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

Alexandre Debant *and Lucca Hirschi*

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

The Computers, Privacy and Data Protection conference (CPDP)
Brussels, May 23rd 2024

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Inria

FRANCE 2030

Context

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

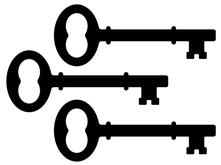
+500 000 ballots cast over the Internet (~77% of all the expressed votes)

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



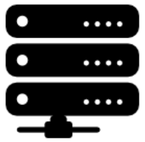
by representatives
and officials

Voter

At home

Voting Client

Javascript running in a browser



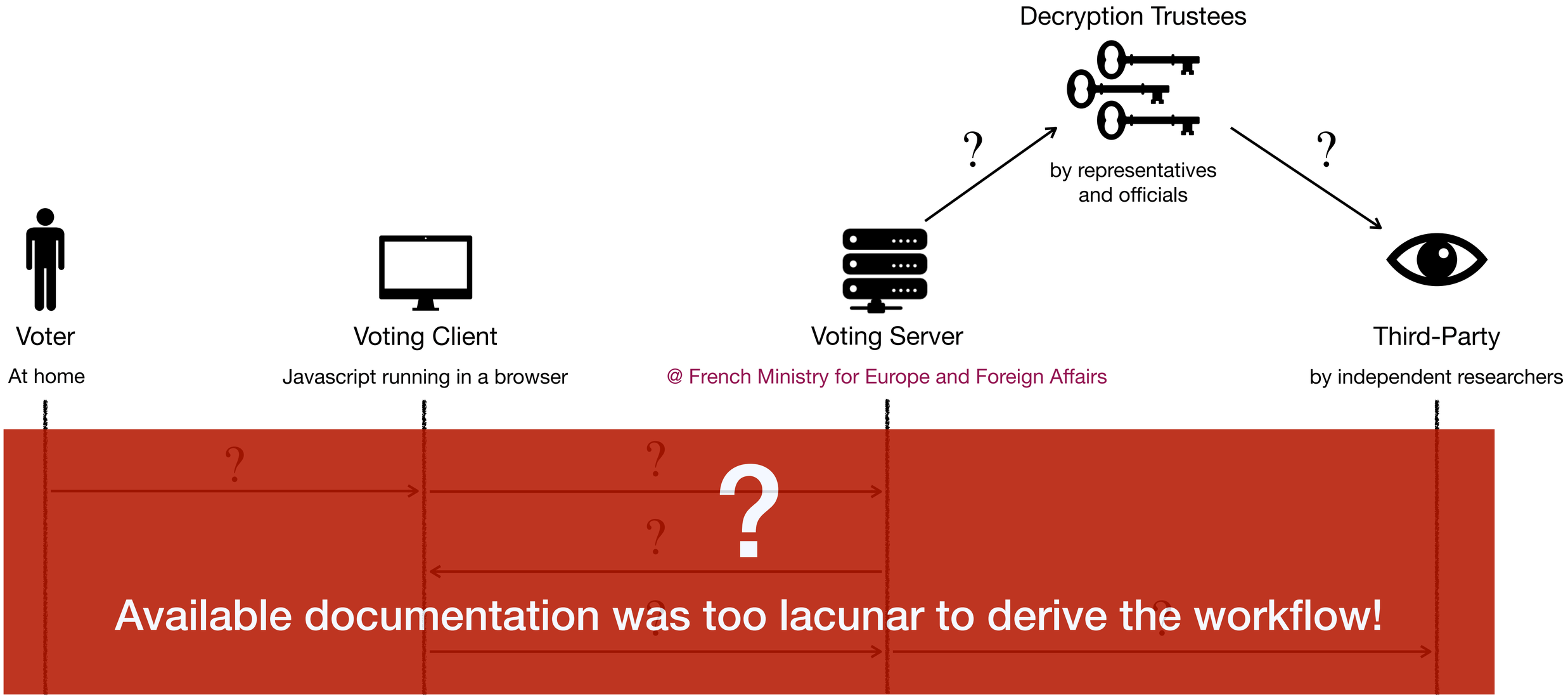
Voting Server

@ 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)

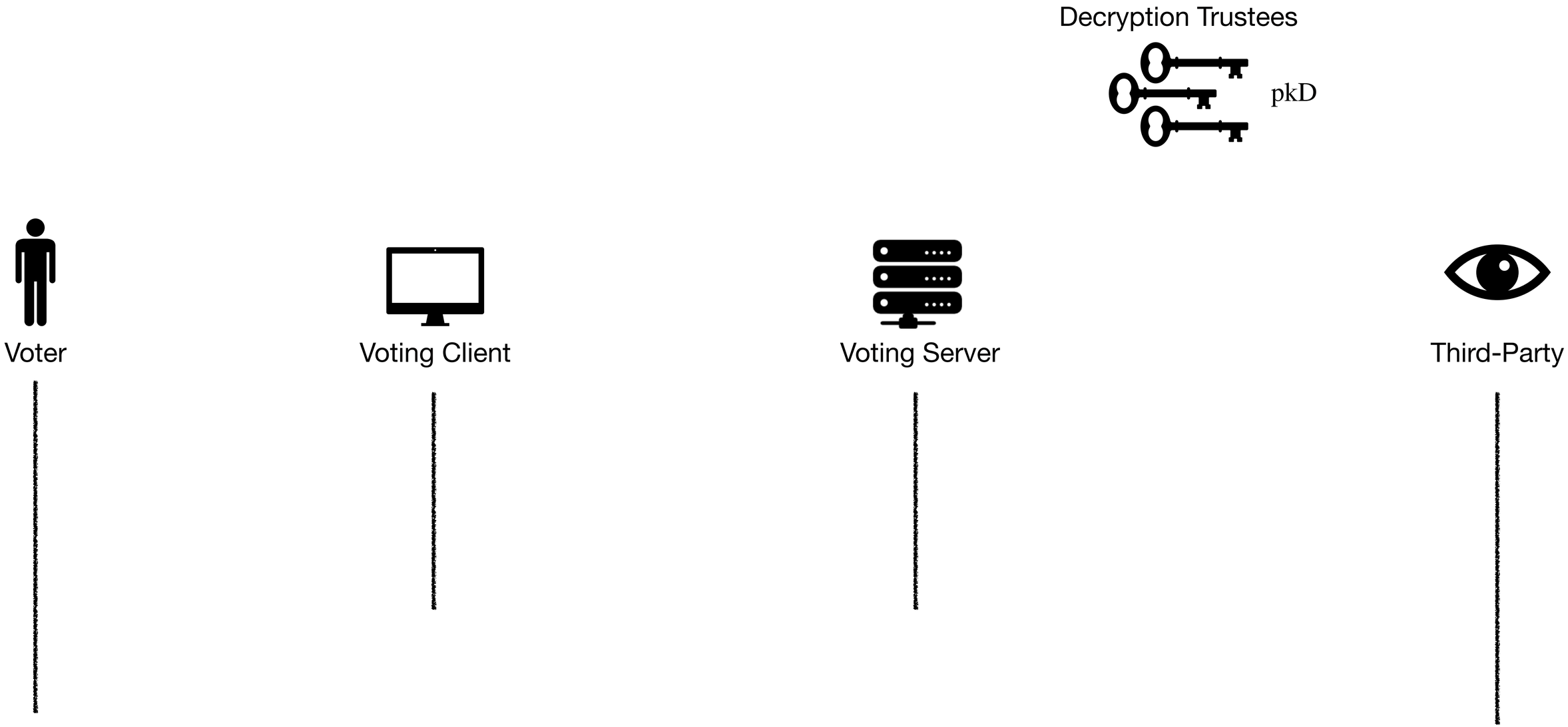
Verifiability and vote privacy can be attacked by a channel/server attacker:

- design an implementation vulnerabilities
- 6 attack variants

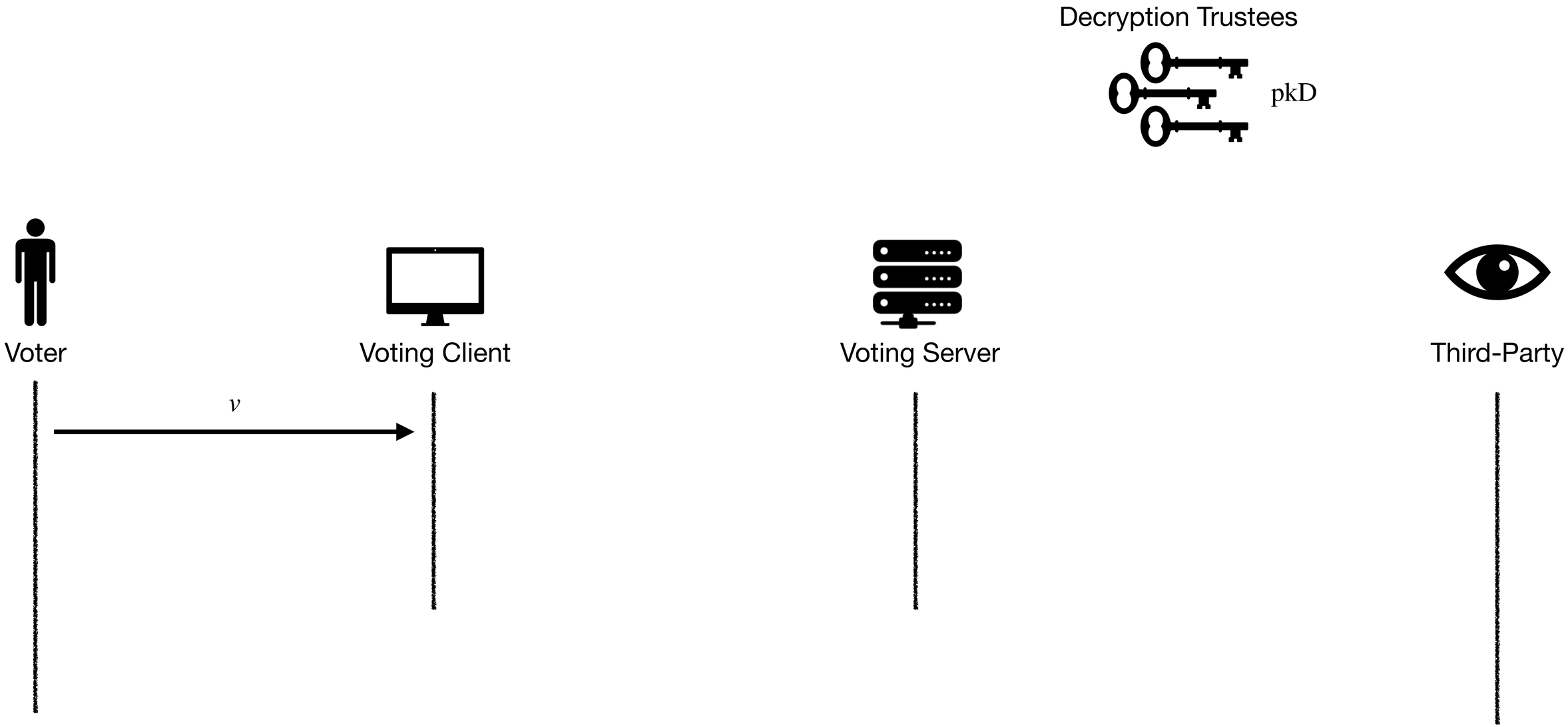
We proposed 6 fixes, most of them implemented for the 2023 elections

Lessons for the organisation of future e-voting elections

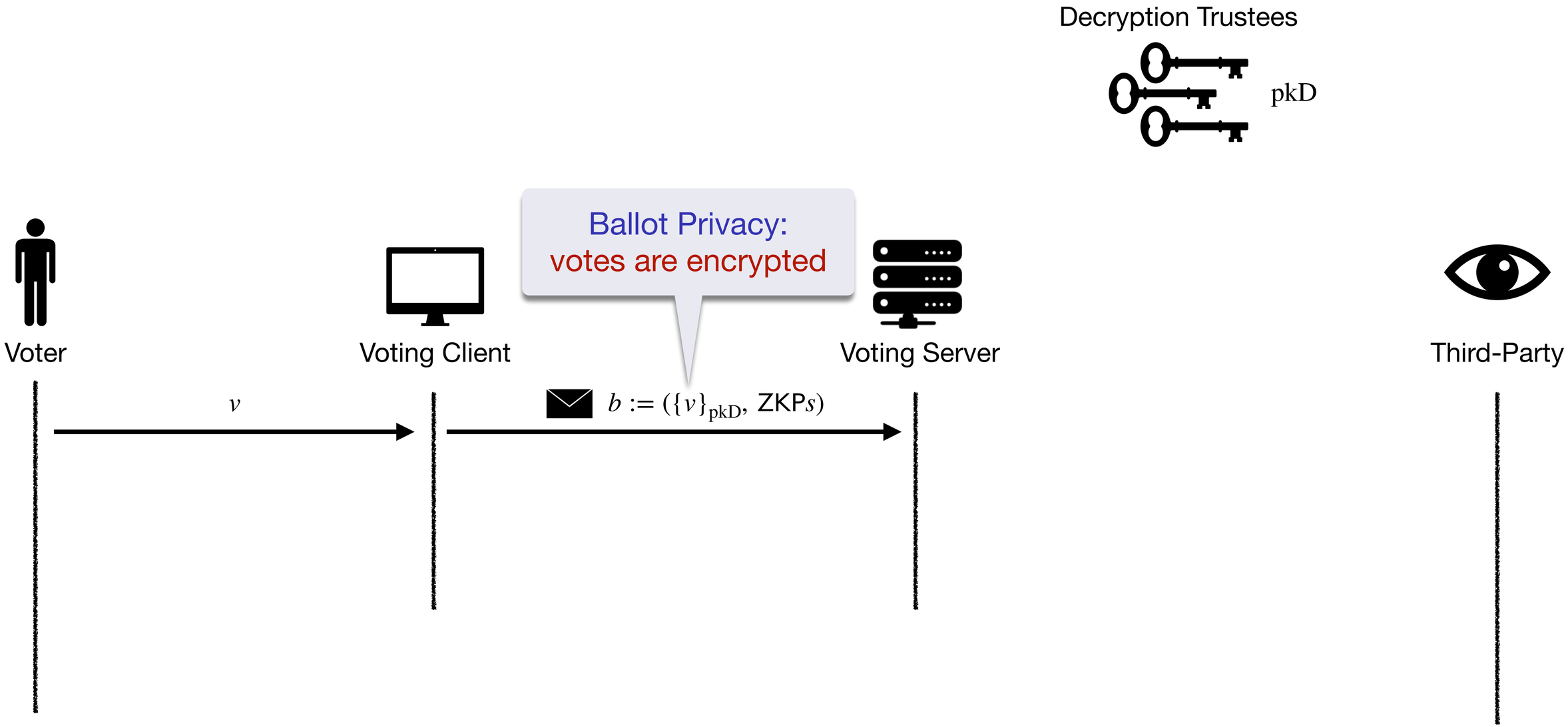
The workflow



The workflow



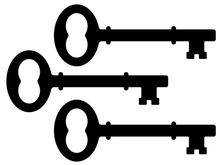
The workflow



The workflow

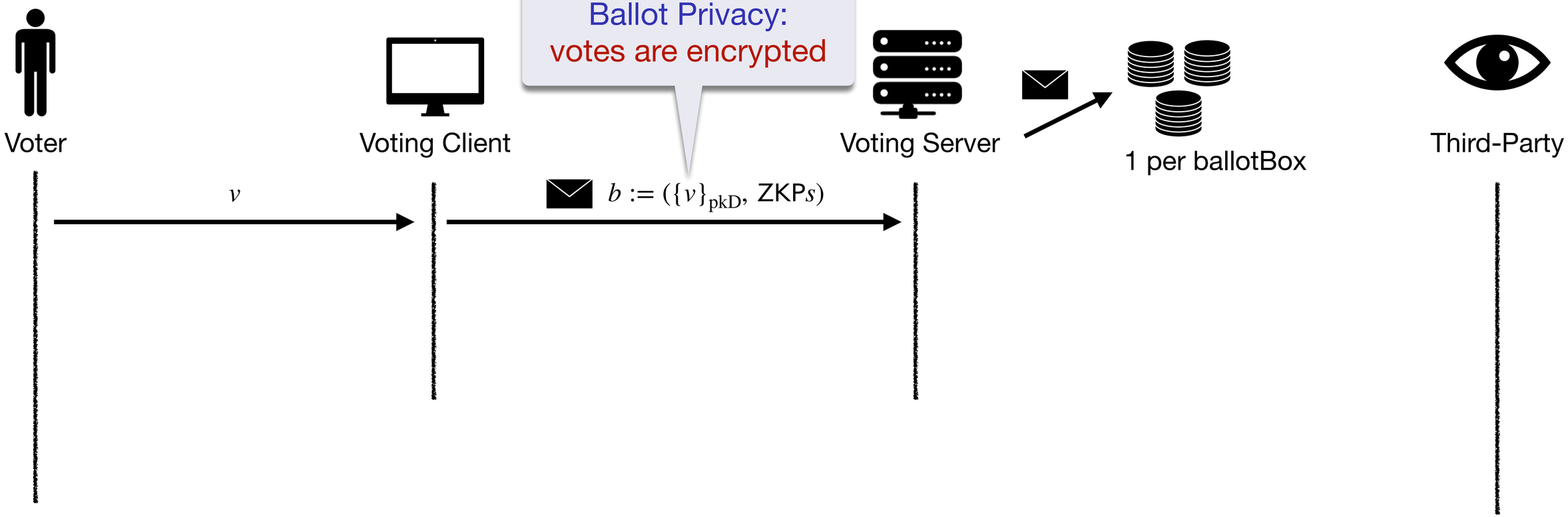
ballotBox for each consular (~city)

Decryption Trustees



pkD

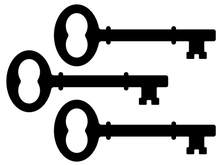
Ballot Privacy:
votes are encrypted



The workflow

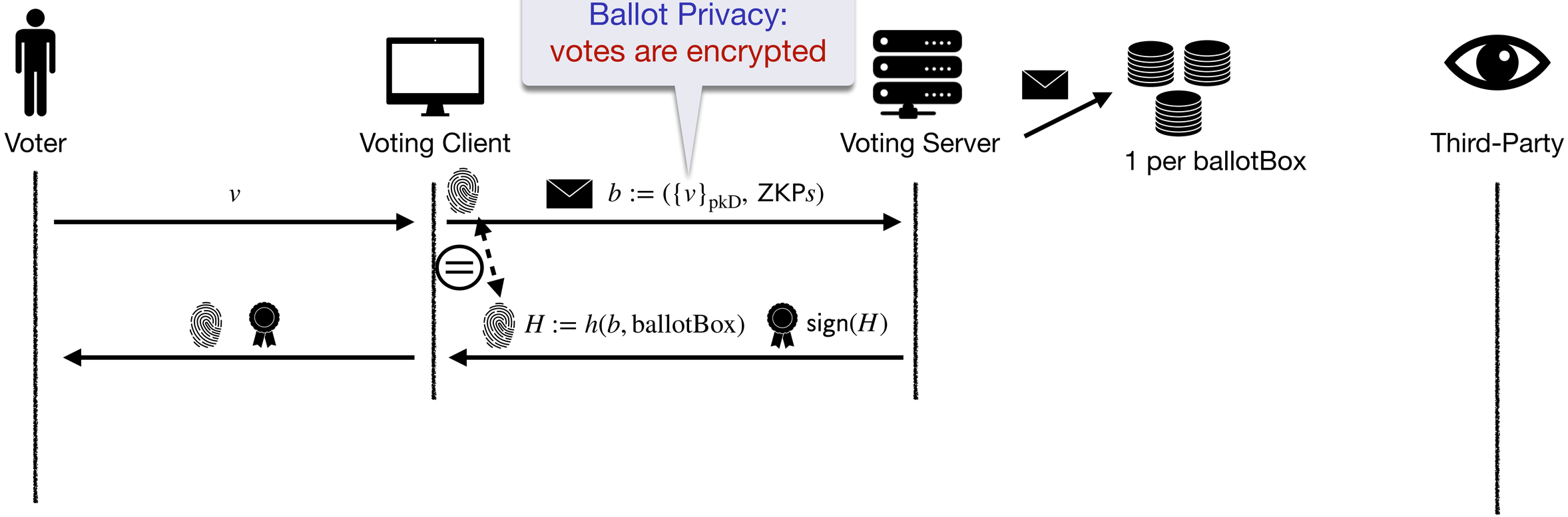
ballotBox for each consular (~city)

Decryption Trustees

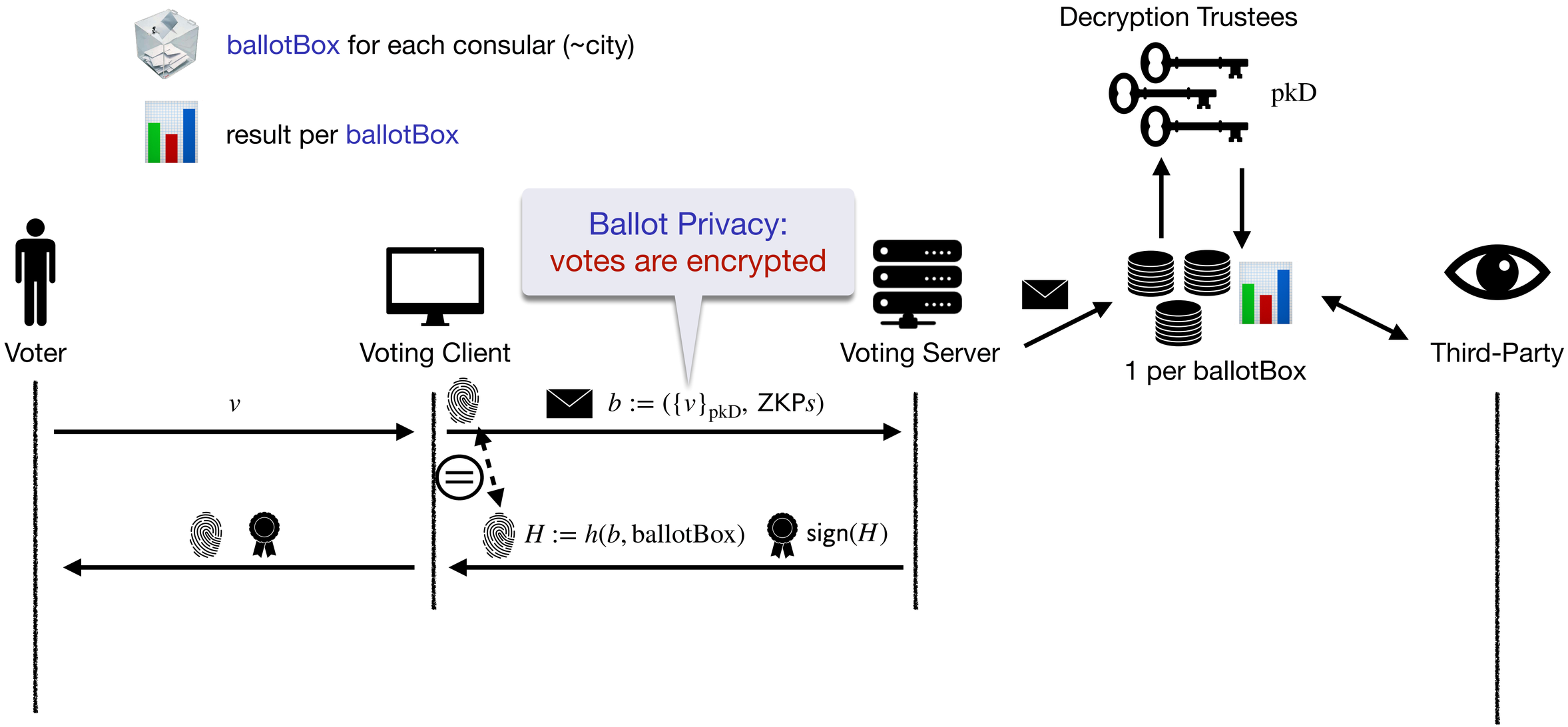


pkD

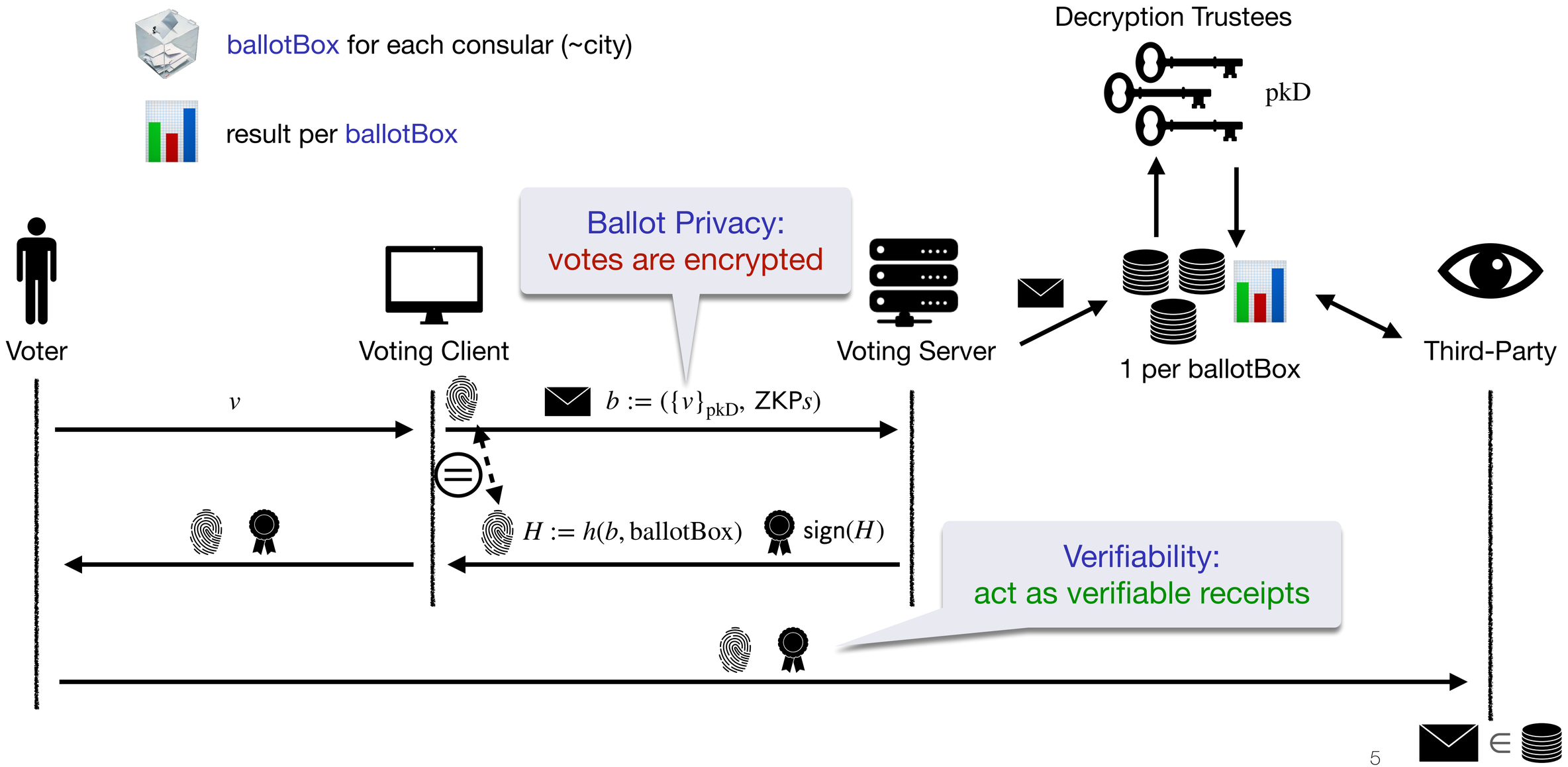
Ballot Privacy:
votes are encrypted



The workflow



The workflow

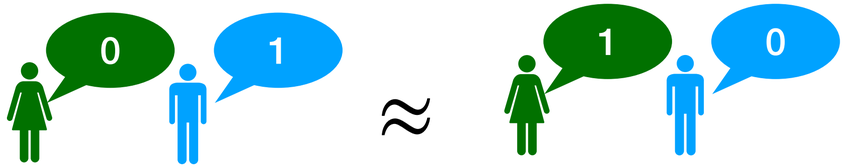


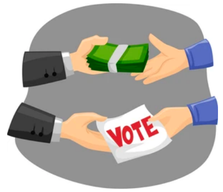
What security do we have?

Confidentiality of the votes

Vote privacy

"No one should know who I voted for"





Receipt-freeness

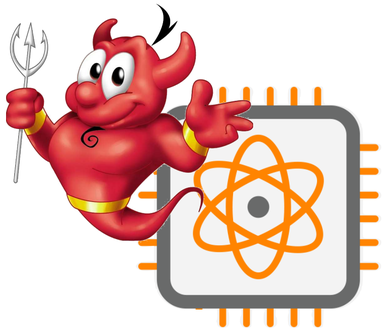
"No one should know who I voted for
even if I want to sell my vote"

Coercion resistance

"No one should know who I voted for
even if I want to sell my vote
and/or someone wants to coerce me "

Everlasting privacy

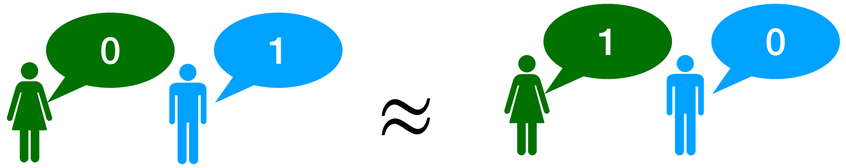
Vote secrecy must still hold in
10, 20, …, 100 years!

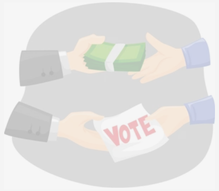


Confidentiality of the votes

Vote privacy

"No one should know who I voted for"





Receipt-freeness

"No one should know who I voted for even if I want to sell my vote"

Outside CNIL recommandations & law

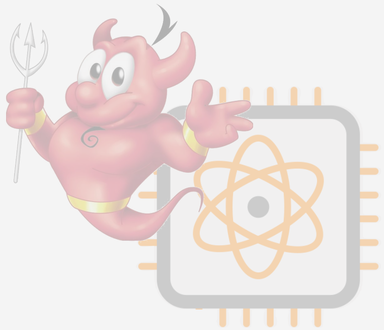
Coercion resistance

"No one should know who I voted for even if I want to sell my vote and/or someone wants to coerce me "

Outside CNIL recommandations & law

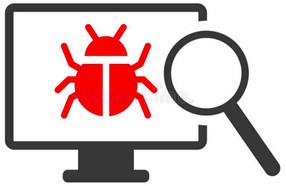
Everlasting privacy

Vote secrecy must still hold in 10, 20, …, 100 years!

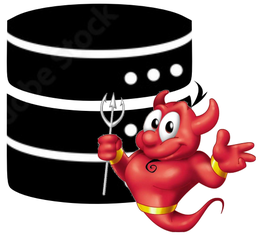


Trivial since nothing is made public

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



Cast-as-intended - “The cast ballot corresponds to the voter’s intent”



Recorded-as-cast - “The cast ballot is correctly added in the ballot-box”

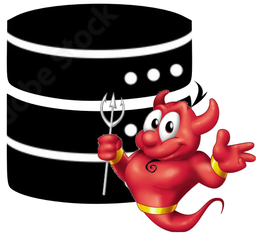
Eligibility - “The ballot-box contains ballots cast by legitimate voters only”

Tallied-as-recorded - “The result corresponds to the ballot-box”

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

Cast-as-intended - “The cast ballot corresponds to the voter’s intent”

Outside CNIL recommandations & law



Recorded-as-cast - “The cast ballot is correctly added in the ballot-box”

Eligibility - “The ballot-box contains ballots cast by legitimate voters only”

Very weak claims…

Tallied-as-recorded - “The result corresponds to the ballot-box”

Threat models

Vote secrecy - "No one should know who I voted for" 

Verifiability - "No one can modify the outcome of the election" 

Threat models —

	Voter	Voting Client	Communication Channel	Voting Server	Decryption Trustees	Third-Party
Vote privacy	😇	😇	😈	😈	😇	😇
Verifiability	😇	😇	😈	😈	😈	😇

Threat models

Vote secrecy - "No one should know who I voted for" 

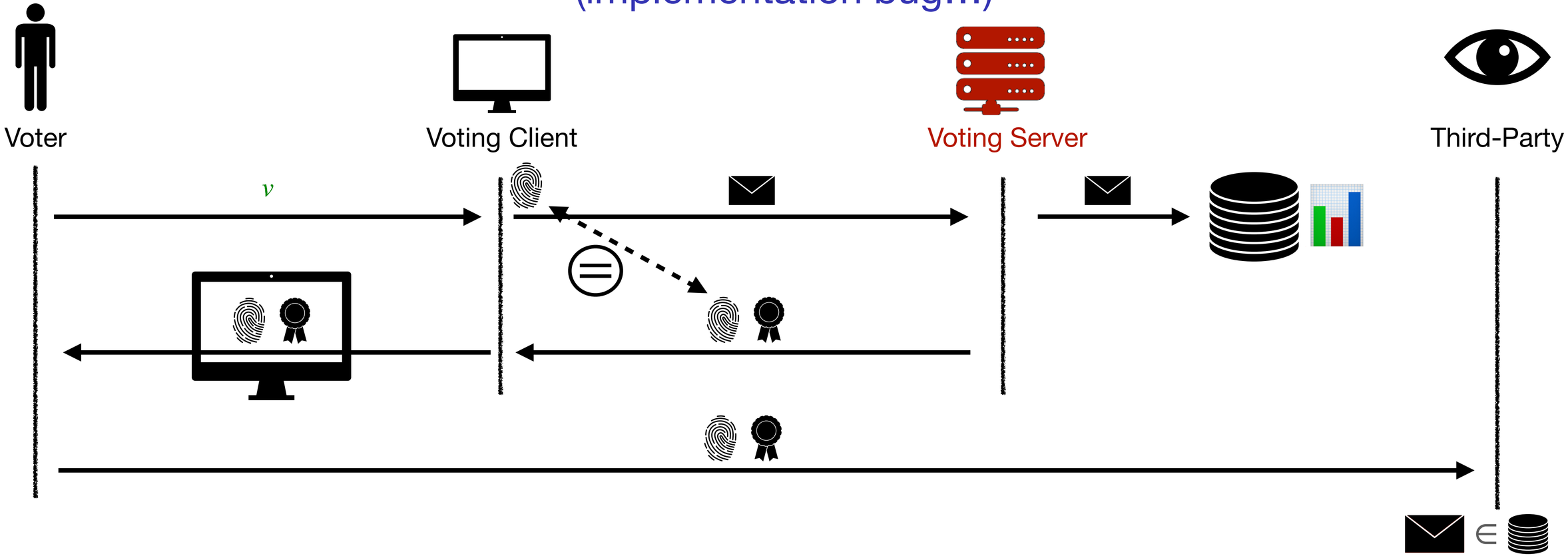
Verifiability - "No one can modify the outcome of the election" 

Threat models —

	Voter	Voting Client	Communication Channel	Voting Server	Decryption Trustees	Third-Party
Vote privacy	😇	😇	😈	😇	😇	😇
Verifiability	😇	😇	😈	😇	😇	😇

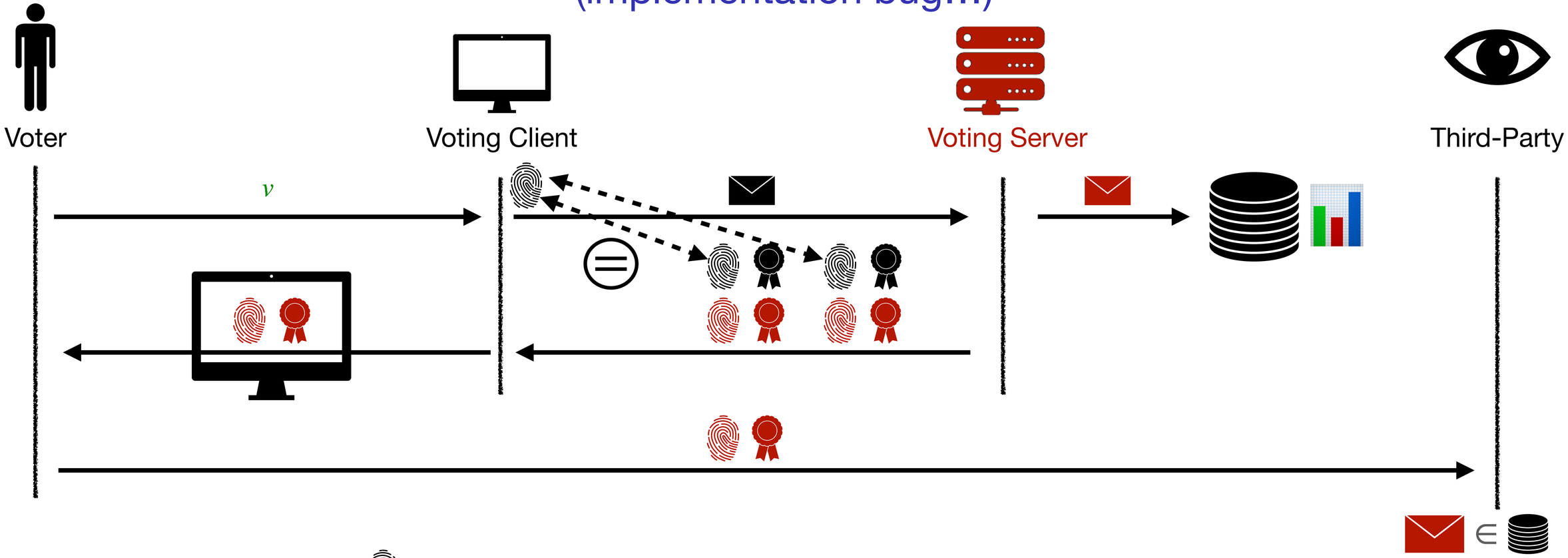
Attack against verifiability

(implementation bug…)



Attack against verifiability

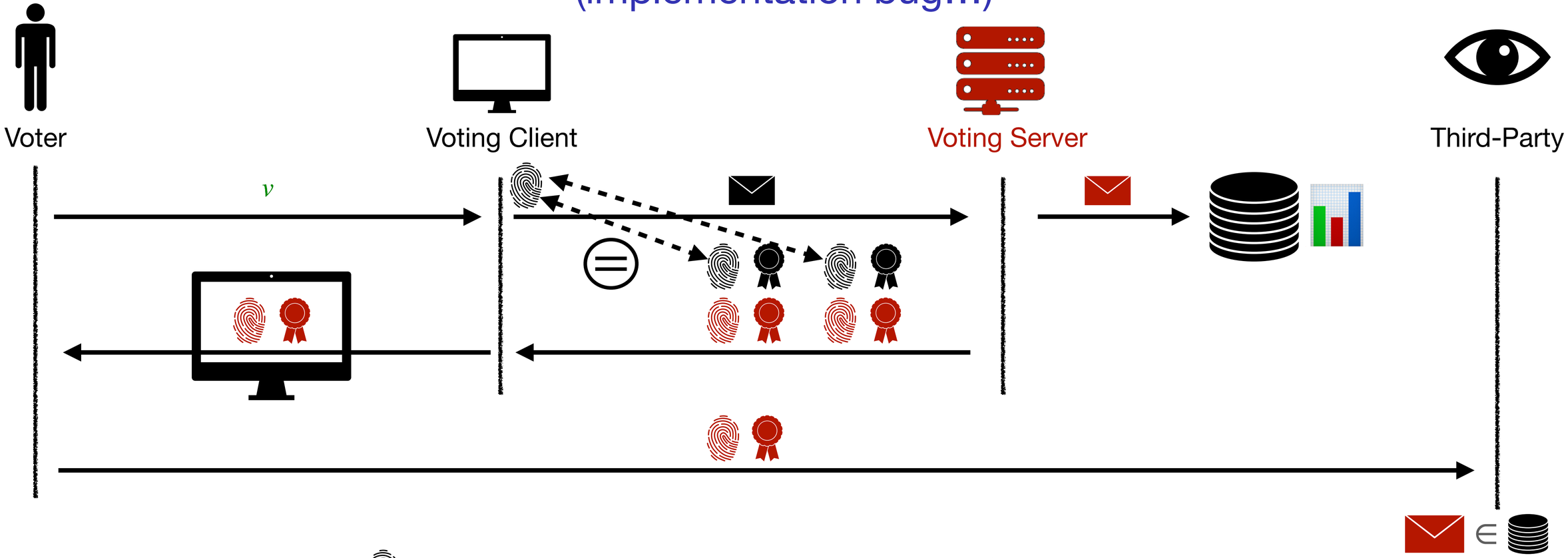
(implementation bug…)



- There are 4 versions of with various consistency checks in the JavaScript voting client
- **Implementation vulnerability** ⇨ the displayed to the voter may not correspond to their ballot

Attack against verifiability

(implementation bug…)



- There are 4 versions of with various consistency checks in the JavaScript voting client
- **Implementation vulnerability** ⇨ the displayed to the voter may not correspond to their ballot

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

Attack against vote privacy

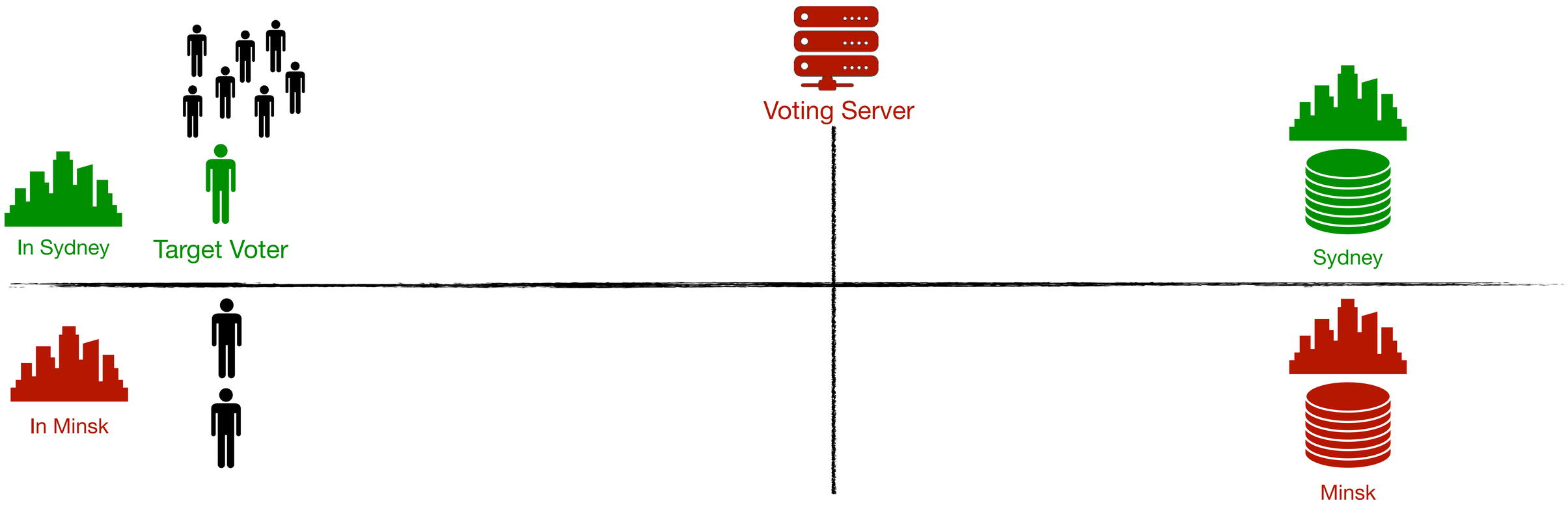
(design vulnerability…)

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

Attack against vote privacy

(design vulnerability…)

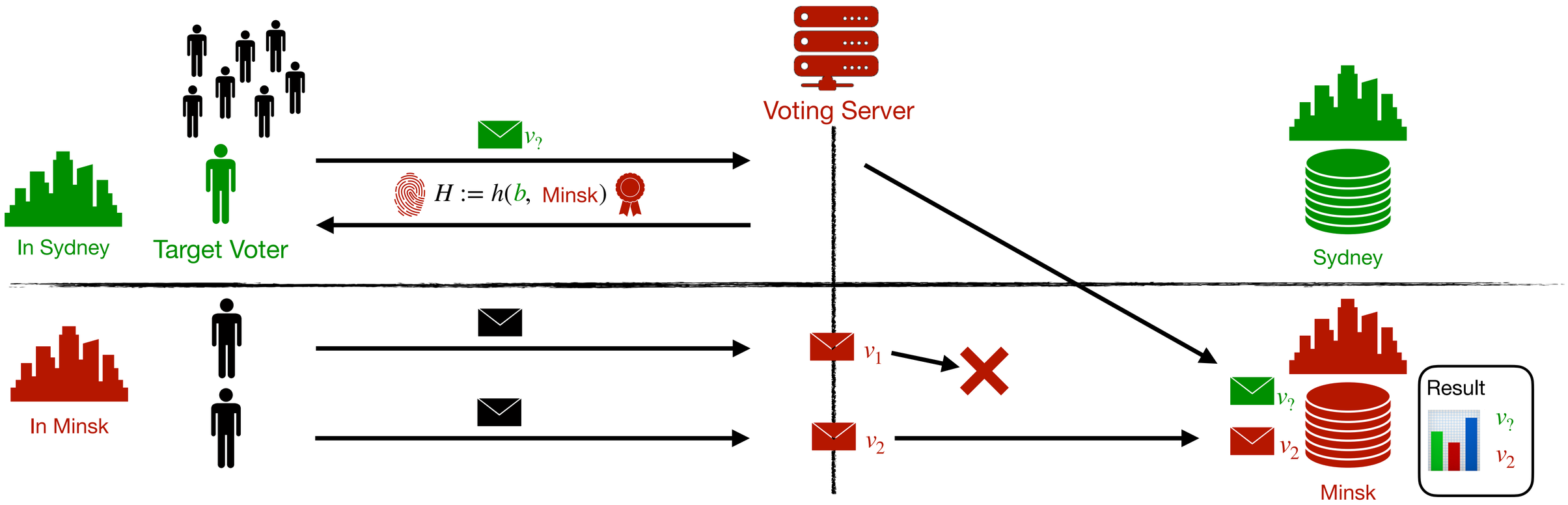
- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Attack against vote privacy

(design vulnerability…)

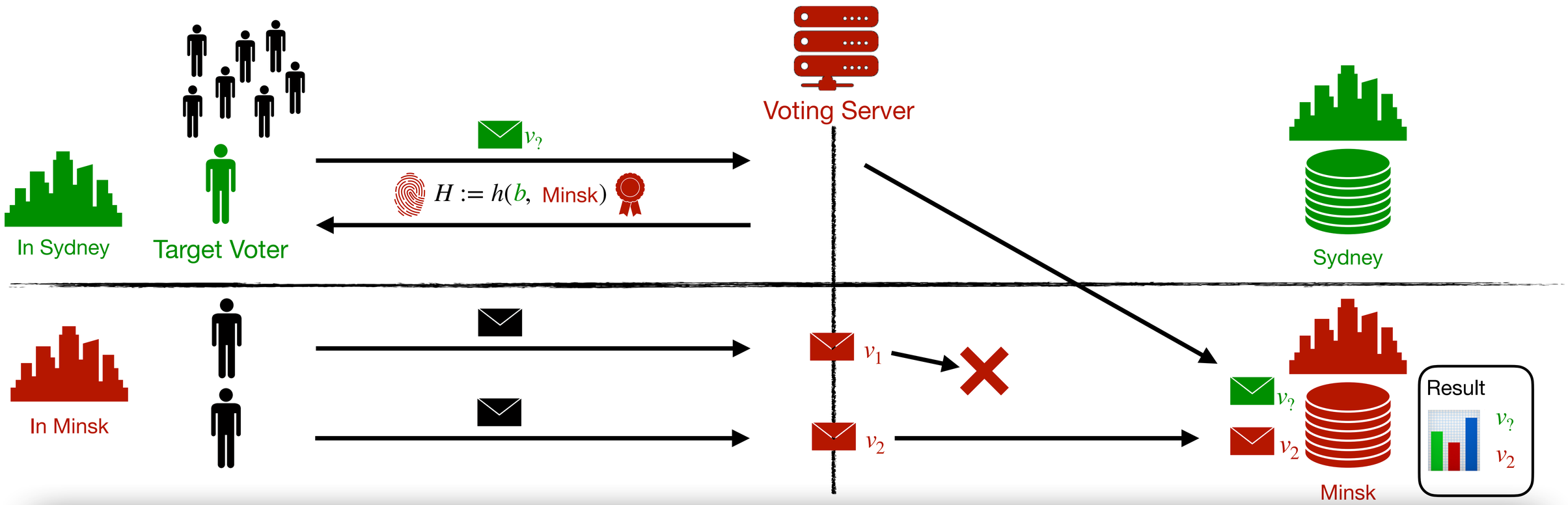
- **Design vulnerability** ⇨ ballots ZKPs do not bind ballotBox



Attack against vote privacy

(design vulnerability…)

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



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  ✔/✘ non-optimally implemented for 2023
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.

Story summary

Story summary

BELENIOS

State-of-art protocol

affected by none of the attacks

Story summary

BELENIOS

State-of-art protocol

affected by none of the attacks

Adapt the design →

FLEP Protocol

- downloadable receipts
- adapt authentication
- support for multiple ballot-boxes

Story summary

BELENIOS

State-of-art protocol

affected by none of the attacks

Adapt the design

- downloadable receipts
- adapt authentication
- support for multiple ballot-boxes

FLEP Protocol

Implement, Deploy, Audit

- customized front-end
- deployment in a specific infrastructure

2022 Election

FLEP 2022

affected by 6 attacks

+ other concerns not discussed here

Lessons learned
(recommandations and research questions)

Designing a security protocol is highly error-prone and…

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

➡ state-of-the-art solutions propose unpractical solutions
- voters authentication currently relies on a single-point-of-trust
- distributed key generation and decryption is actually centralized on a single machine

Lessons learned
(recommandations and research questions)

Designing a security protocol is highly error-prone and…

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

➡ state-of-the-art solutions propose unpractical solutions
- voters authentication currently relies on a single-point-of-trust
- distributed key generation and decryption is actually centralized on a single machine

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

Lessons learned
(recommandations and research questions)

Designing a security protocol is highly error-prone and…

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

➡ state-of-the-art solutions propose unpractical solutions
- voters authentication currently relies on a single-point-of-trust
- distributed key generation and decryption is actually centralized on a single machine

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

Implementing/deploying is as difficult…

➡ 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)

Designing a security protocol is highly error-prone and…

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

➡ state-of-the-art solutions propose unpractical solutions
- voters authentication currently relies on a single-point-of-trust
- distributed key generation and decryption is actually centralized on a single machine

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

Implementing/deploying is as difficult…

➡ 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 privacy 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