Reversing, Breaking, and Fixing the French Legislative Election E-Voting Protocol

Alexandre Debant *and Lucca Hirschi*

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

USENIX Security Symposium
August 11th 2023

Context

+1.5 millions legitimate voters (French citizens resident overseas only)

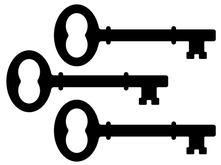
+500 000 ballots cast over the Internet (~77% of all the expressed votes)

11 deputies chosen for 5 years (11 constituencies split in ~200 consulates)

This protocol was based on a new protocol (FLEP), better be sure it is secure!

The different roles

Decryption Trustees



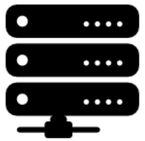
by representatives
and officials

Voter

At home

Voting Client

Javascript running in a browser



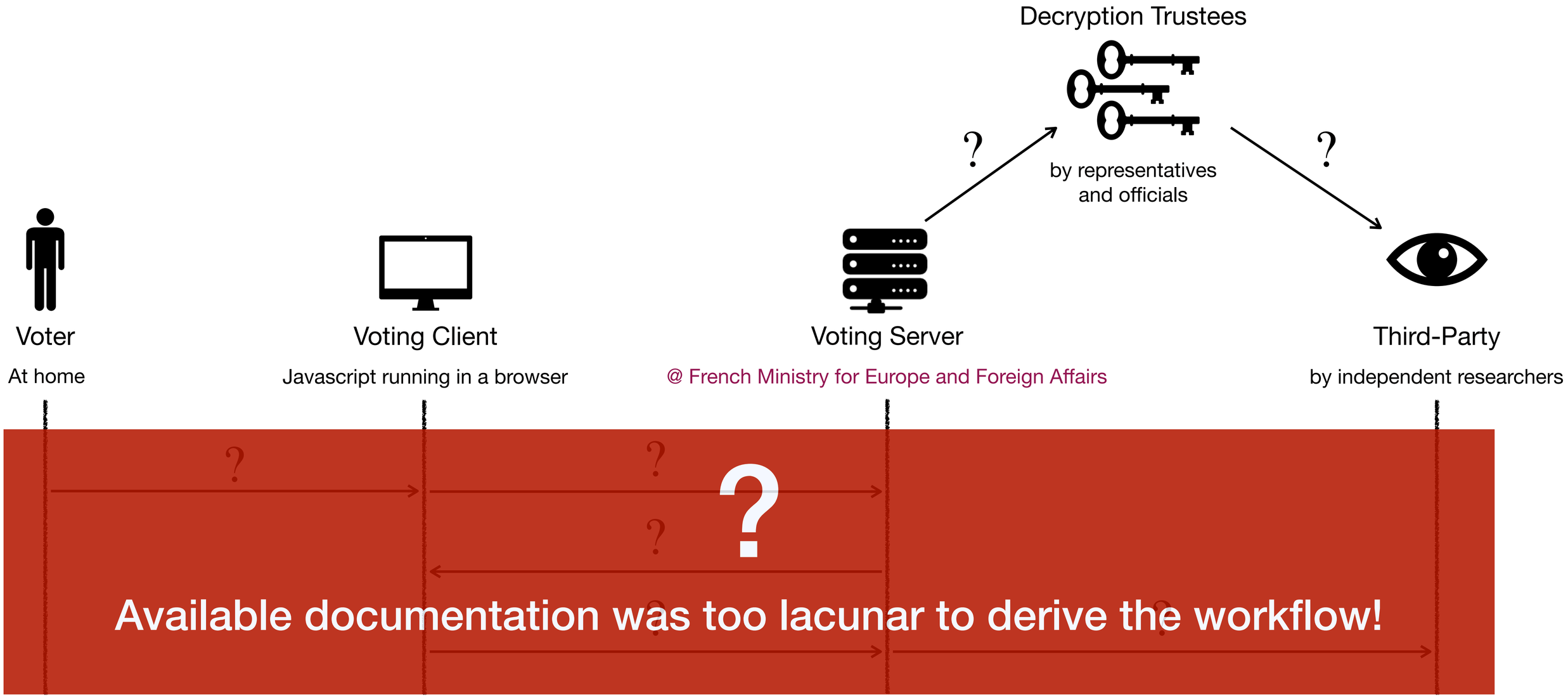
Voting Server

@ French Ministry for Europe and Foreign Affairs

Third-Party

by independent researchers

The different roles



Contributions

First public and comprehensive specification of the protocol
➡ by reversing the obfuscated voting client (Javascript & HTML)

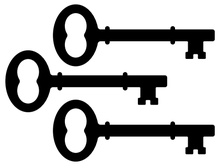
Verifiability and vote secrecy can be attacked by a channel/server attacker:
- design an implementation vulnerabilities
- 6 attack variants

We proposed 6 fixes, most of them implemented for the 2023 elections

Lessons for the organisation of future e-voting elections

The workflow

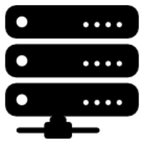
Decryption Trustees



pkD

Voter

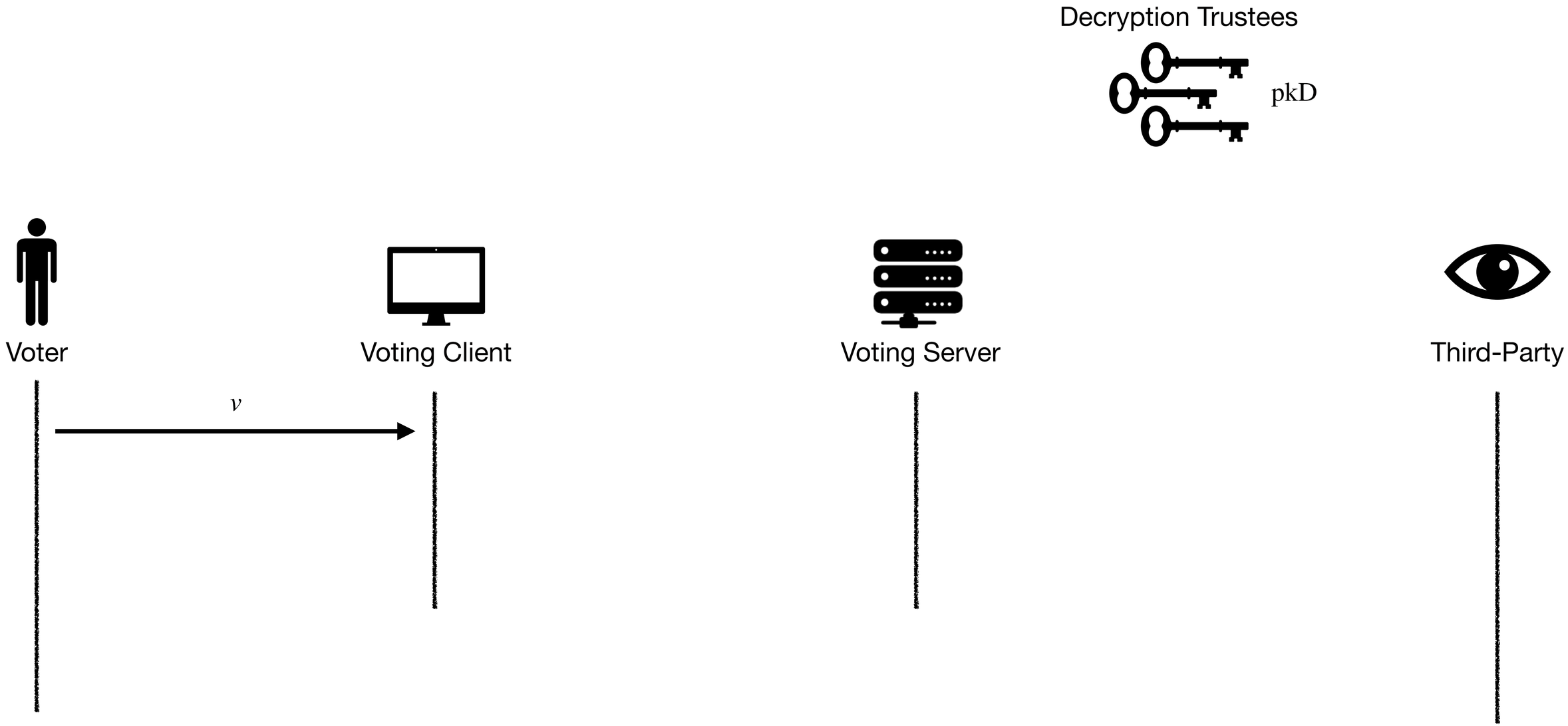
Voting Client



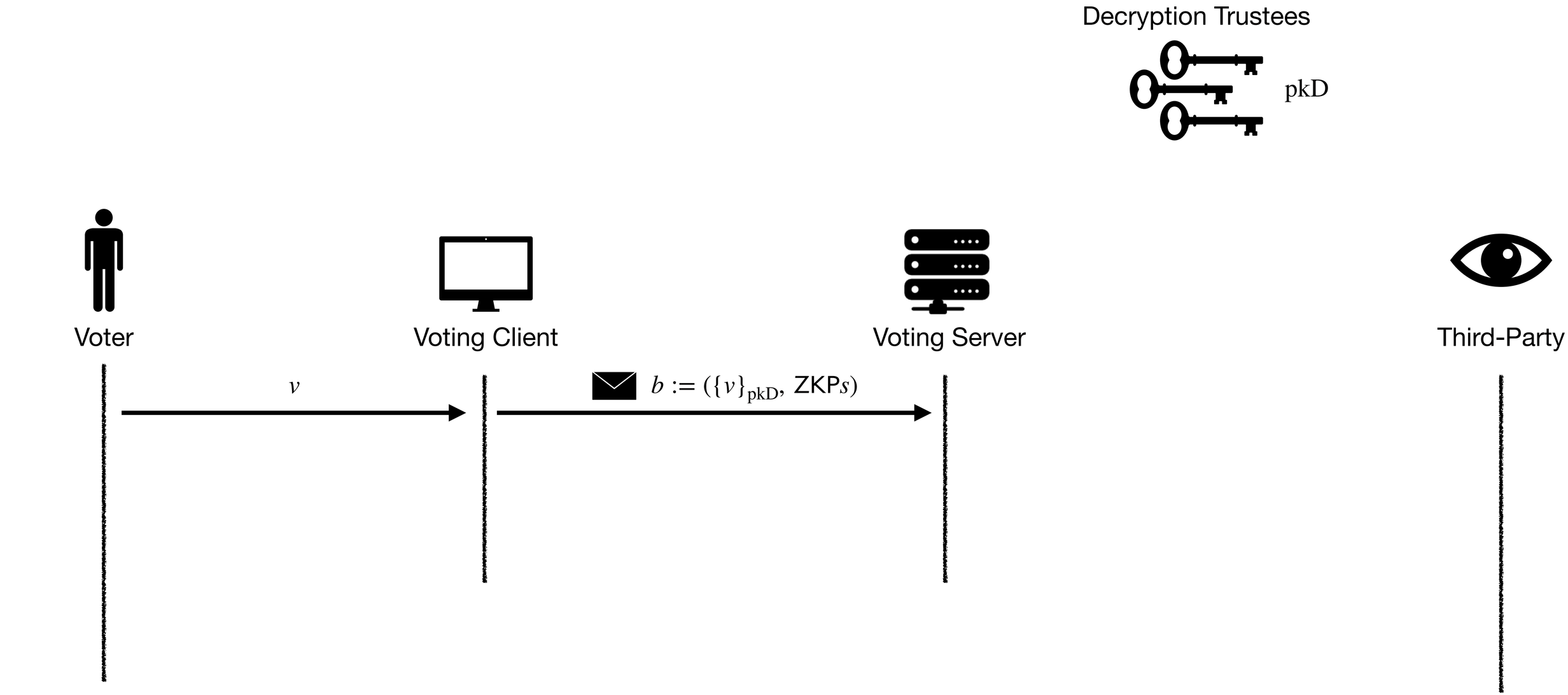
Voting Server

Third-Party

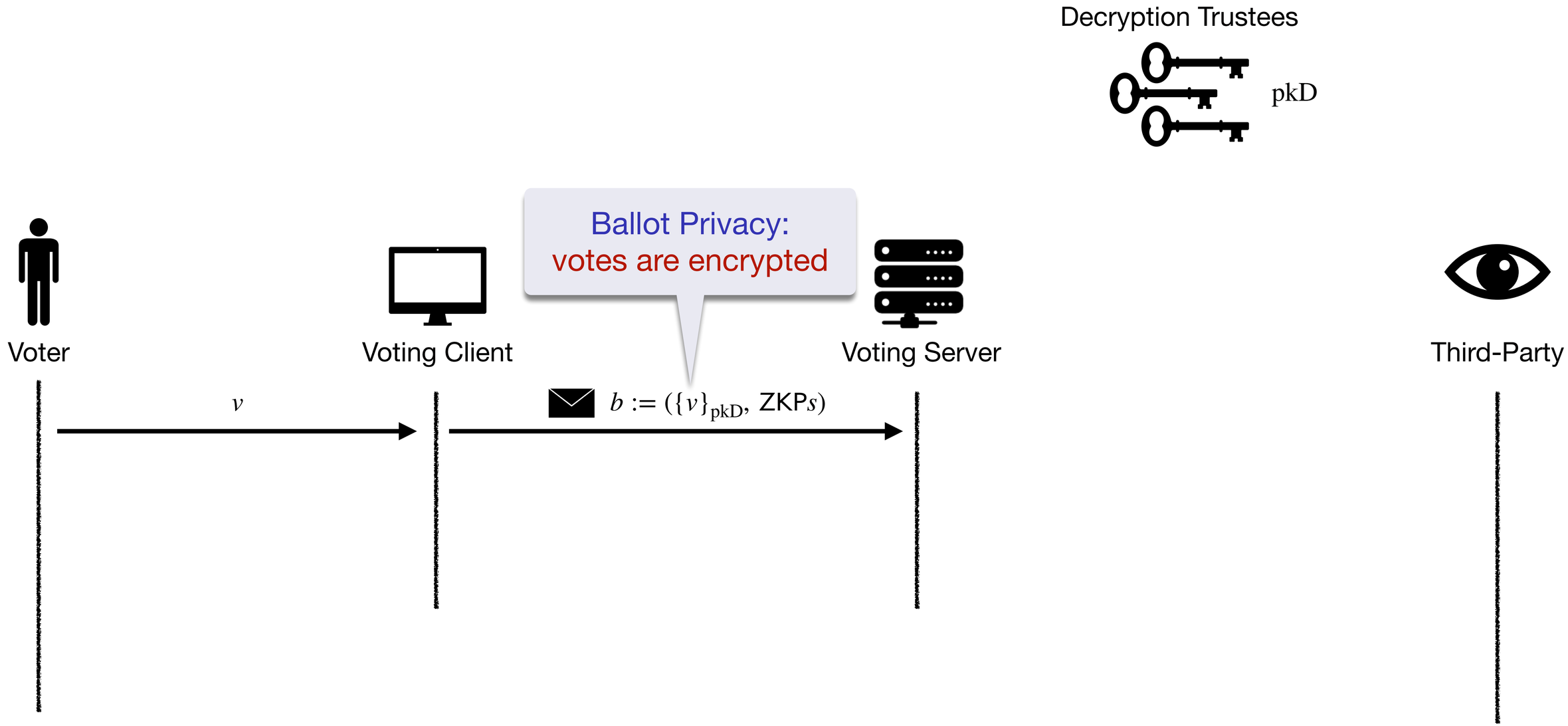
The workflow



The workflow



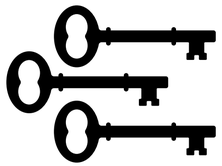
The workflow



The workflow

ballotBox for each consular (~city)

Decryption Trustees

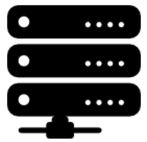


pkD

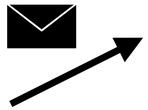
Ballot Privacy:
votes are encrypted

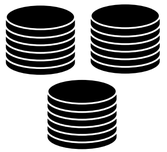
Voter

Voting Client



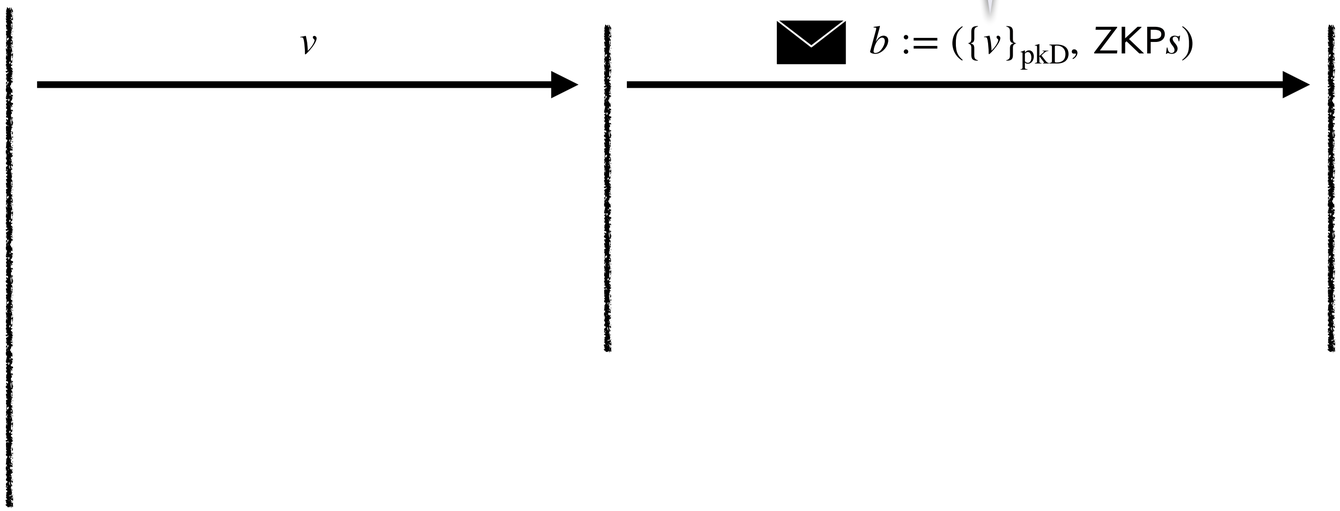
Voting Server





1 per ballotBox

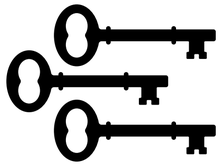
Third-Party



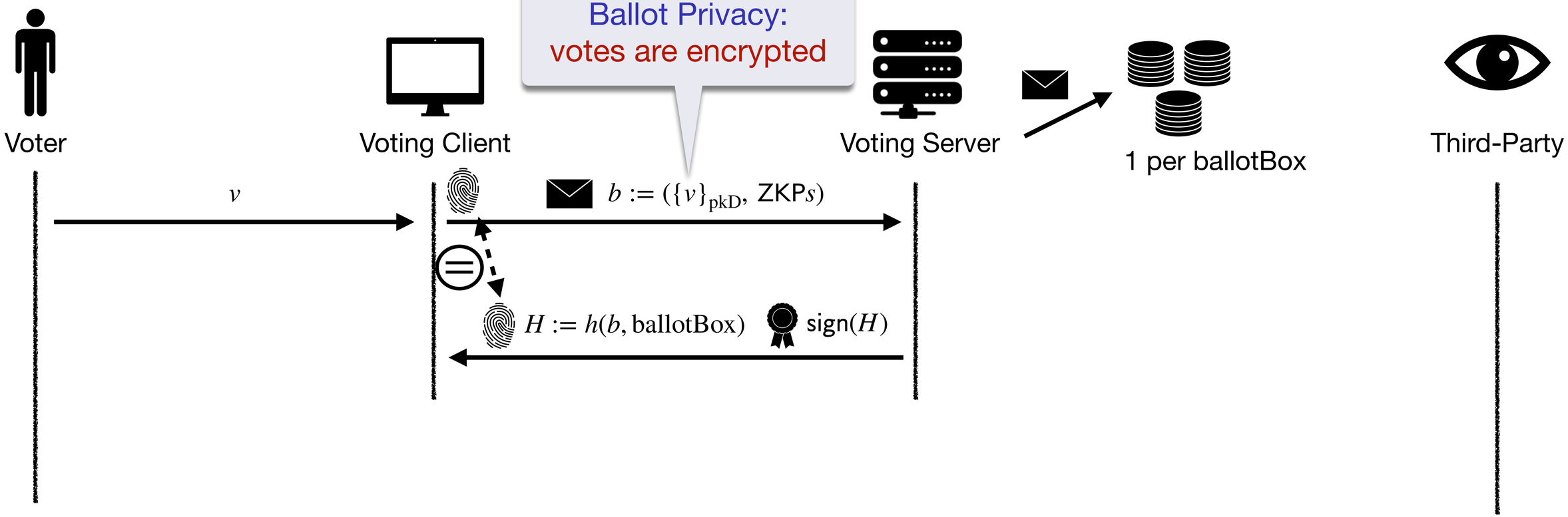
The workflow

ballotBox for each consular (~city)

Decryption Trustees



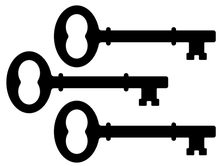
pkD



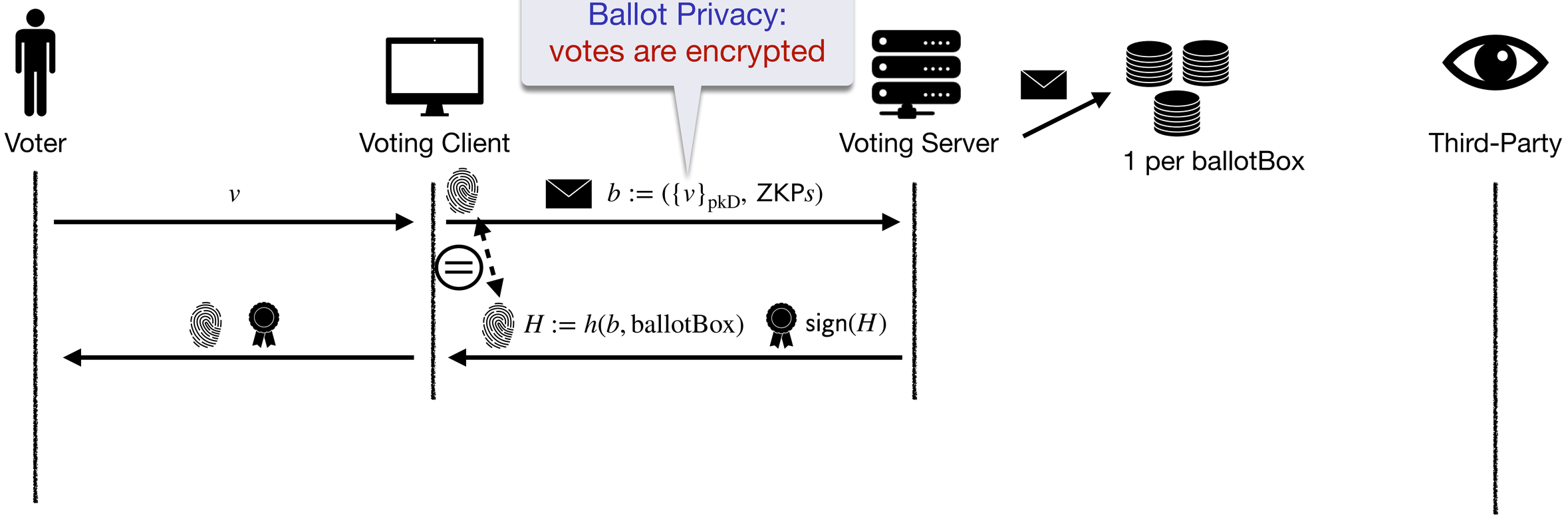
The workflow

ballotBox for each consular (~city)

Decryption Trustees



pkD



The workflow

ballotBox for each consular (~city)

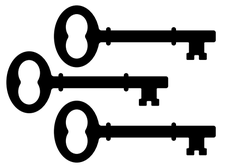
result per ballotBox

The workflow

ballotBox for each consular (~city)

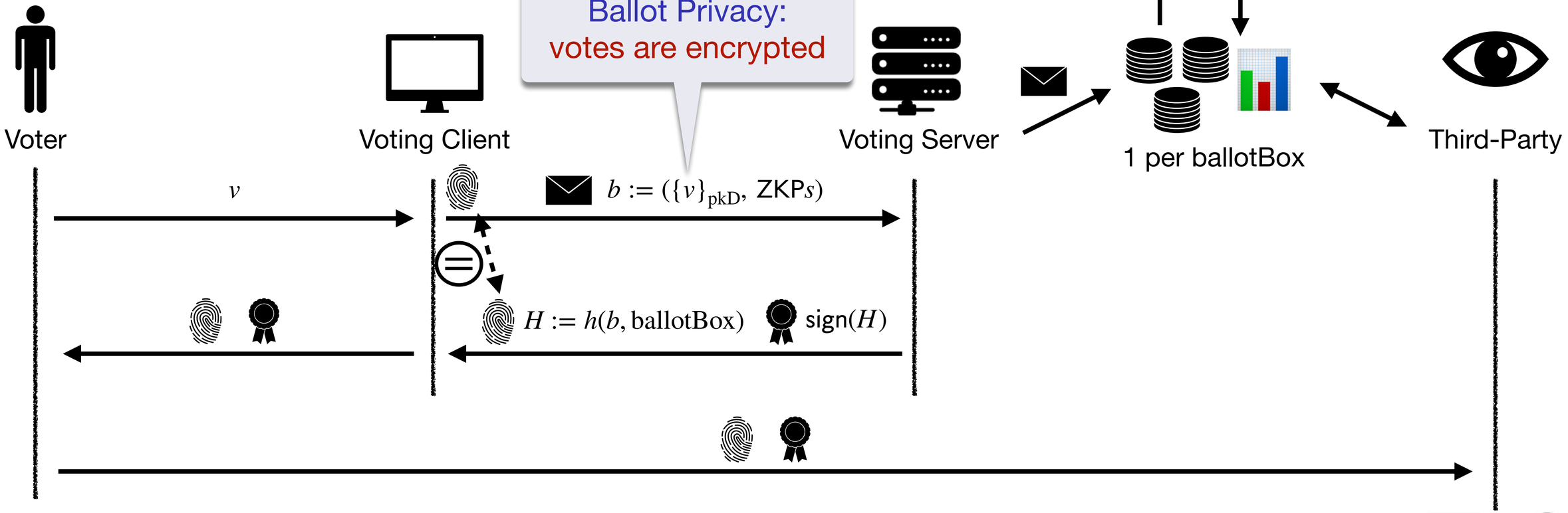
result per ballotBox

Decryption Trustees



pkD

Ballot Privacy:
votes are encrypted

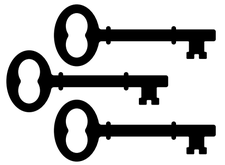


The workflow

ballotBox for each consular (~city)

result per ballotBox

Decryption Trustees

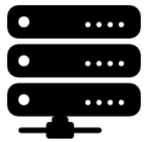


pkD

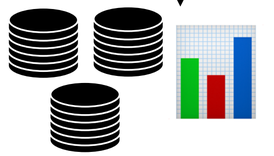
Ballot Privacy:
votes are encrypted

Voter

Voting Client

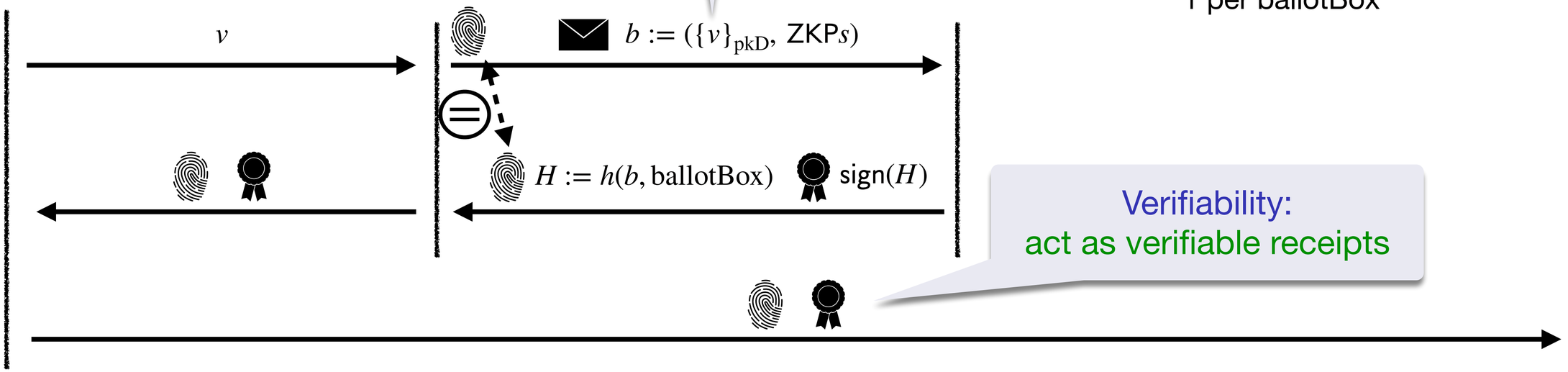


Voting Server



1 per ballotBox

Third-Party



Verifiability:
act as verifiable receipts

Security goals and threat models

Vote secrecy - “No one should know who I voted for” 

Verifiability - “No one can modify the outcome of the election” 

Threat models — security expectations under

	Voter	Voting Client	Communication Channel	Voting Server	Decryption Trustees	Third-Party
Ballot Privacy	😇	😇	😈	😈	😇	😇
Verifiability	😇	😇	😈	😈	😈	😇

Security goals and threat models

Vote secrecy - “No one should know who I voted for”

Verifiability - “No one can modify the outcome of the election”

Cast-as-intended is acknowledge as not satisfied

Threat models — security expectations under

	Voter	Voting Client	Communication Channel	Voting Server	Decryption Trustees	Third-Party
Ballot Privacy	😇	😇	😈	😈	😇	😇
Verifiability	😇	😇	😈	😈	😈	😇

Security goals and threat models

Vote secrecy - “No one should know who I voted for”

Verifiability - “No one can modify the outcome of the election”

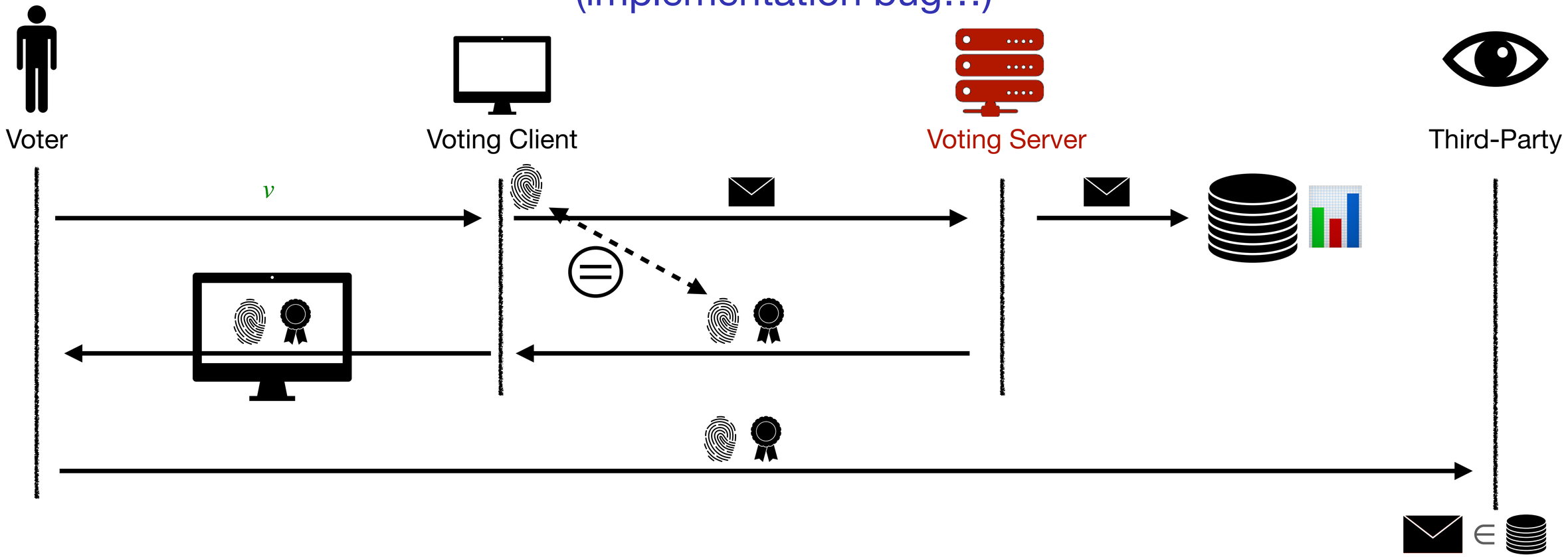
Cast-as-intended is acknowledge as not satisfied

Threat models — attacks under

	Voter	Voting Client	Communication Channel	Voting Server	Decryption Trustees	Third-Party
Ballot Privacy	😇	😇	😈	😇😈	😇	😇
Verifiability	😇	😇	😈	😇😈	😇😈	😇

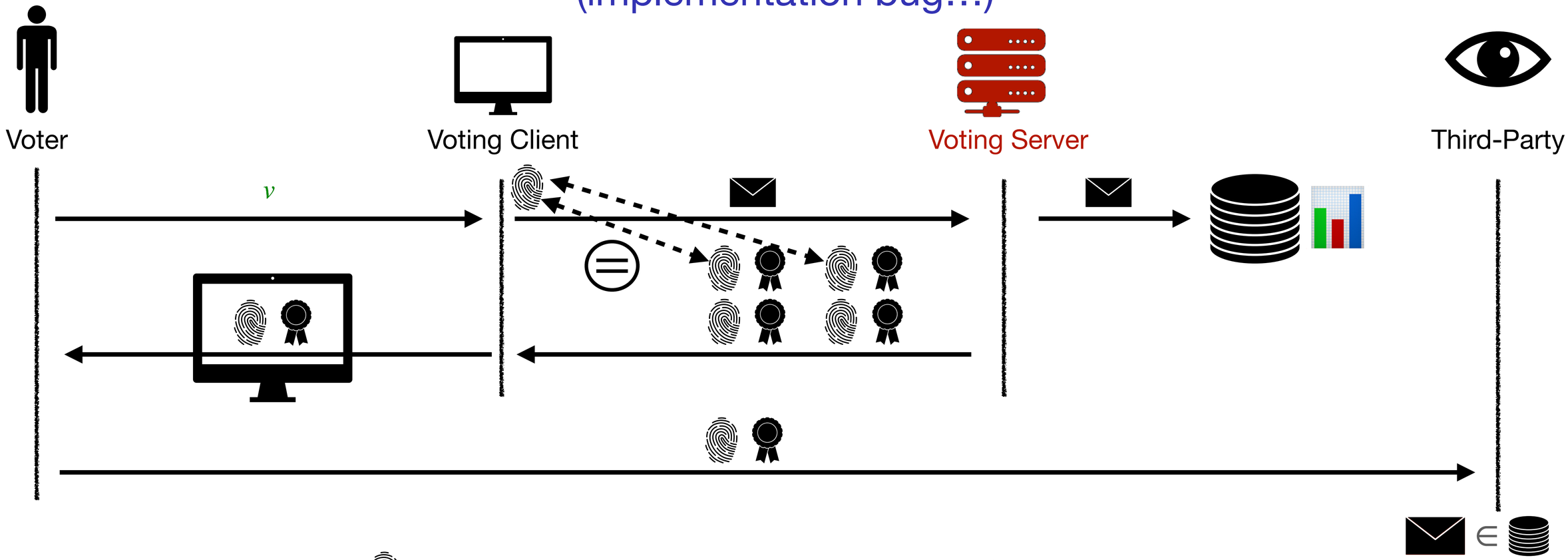
Attack against verifiability

(implementation bug…)



Attack against verifiability

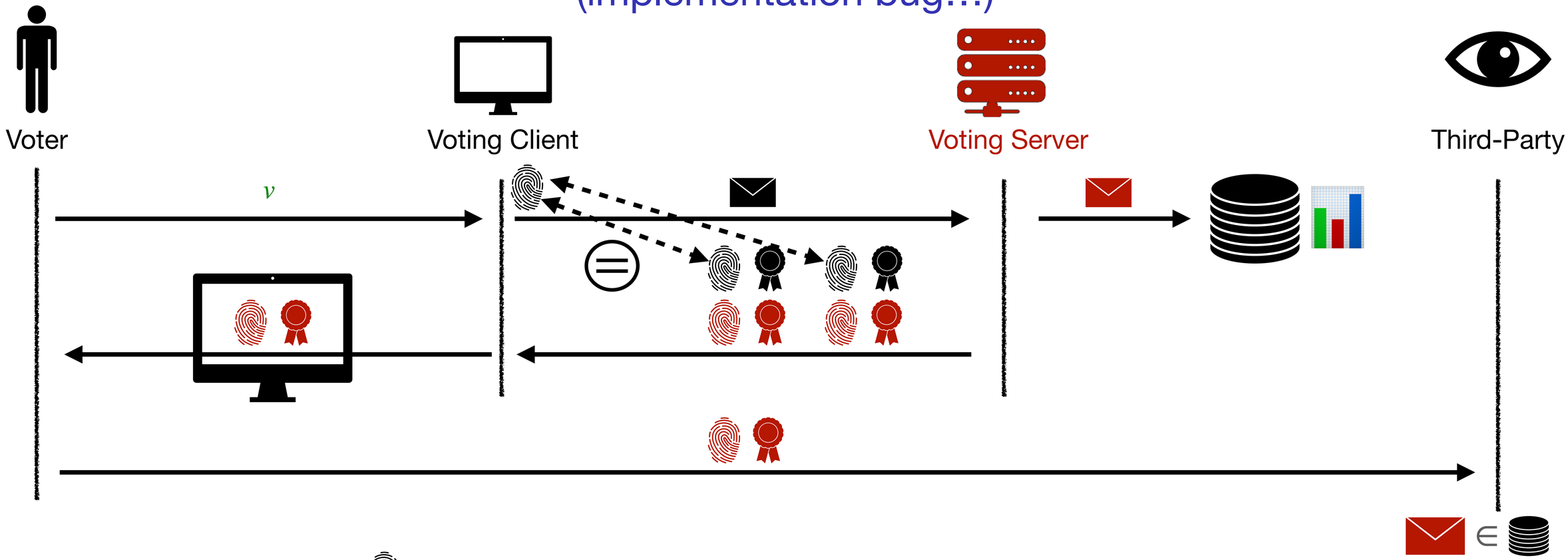
(implementation bug…)



- There are 4 versions of  with various consistency checks in the JavaScript voting client

Attack against verifiability

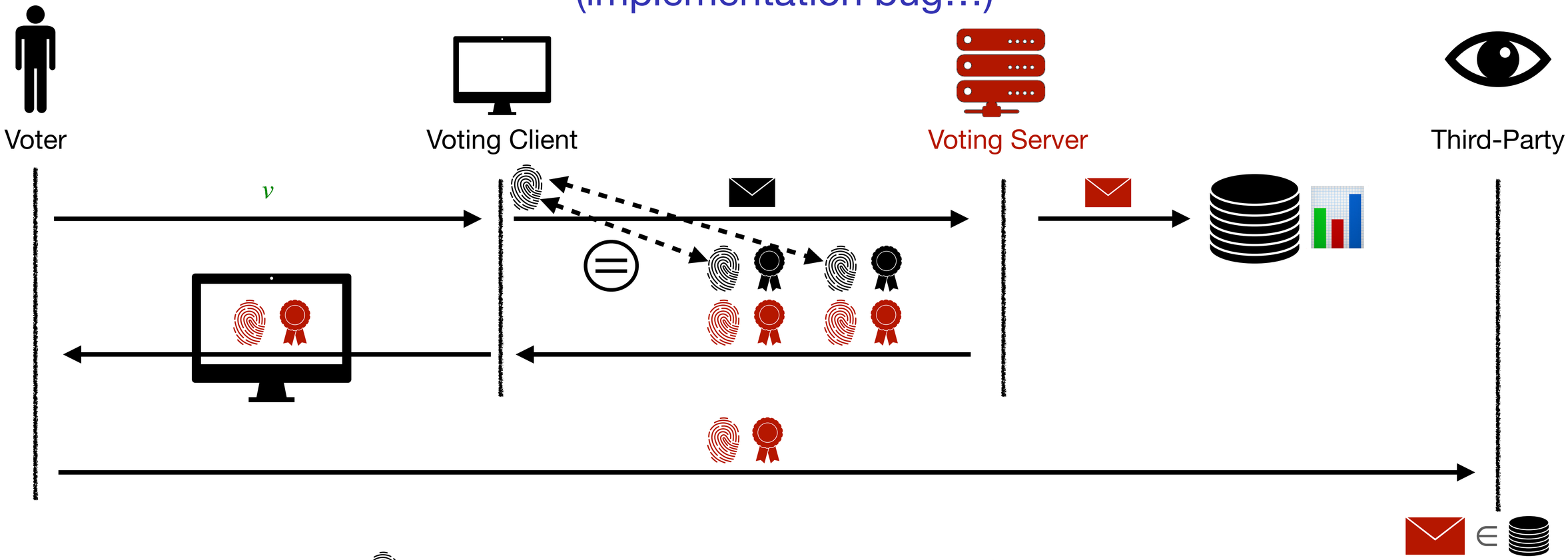
(implementation bug…)



- There are 4 versions of  with various consistency checks in the JavaScript voting client
- **Implementation vulnerability** ⇨ the  actually displayed to the voter can be attacker-controlled

Attack against verifiability

(implementation bug…)



- There are 4 versions of  with various consistency checks in the JavaScript voting client
- **Implementation vulnerability** ⇨ the  actually displayed to the voter can be attacker-controlled

Impact: channel or server attacker can **stealthily modify the outcome by replacing or dropping ballots**

Attack against vote privacy

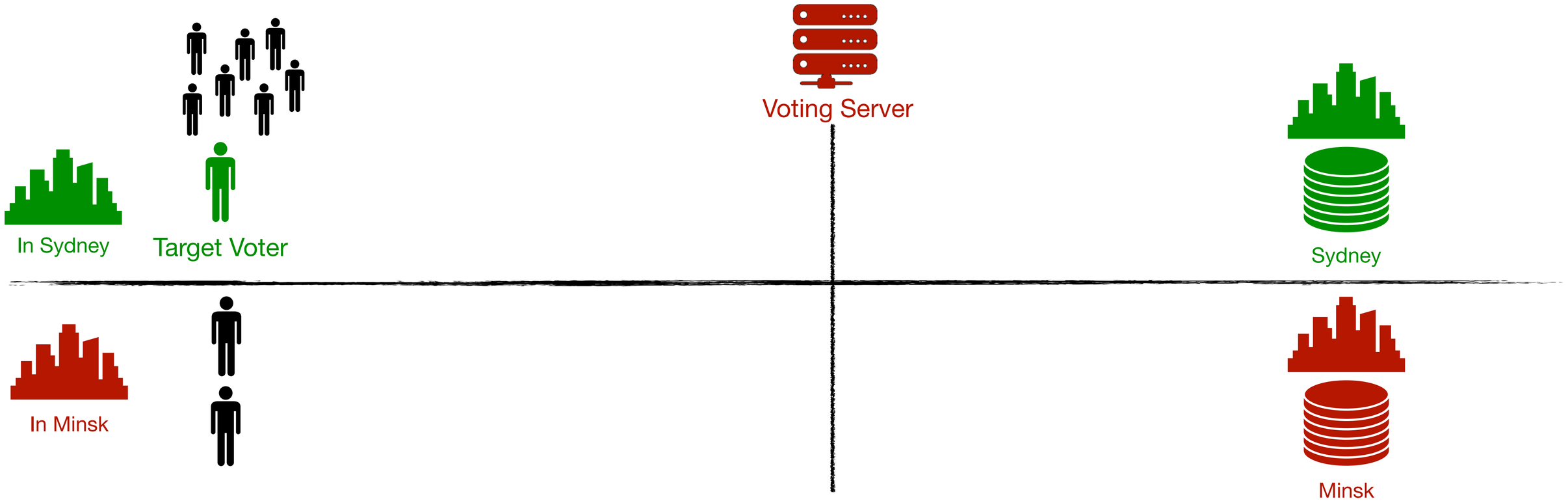
(design vulnerability…)

- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox

Attack against vote privacy

(design vulnerability…)

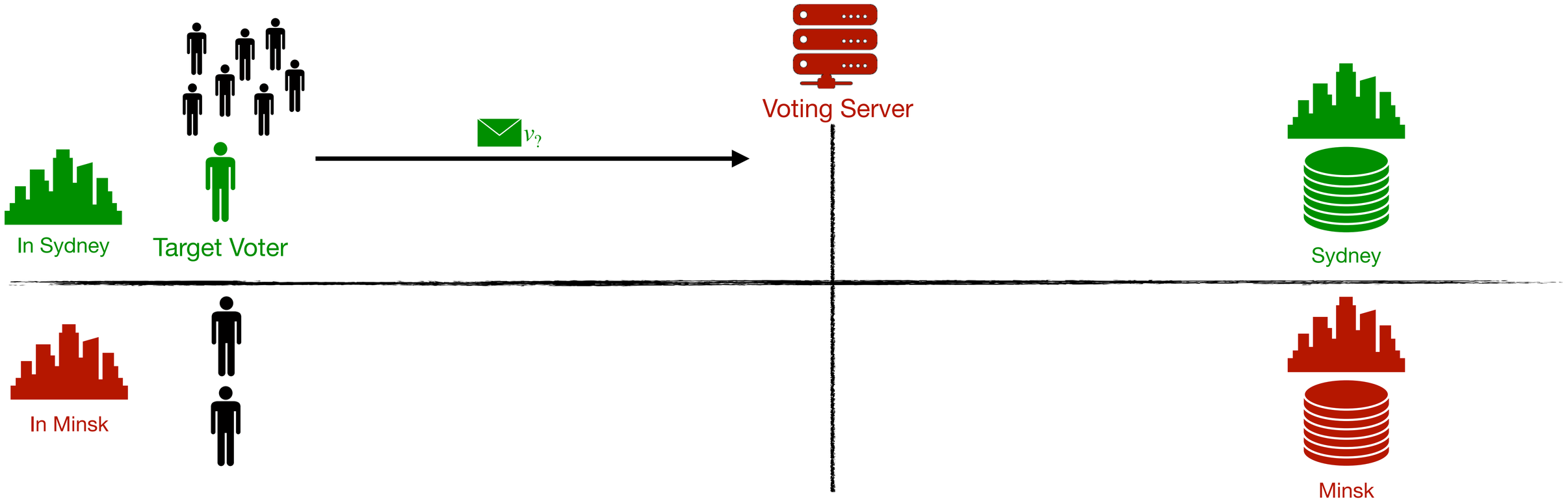
- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Attack against vote privacy

(design vulnerability…)

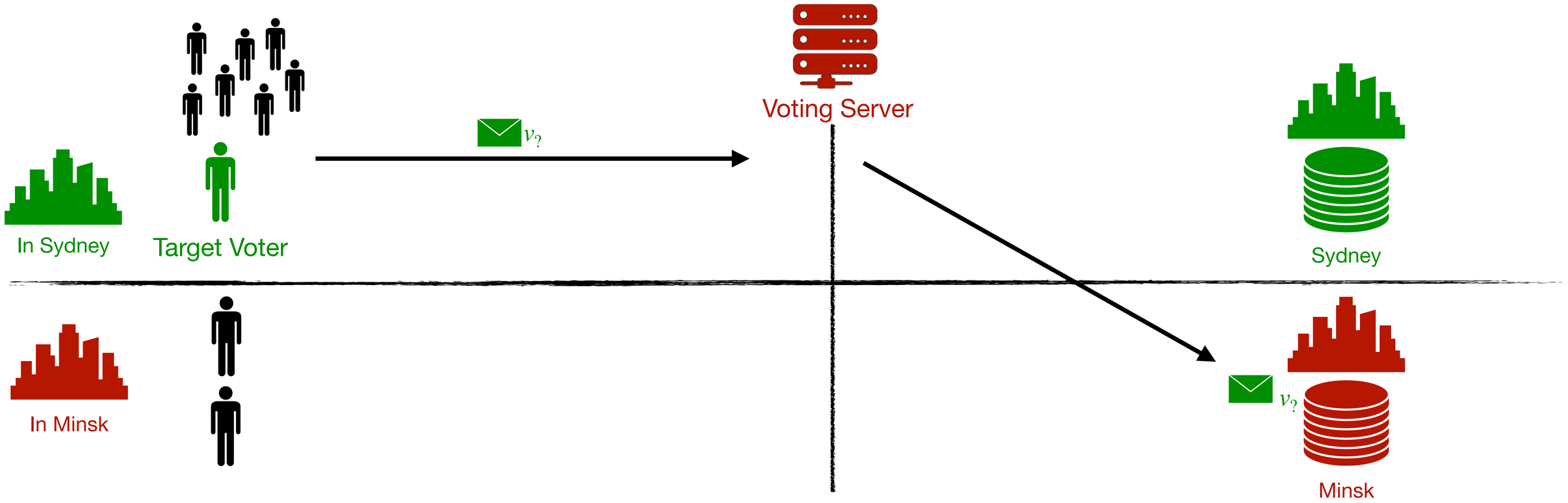
- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Attack against vote privacy

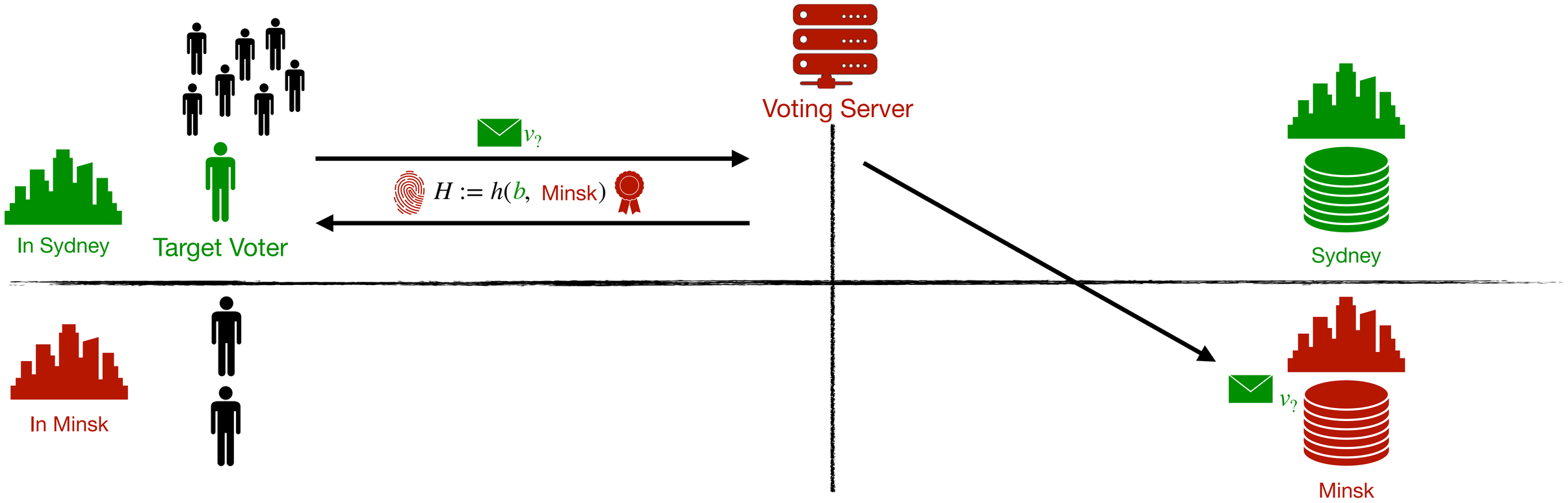
(design vulnerability…)

- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Attack against vote privacy
(design vulnerability…)

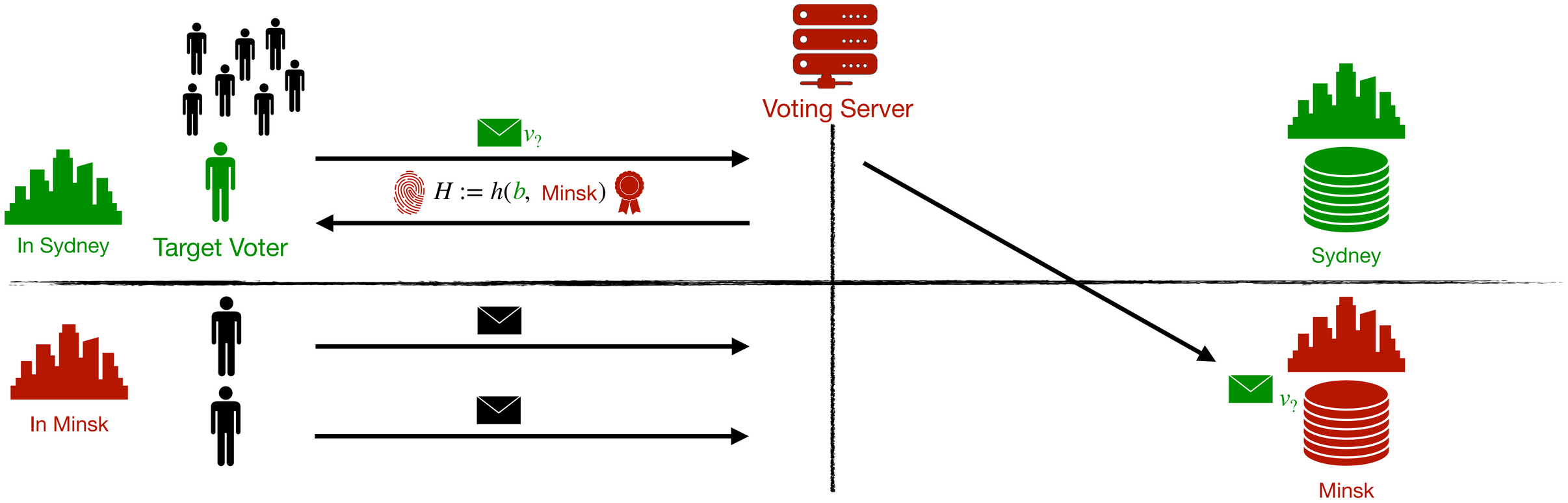
- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Attack against vote privacy

(design vulnerability…)

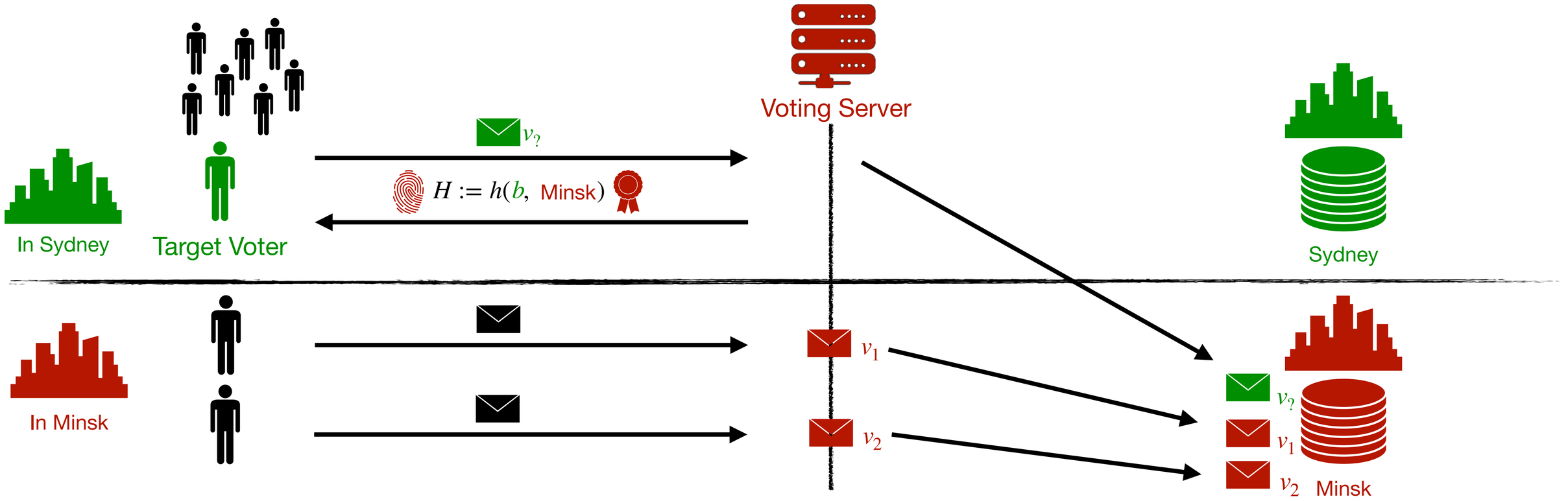
- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Attack against vote privacy

(design vulnerability…)

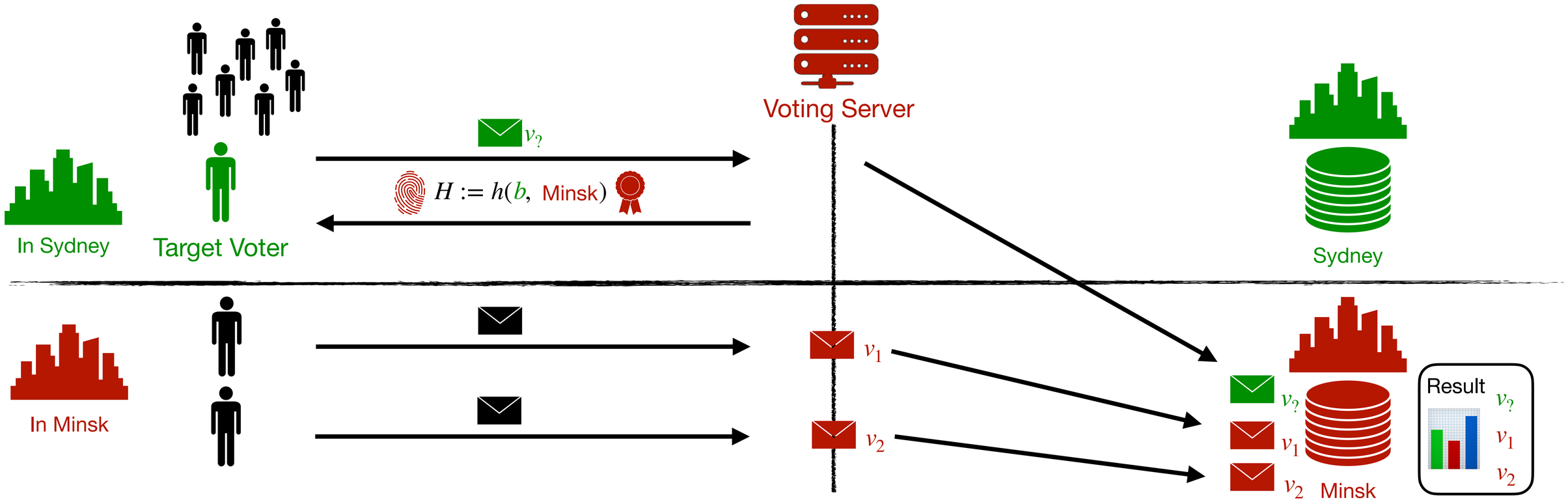
- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Attack against vote privacy

(design vulnerability…)

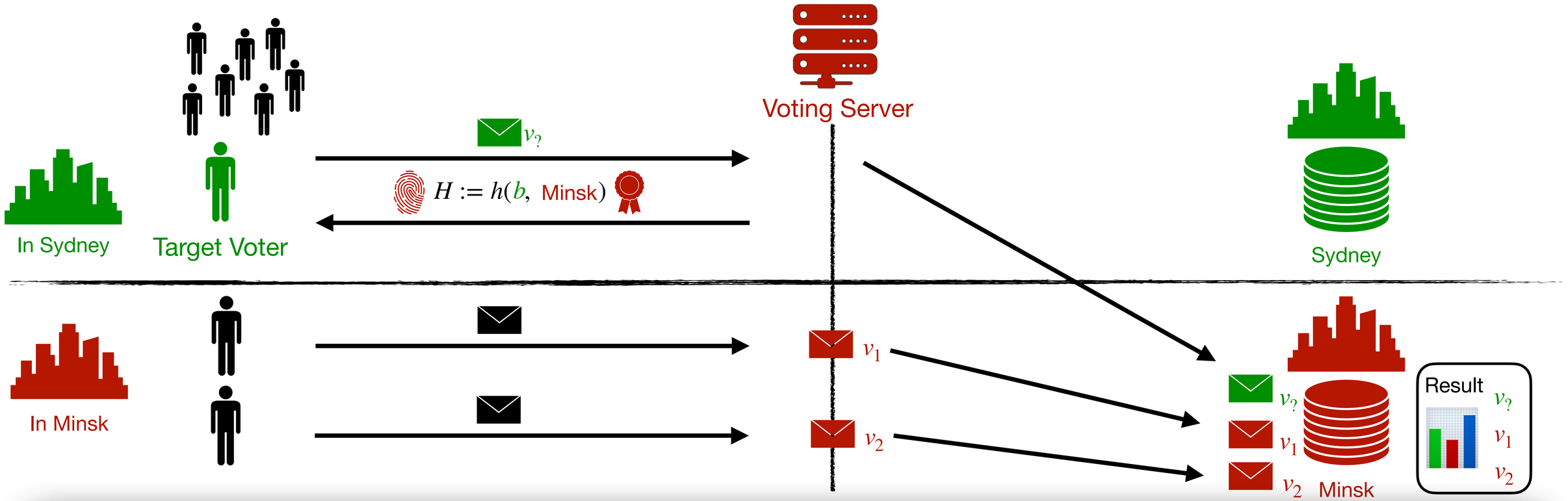
- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Attack against vote privacy

(design vulnerability…)

- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Impact: channel or server attacker can **stealthily learn some target voters' vote** (and perform remote coercion)

Fixes for future elections

We proposed 6 fixes and notably:

1. Display and check  instead of 
2. Binds ballotBox to the ballot ZKPs
3. Third-Party checks ballotBox

(Attacks and fixes were responsibly disclosed to the vendor and stakeholders.)
Special thanks to the ANSSI who have been proactive in this process.

Fixes for future elections

We proposed 6 fixes and notably:

1. Display and check  instead of  ✔/✘ partially done for 2023 election
2. Binds ballotBox to the ballot ZKPs ✔ already implemented for 2023
3. Third-Party checks ballotBox ✔ already implemented for 2023

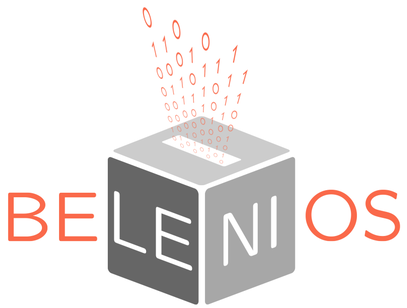
(Attacks and fixes were responsibly disclosed to the vendor and stakeholders.)
Special thanks to the ANSSI who have been proactive in this process.

Lessons learned

(recommandations and research questions)

Lessons learned

(recommandations and research questions)



State-of-art protocol

affected by none of the attacks

Adapt the design

FLEP Protocol

Implement, Deploy, Audit

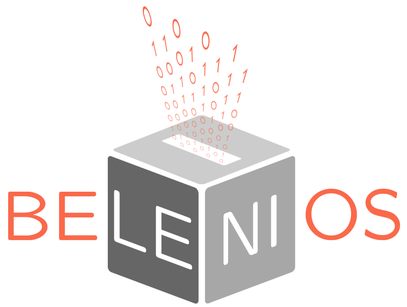
2022 Election

FLEP 2022

affected by 6 attacks

+ other concerns
not discussed here

Lessons learned
(recommandations and research questions)



State-of-art protocol

affected by none of the attacks

Adapt the design

FLEP Protocol

Implement, Deploy, Audit

2022 Election

FLEP 2022

affected by 6 attacks

+ other concerns
not discussed here

How can this happen? 🤔

Lessons learned
(recommandations and research questions)

1: Adapt the design:

➡ state-of-the art solutions lack features
- multi-ballot-box for announcing fine-grain results
- downloadable receipts

➡ state-of-the-art solutions propose unpractical features
- voters authentication currently relies on a single-point-of-trust

Lessons learned
(recommandations and research questions)

1: Adapt the design:

➡ state-of-the art solutions lack features
- multi-ballot-box for announcing fine-grain results
- downloadable receipts

➡ state-of-the-art solutions propose unpractical features
- voters authentication currently relies on a single-point-of-trust

Academic papers should take into account **operational constraints**

Lessons learned
(recommandations and research questions)

1: Adapt the design:

➡ state-of-the art solutions lack features
- multi-ballot-box for announcing fine-grain results
- downloadable receipts

➡ state-of-the-art solutions propose unpractical features
- voters authentication currently relies on a single-point-of-trust

Academic papers should take into account **operational constraints**

2: Implement, deploy, audit

➡ transparency and openness
- clear security objectives and threat models
- open specification, promote public scrutiny (e.g. as in Switzerland)

➡ identify the (most) critical components, e.g. Voting client > Server
- make it auditable (specification, open source, etc)
- make it monitorable

Lessons learned
(recommandations and research questions)

1: Adapt the design:

➡ state-of-the art solutions lack features
- multi-ballot-box for announcing fine-grain results
- downloadable receipts

➡ state-of-the-art solutions propose unpractical features
- voters authentication currently relies on a single-point-of-trust

Academic papers should take into account **operational constraints**

2: Implement, deploy, audit

➡ transparency and openness
- clear security objectives and threat models
- open specification, promote public scrutiny (e.g. as in Switzerland)

➡ identify the (most) critical components, e.g. Voting client > Server
- make it auditable (specification, open source, etc)
- make it monitorable

Any component that **needs to be trusted** is **critical**

Conclusion

☞ https://eprint.iacr.org/2022/1653

First public and comprehensive specification of the protocol

Verifiability and vote secrecy can be attacked by a channel/server attacker:

- design an implementation vulnerabilities
- 6 attack variants

We proposed 6 fixes, most of them implemented for the 2023 elections

Lessons for future e-voting elections