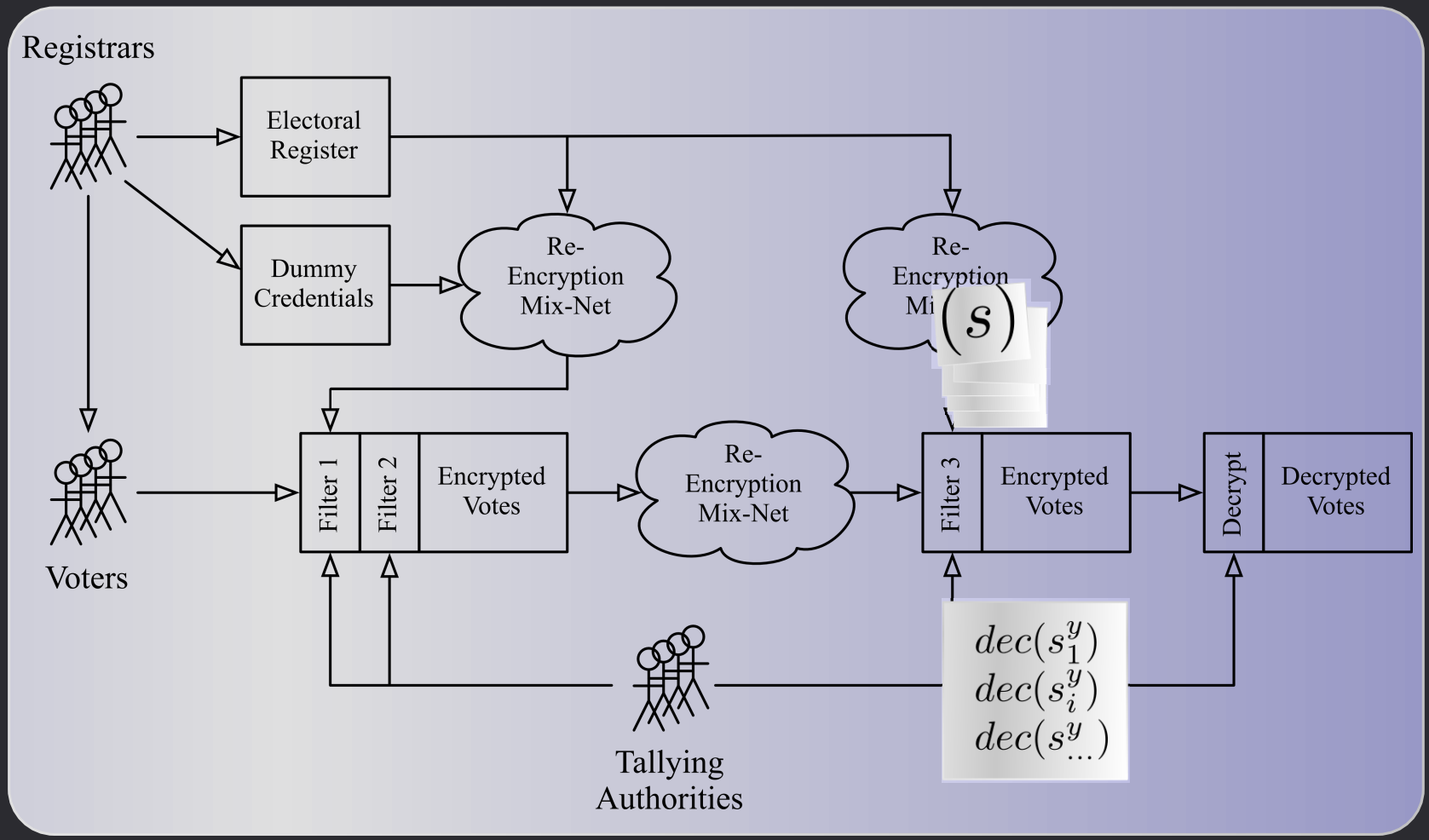
The filter 'blindly' rejects unknown credentials (bogus)



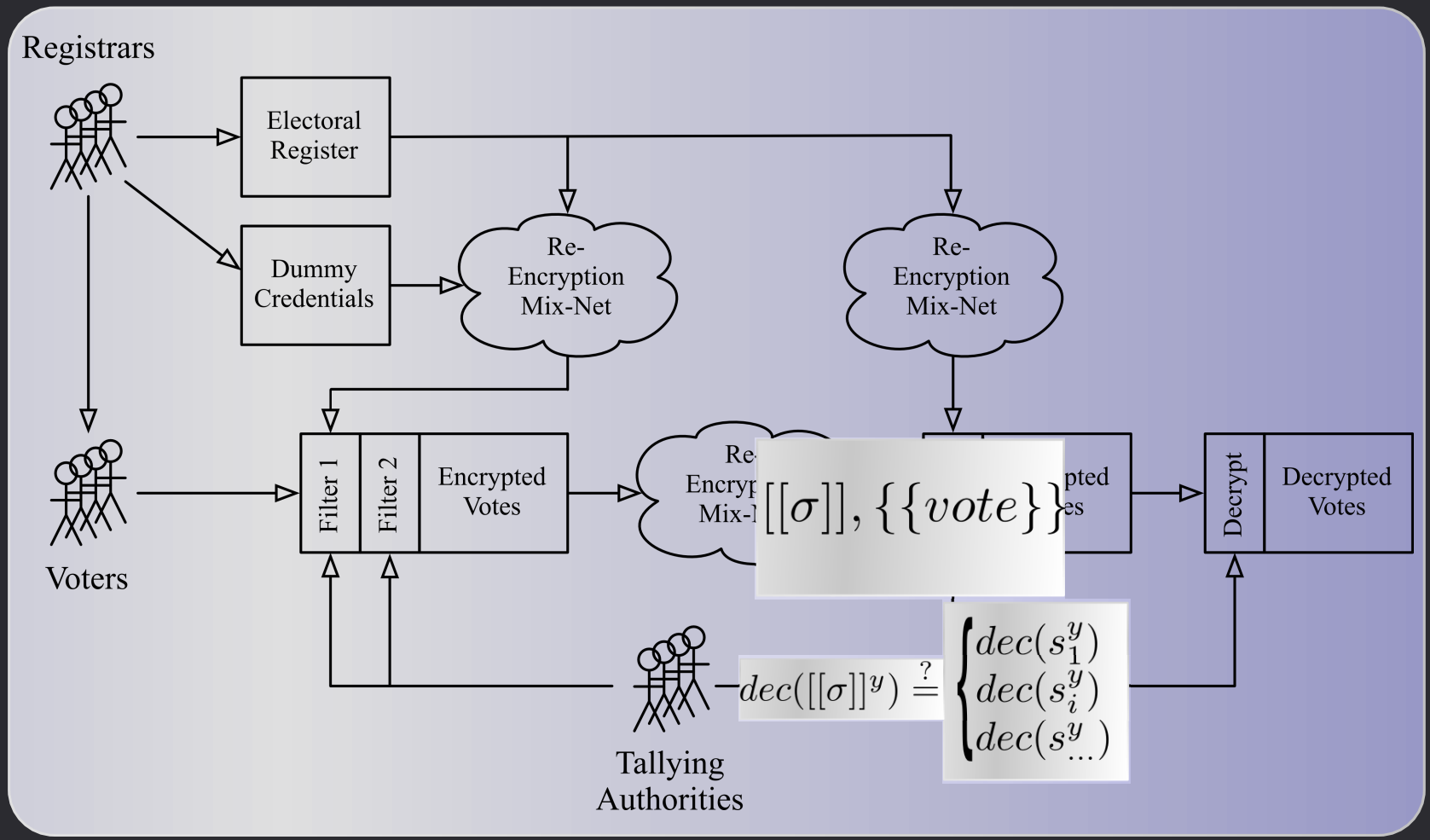
The filter 'blindly' rejects credentials already used (duplicates)



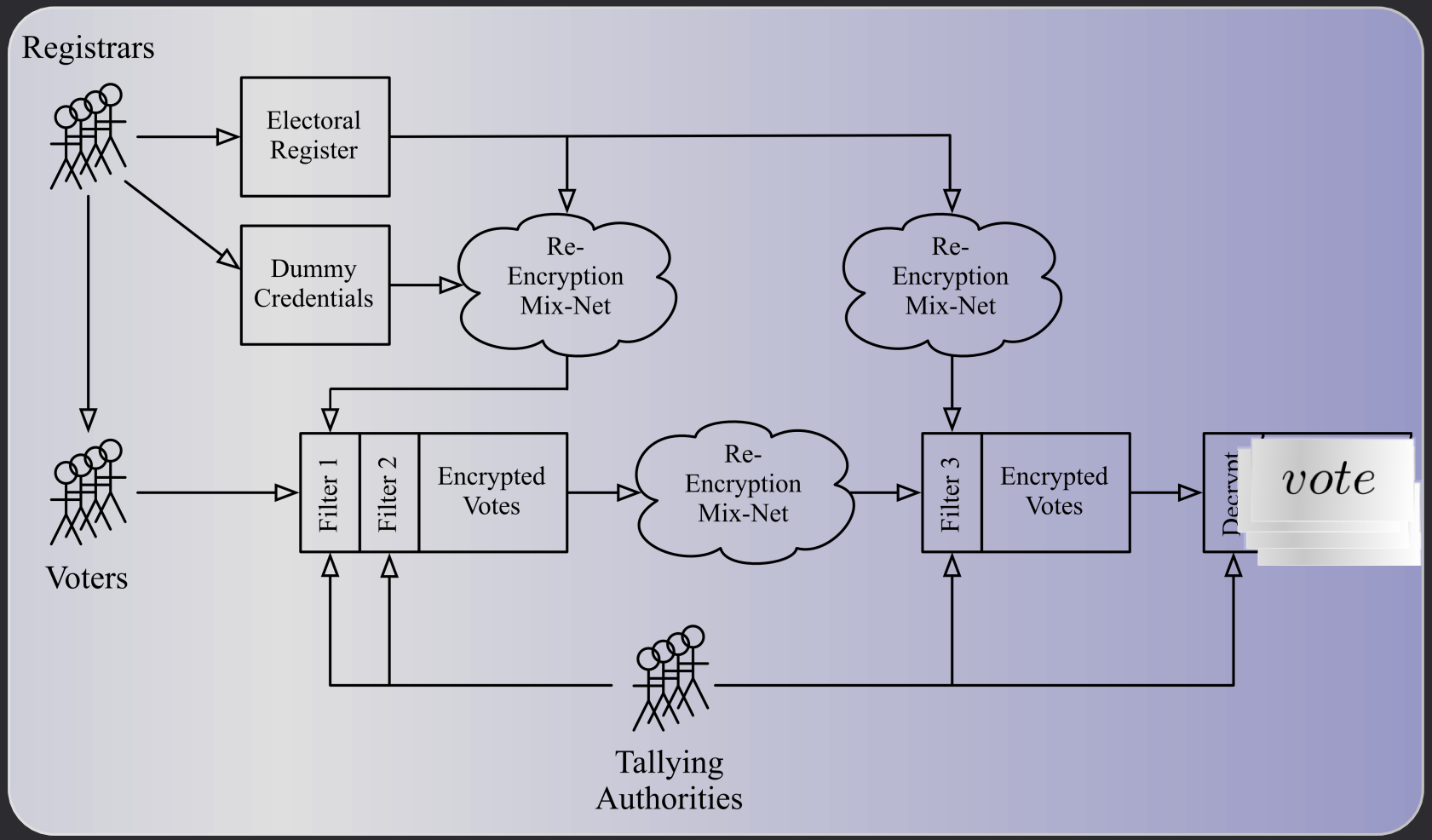
System creates a shuffled list of encrypted credentials σ



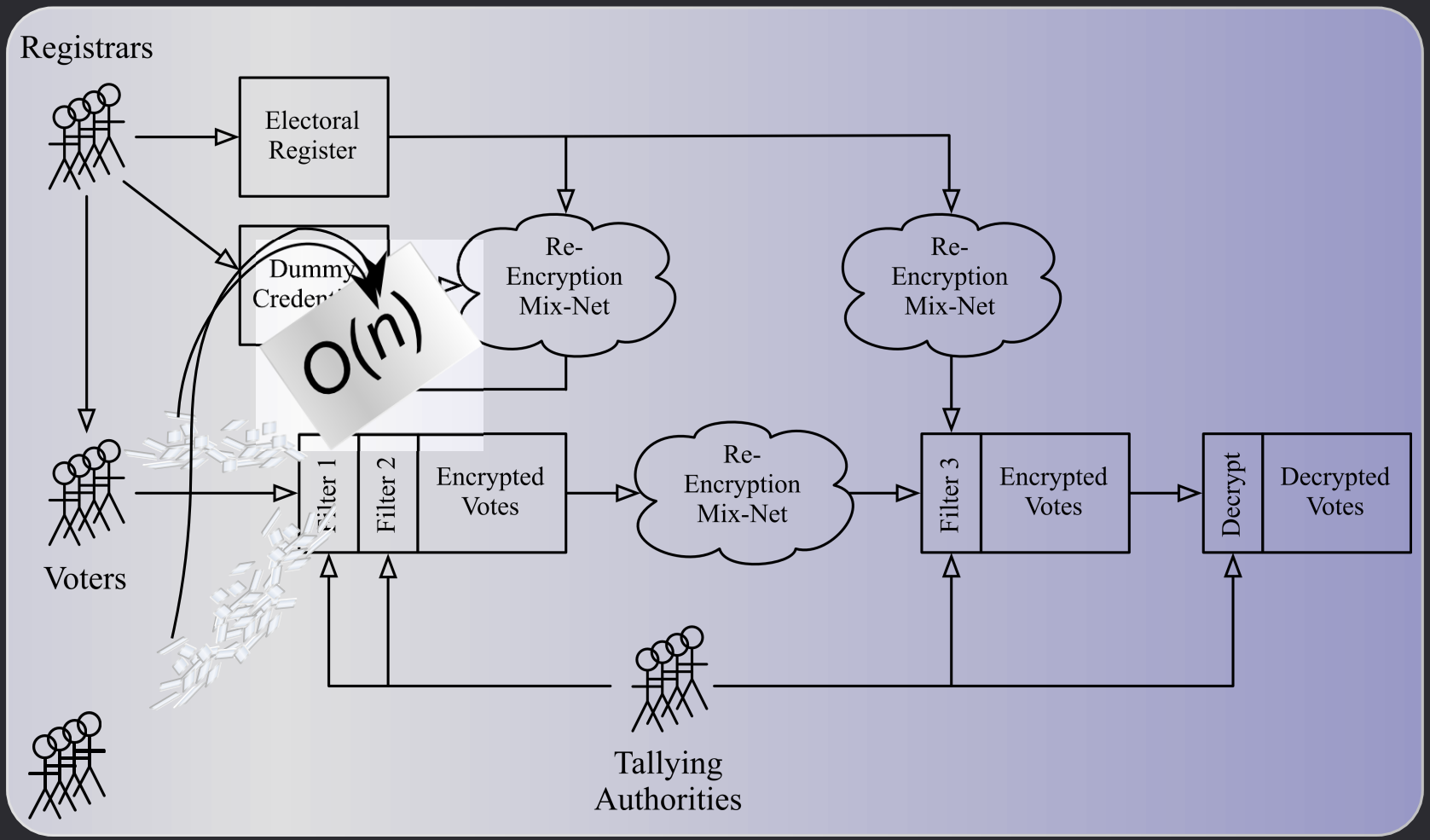
System 'blindly' removes invalid credentials (dummy votes)



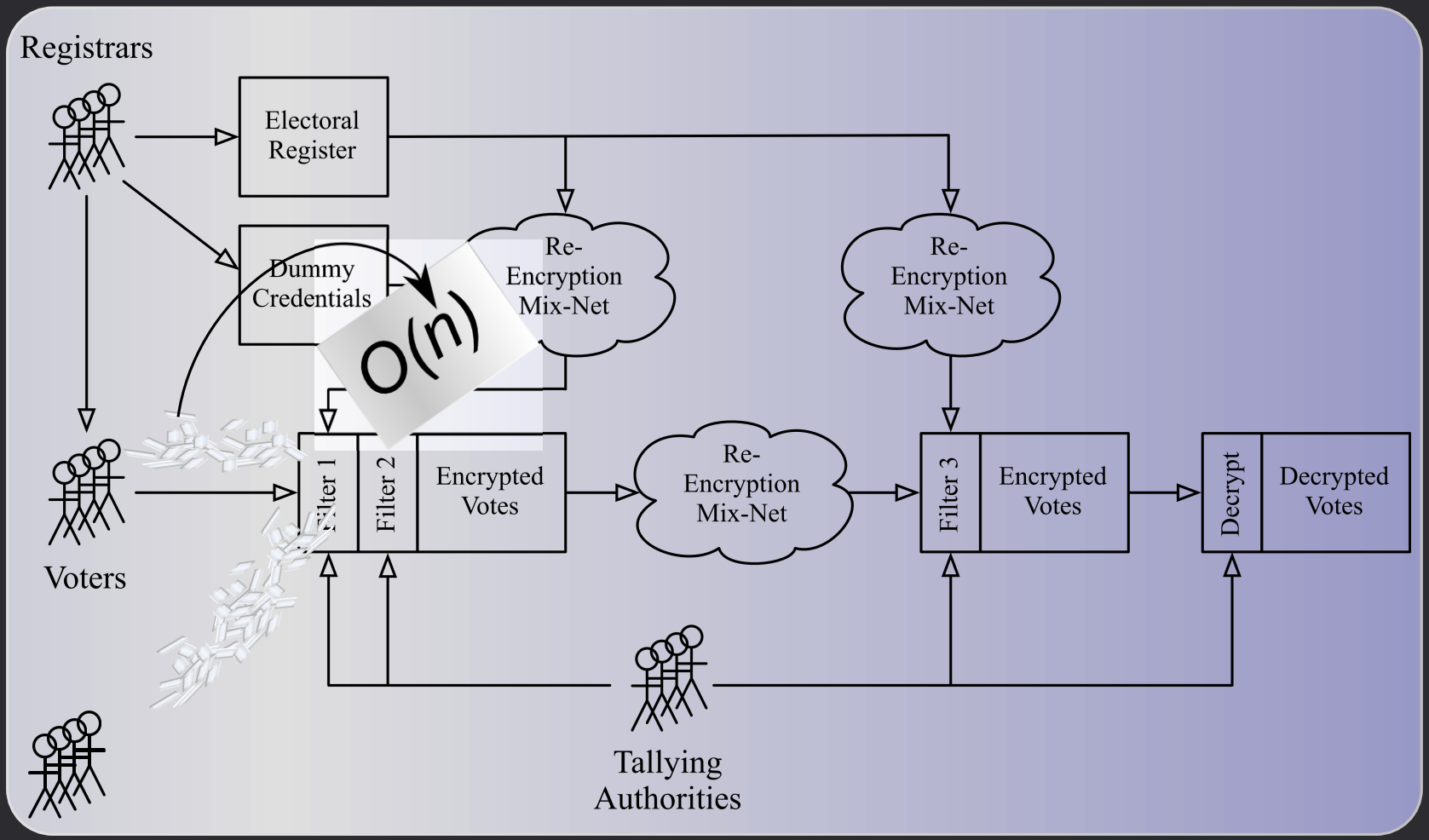
System decrypts remaining votes



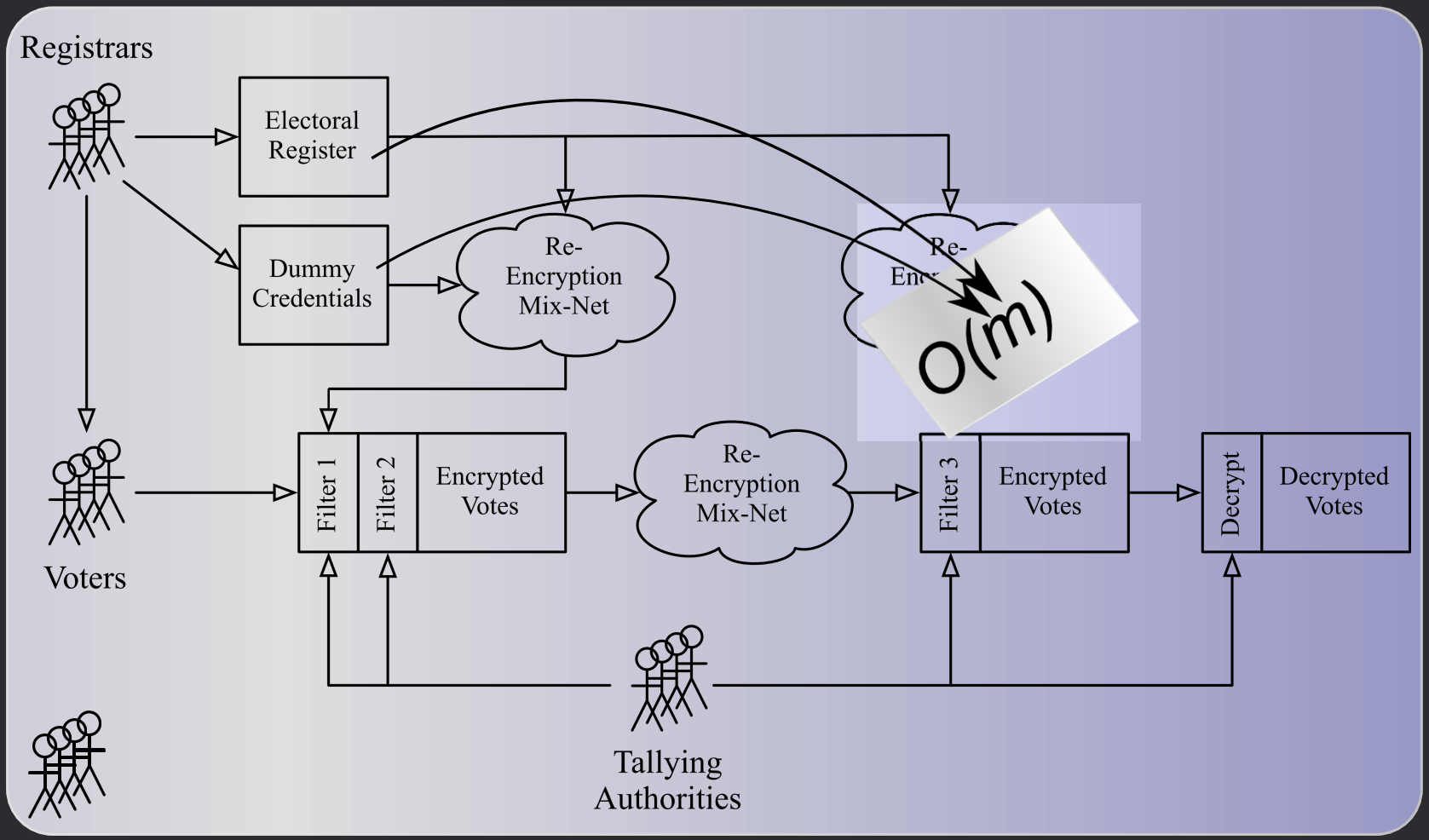
Linear complexity due to Smith/Weber PET



n depends on eligible ballots only



m depends on the predefined amount of $\sigma\&\tau$ credentials



So the question for the system is answered...

We will make it in time, and every filter operates in linear time!

- Test eligibility during vote casting period!
- Test duplicates during vote casting period!
- Decrypt (worst scenario known in advance!)
- Count (worst scenario known in advance!)

Our contribution