COMPUTER SECURITY LAB

RICE UNIVERSITY

VoteBox: a verifiable, tamper-evident electronic voting system

Dan S. Wallach
Rice University

(Joint work with Daniel R. Sandler)

Talk outline

Talk outline

I. Background

Trustworthiness of electronic voting machines
Why it’s worth improving them
Related work

Talk outline

I. Background

Trustworthiness of electronic voting machines
Why it's worth improving them
Related work

II. The design of VoteBox

Durability and audit
Privacy and verifiability
User interface
Extensions

Talk outline

I. Background

Trustworthiness of electronic voting machines
Why it's worth improving them
Related work

II. The design of VoteBox

Durability and audit
Privacy and verifiability
User interface
Extensions

III. Conclusion

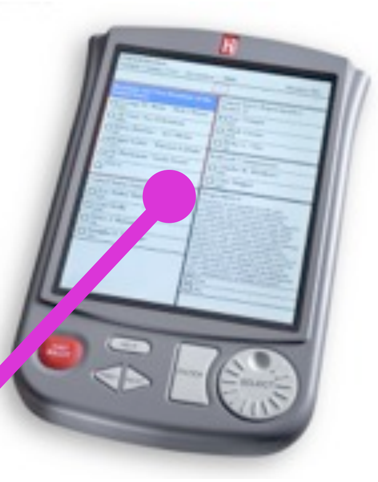
1. Background

DRE voting machines

(Direct Recording Electronic)

DRE voting machines

(Direct Recording Electronic)



touch screen / buttons
graphical display

flash memory

DRE voting machines

(Direct Recording Electronic)

touch screen / buttons
graphical display

DREs discredited

Software bugs & design flaws identified by e-voting researchers

DREs discredited

Software bugs & design flaws identified by e-voting researchers

2003 Analysis of Diebold AccuVote TS

- **Leaked source code analyzed [Kohno et al. 2004]**
- **Poor software engineering, incorrect cryptography, vulnerable to malicious upgrades, multiple voting**

DREs discredited

Software bugs & design flaws identified by e-voting researchers

2003 Analysis of Diebold AccuVote TS

- **Leaked source code analyzed [Kohno et al. 2004]**
- **Poor software engineering, incorrect cryptography, vulnerable to malicious upgrades, multiple voting**

2006 "Voting-machine virus" developed

- **Self-propagating malicious upgrades that spread from machine to machine, altering votes and leaving no trace [Feldman et al. 2006]**

DREs discredited

Software bugs & design flaws identified by e-voting researchers

DREs discredited

Software bugs & design flaws identified by e-voting researchers

2007 Involvement by computer scientists in statewide voting systems audits

- **groundbreaking access to source code of commercial voting systems**

DREs discredited

Software bugs & design flaws identified by e-voting researchers

2007 Involvement by computer scientists in statewide voting systems audits

- **groundbreaking access to source code of commercial voting systems**

- **Top-To-Bottom Review (California)**
  - **All machines certified for use in CA found to have serious bugs & be vulnerable to attack**
  - **Viral-style attacks found in all systems**

DREs discredited

Software bugs & design flaws identified by e-voting researchers

2007 Involvement by computer scientists in statewide voting systems audits

groundbreaking access to source code of commercial voting systems

Top-To-Bottom Review (California)

- All machines certified for use in CA found to have serious bugs & be vulnerable to attack
- Viral-style attacks found in all systems

EVEREST study (Ohio)

- All machines certified in OH found vulnerable (validating CA-TTBR)
- Showed that hundreds of votes were lost in 2004

Result:
undermined trust in elections

Trash

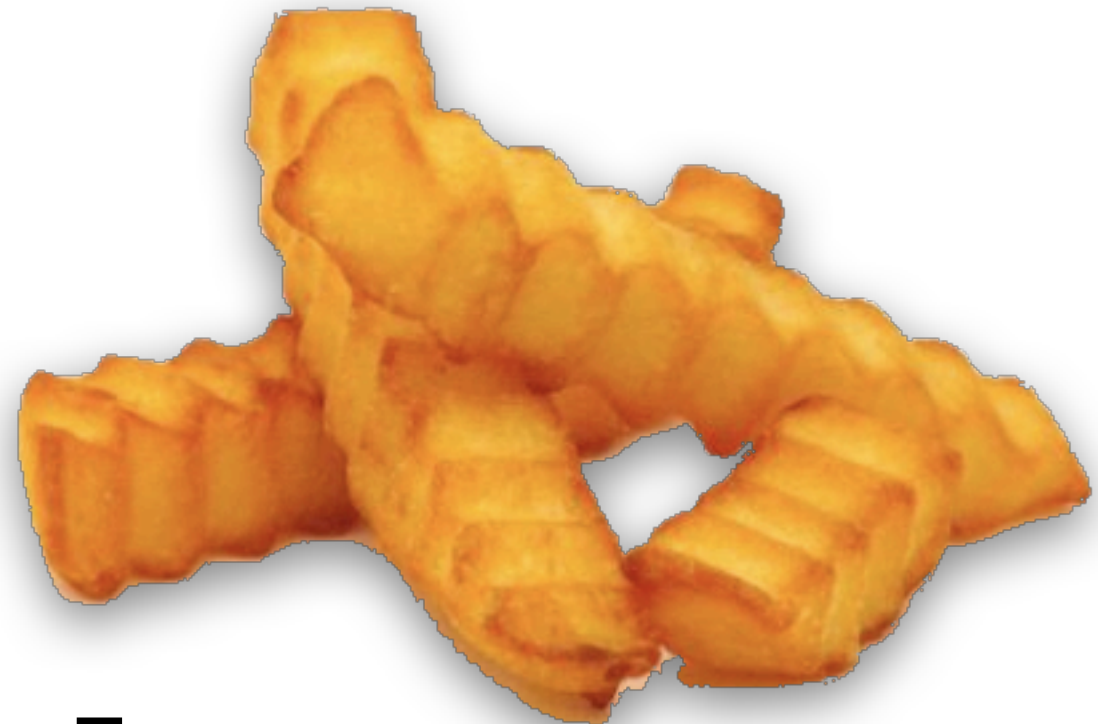
?

voters
prefer
electronic
voting

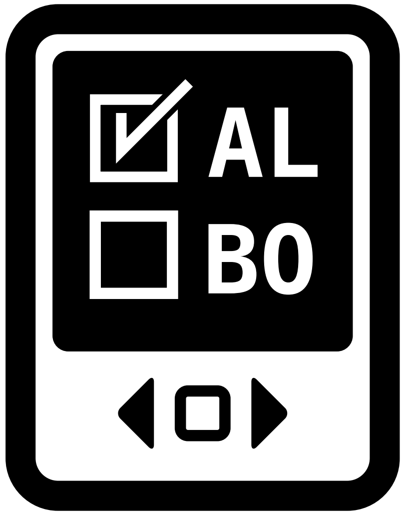
voters prefer electronic voting

S. P. Everett, K. K. Greene, M. D. Byrne, D. S. Wallach, K. Derr, **D. R. Sandler**, and T. Torous.
Electronic voting machines versus traditional methods: Improved preference, similar performance.
In CHI 2008.

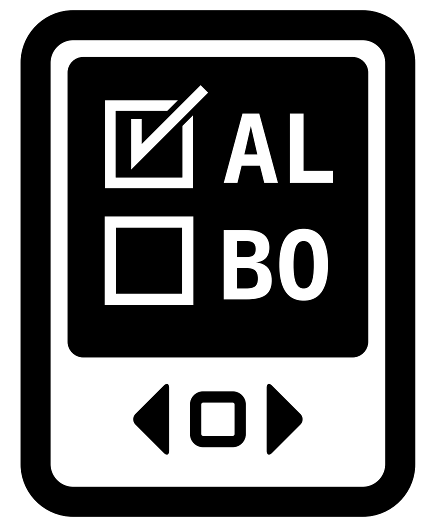
voters **prefer** electronic voting



S. P. Everett, K. K. Greene, M. D. Byrne, D. S. Wallach, K. Derr, **D. R. Sandler**, and T. Torous.
Electronic voting machines versus traditional methods: Improved preference, similar performance.
In CHI 2008.

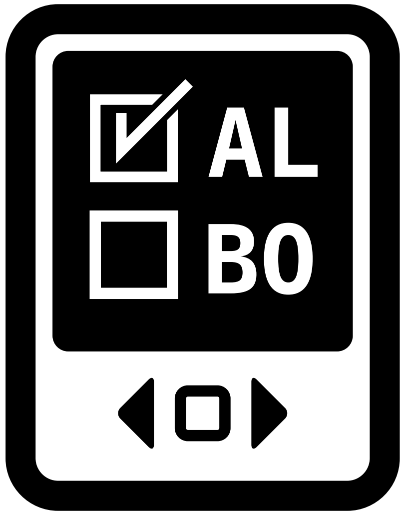

AL
BO

legitimate benefits



legitimate benefits

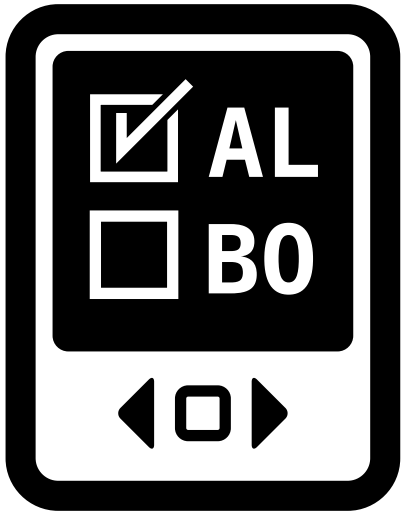
accessibility



legitimate benefits

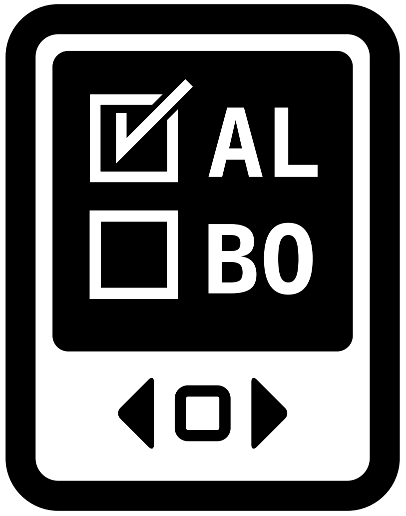
accessibility

feedback



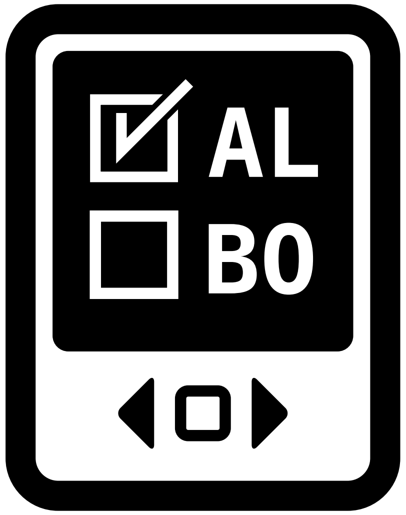
legitimate benefits

accessibility
feedback
flexibility

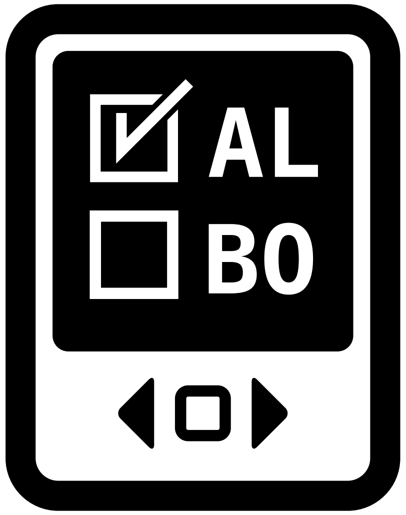


legitimate benefits

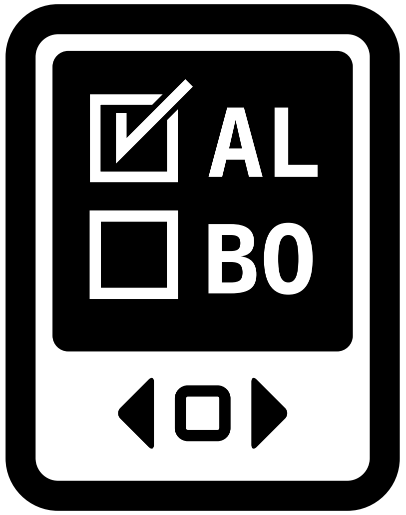
accessibility
feedback
flexibility
satisfaction



can we design a better DRE?



can we design a better DRE?



“better” = ?

goals

1. **resistance to failure & tampering**
   essential vote data should survive hardware failure, poll worker mistakes, attempts to attack the system

goals

2. **tamper-evidence**
   if we are unable to prevent data loss, we must always be able to detect the failure

goals

3. verifiability

two useful properties:

cast-as-intended

"Was my vote recorded faithfully?"
very, very hard for DREs to satisfy

counted-as-cast

"Has my vote been tallied correctly?"
can be somewhat addressed with recounts

goals

resistance to failure & tampering

prevent or minimize data loss

tamper-evidence

if resistance is futile

verifiability

cast-as-intended; counted-as-cast

DRE user experience

smaller codebase

a computer science problem

resistance to failure & tampering

replication; gossip

tamper-evidence

secure logs

verifiability

cryptography

DRE user experience

smaller codebase

a computer science problem

resistance to failure & tampering

replication; gossip

tamper-evidence

secure logs

verifiability

cryptography

DRE user experience

smaller codebase

a computer science problem

resistance to failure & tampering

replication; gossip

tamper-evidence

secure logs

Auditorium

verifiability

cryptography

DRE user experience

smaller codebase

a computer science problem

resistance to failure & tampering

replication; gossip

tamper-evidence

secure logs

Auditorium

verifiability

cryptography

DRE user experience

smaller codebase

a computer science problem

resistance to failure & tampering

replication; gossip

tamper-evidence

secure logs **Auditorium**

verifiability

cryptography **Ballot challenge**

DRE user experience

smaller codebase

a computer science problem

resistance to failure & tampering

replication; gossip

tamper-evidence

secure logs

Auditorium

verifiability

cryptography

Ballot challenge

DRE user experience

smaller codebase

a computer science problem

resistance to failure & tampering

replication; gossip

tamper-evidence

secure logs

Auditorium

verifiability

cryptography

Ballot challenge

DRE user experience

smaller codebase

PRUI

other research voting systems

Mixnets

Chaum ’81

Neff ’01

Chaum ’04 (visual crypto)

Prêt-à-voter: Chaum, Ryan, Schneider ’05

Blind signatures

FOO: Fujioka, Okamoto, Ohta ’92

EVOX: Herschberg ’97

Sensus: Cranor, Cytron ’97

Storage

Molnar, Kohno, Sastry, Wagner ’06

Bethencourt, Boneh, Waters ’07

Homomorphic crypto and NIZKs

Benaloh ’87

Adder: Kiayias, Korman, Walluck ’06

Moran, Naor ’06

Benaloh ’07

Helios: Adida ’08

Civitas: Clarkson, Chong, Myers ’08

TCB reduction

Pvote: Yee ’06, ’07

Sastry, Kohno, Wagner ’06

Paper

Punchscan: Chaum ’05

ThreeBallot: Rivest ’06

Scantegrity: Chaum ’07

other research voting systems

Mixnets

Chaum ’81

Neff ’01

Chaum ’04 (visual crypto)

Prêt-à-voter: Chaum, Ryan, Schneider ’05

Blind signatures

FOO: Fujioka, Okamoto, Ohta ’92

EVOX: Herschberg ’97

Sensus: Cranor, Cytron ’97

Storage

Molnar, Kohno, Sastry, Wagner ’06

Bethencourt, Boneh, Waters ’07

Homomorphic crypto and NIZKs

Benaloh ’87

Adder: Kiayias, Korman, Walluck ’06

Moran, Naor ’06

Benaloh ’07

Helios: Adida ’08

Civitas: Clarkson, Chong, Myers ’08

TCB reduction

Pvote: Yee ’06, ’07

Sastry, Kohno, Wagner ’06

Paper

Punchscan: Chaum ’05

ThreeBallot: Rivest ’06

Scantegrity: Chaum ’07

“The Auditorium”

“The Auditorium”

Auditorium’s approach

D. Sandler and D. S. Wallach. **Casting Votes in the Auditorium.** In Proceedings of the 2nd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT’07).

Auditorium's approach

Store everything everywhere

Massive **redundancy**

Stop trusting DREs to keep their own audit data

D. Sandler and D. S. Wallach. **Casting Votes in the Auditorium.** In Proceedings of the 2nd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07).

Auditorium’s approach

Store everything everywhere

Massive **redundancy**

Stop trusting DREs to keep their own audit data

Link all votes, events together

Create a **secure timeline** of election events

Tamper-evident proof of each vote’s legitimacy

D. Sandler and D. S. Wallach. **Casting Votes in the Auditorium.** In Proceedings of the 2nd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT’07).

Ingredient: hash chains

```
           “Machine turned on” (HASH = 0x1234)
“Cast a vote after event 0x1234” (HASH = 0xABCD)
“Cast a vote after event 0xABCD” (HASH = 0xBEEF)
 “Turned off after event 0xBEEF” (HASH = 0x4242)
```

Ingredient: hash chains

```
        “Machine turned on” (HASH = 0x1234)
“Cast a vote after event 0x1234” (HASH = 0xABCD)
“Cast a vote after event 0xABCD” (HASH = 0xBEEF)
 “Turned off after event 0xBEEF” (HASH = 0x4242)
```

A hash-chained secure log

Every event includes the cryptographic hash (e.g. SHA1) of a previous event

[Schneier & Kelsey ’99]

Ingredient: hash chains

```
          "Machine turned on" (HASH = 0x1234)
"Cast a vote after event 0x1234" (HASH = 0xABCD)
"Cast a vote after event 0xABCD" (HASH = 0xBEEF)
 "Turned off after event 0xBEEF" (HASH = 0x4242)
```

A hash-chained secure log

Every event includes the cryptographic hash (e.g. SHA1) of a previous event

[Schneier & Kelsey '99]

Result: provable order

If **Y** includes H(**X**), then **Y** must have happened after **X**

Ingredient: hash chains

```
              "Machine turned on" (HASH = 0x1234)
"Cast a vote after event 0x1234" (HASH = 0xABCD)
"Cast a vote after event 0xABCD" (HASH = 0xBEEF)
 "Turned off after event 0xBEEF" (HASH = 0x4242)
```

A hash-chained secure log

Every event includes the cryptographic hash (e.g. SHA1) of a previous event

[Schneier & Kelsey '99]

Result: provable order

If **Y** includes H(**X**), then **Y** must have happened after **X**

Any attempted change to the log

invalidates all later hashes (breaks the chain)

Ingredient: hash chains

```
           “Machine turned on” (HASH = 0x1234)
“Cast a vote after event 0x1234” (HASH = 0xABCD)
“Cast a vote after event 0xABCD” (HASH = 0xBEEF)
 “Turned off after event 0xBEEF” (HASH = 0x4242)
```

A hash-chained secure log

Every event includes the cryptographic hash (e.g. SHA1) of a previous event

[Schneier & Kelsey ’99]

Result: provable order

If **Y** includes H(**X**), then **Y** must have happened after **X**

Any attempted change to the log

invalidates all later hashes (breaks the chain)

To alter, insert, or delete a single record

you must alter every subsequent event as well!

Ingredient: timeline entanglement

Ingredient: timeline entanglement

Entanglement = “chain with hashes from others”

Result: event ordering **between** participants

[Maniatis & Baker ’02]

Ingredient: timeline entanglement

Entanglement = “chain with hashes from others”

Result: event ordering **between** participants

[Maniatis & Baker ’02]

Malicious machines can’t retroactively alter their own logs

it would violate commitments they have already exchanged with others

Ingredient: timeline entanglement

Entanglement = “chain with hashes from others”
Result: event ordering **between** participants

[Maniatis & Baker ’02]

Malicious machines can’t retroactively alter their own logs
it would violate commitments they have already exchanged with others

So with whom should a VoteBox entangle?

Ingredient: broadcast

Ingredient: broadcast

All-to-all communication

All messages signed & distributed to every VoteBox

Each machine records each message independently

Ingredient: broadcast

All-to-all communication

All messages signed & distributed to every VoteBox

Each machine records each message independently

result: massive replication

$O(\boldsymbol{N}^2)$, but $\boldsymbol{N}$ is small in a polling place

Ingredient: broadcast

All-to-all communication

All messages signed & distributed to every VoteBox

Each machine records each message independently

result: massive replication

O(N^2), but N is small in a polling place

Mechanism for entanglement

each log fills up with local and remote messages

when sending new messages, include recent hashes (regardless of origin)

Broadcast entanglement =
Auditorium

Unusual prior art

The Papal Conclave

Proceedings **closed to outsiders**

All ballots cast **in plain view**

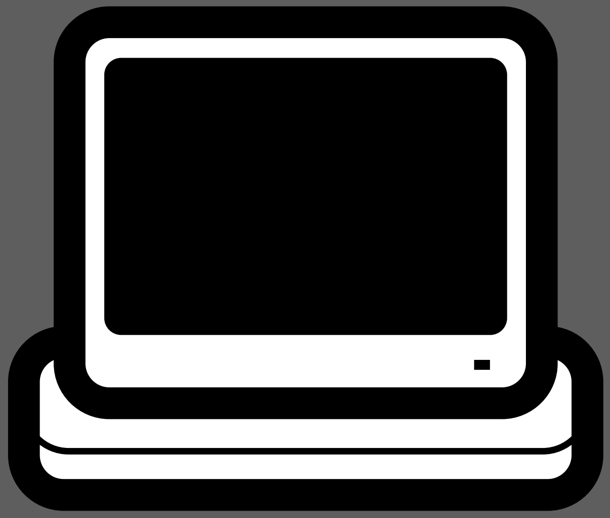
All ballots **secret**

A pragmatic benefit

A pragmatic benefit

The supervisor console

Assistance for poll workers



A pragmatic benefit

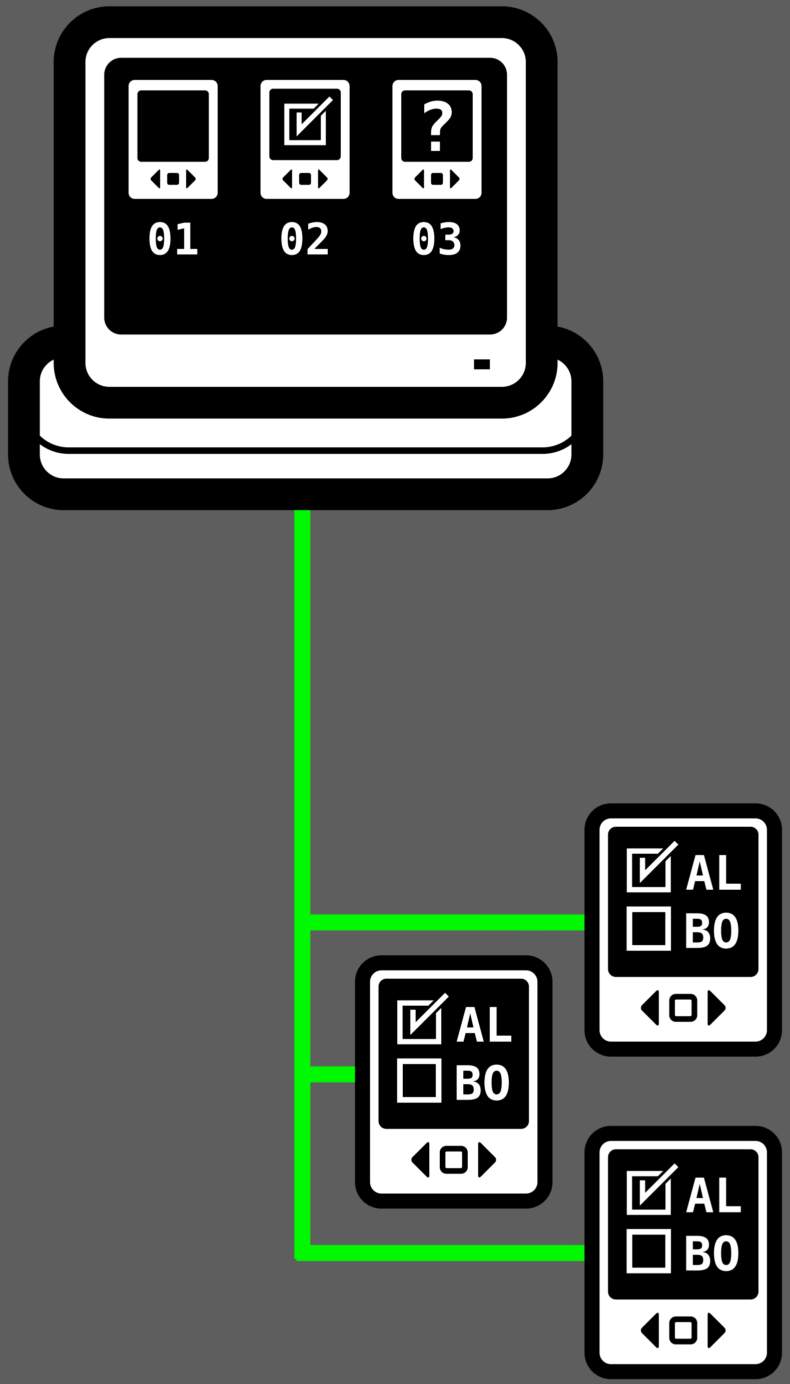
The supervisor console

Assistance for poll workers

Helps conduct the election

Open/close polls, authorize machines to cast ballots

Less opportunity for poll-worker error



A pragmatic benefit

The supervisor console

Assistance for poll workers

Helps conduct the election

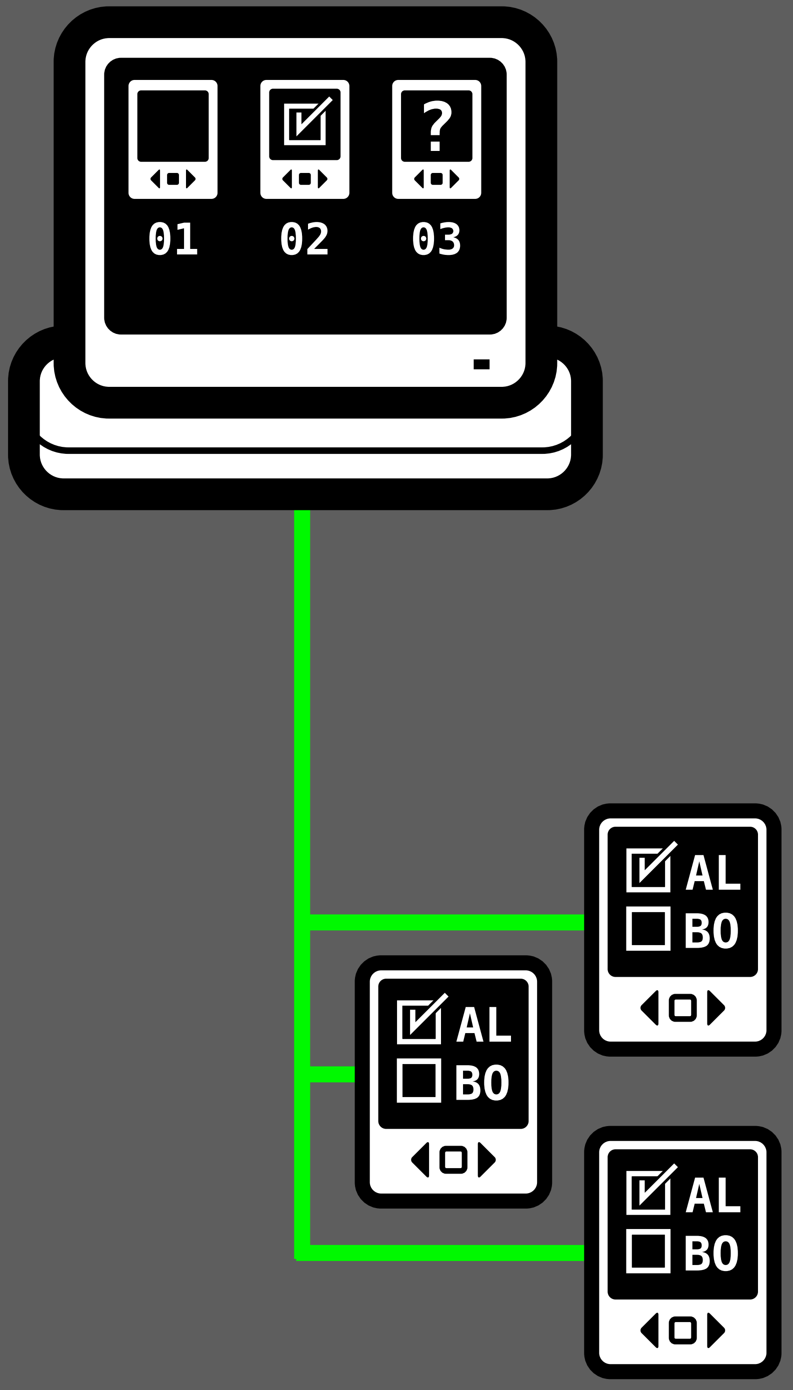
Open/close polls, authorize machines to cast ballots

Less opportunity for poll-worker error

Ballots distributed over the network

Booths are **stateless**, interchangeable

(Supervisor can have a spare as well)



A pragmatic benefit

The supervisor console

Assistance for poll workers

Helps conduct the election

Open/close polls, authorize machines to cast ballots

Less opportunity for poll-worker error

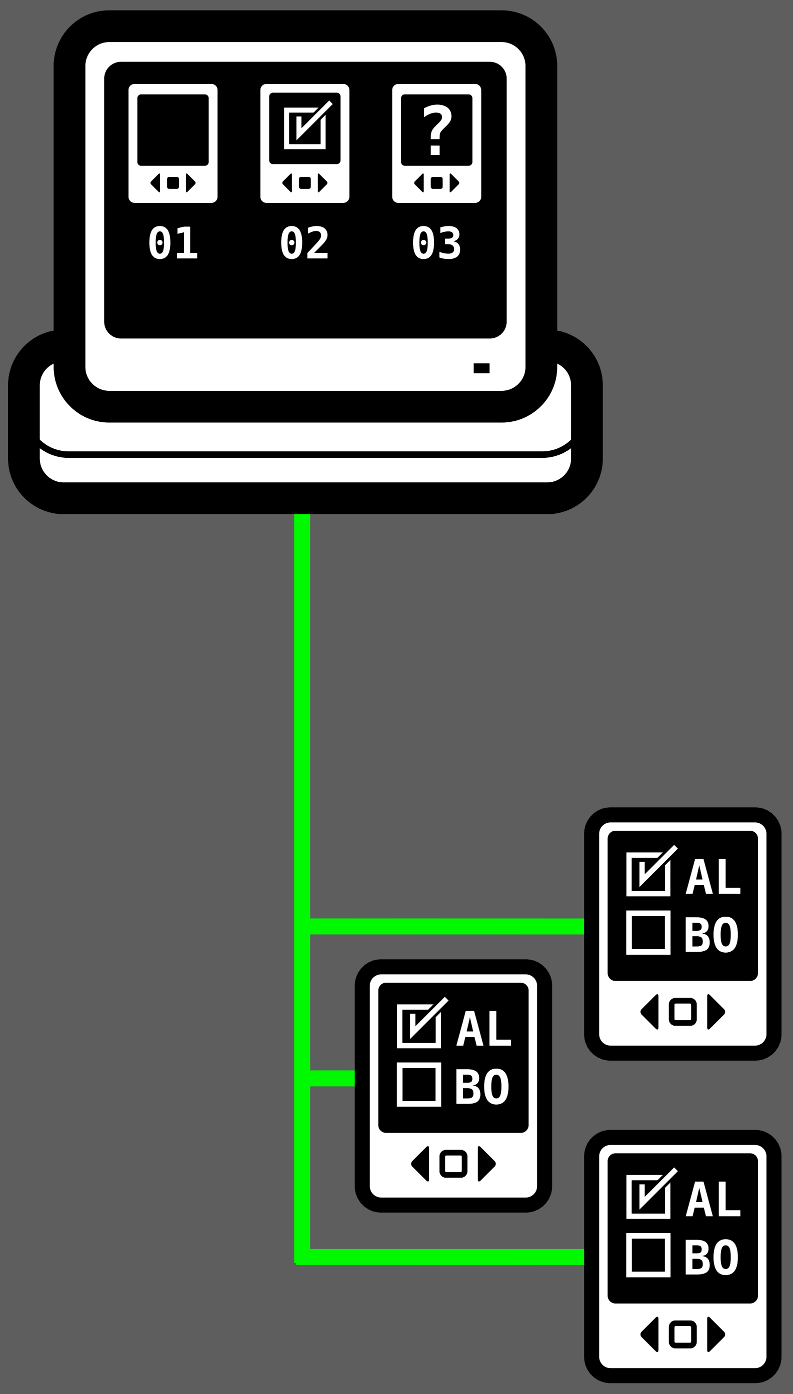
Ballots distributed over the network

Booths are **stateless**, interchangeable

(Supervisor can have a spare as well)

Shows status of all machines

Votes cast, battery running low, etc.



How do you audit a secure log?

How do you audit a secure log?

"Audit logs are useless unless someone reads them. Hence, we first assume that there is a software program whose job it is to scan all audit logs and look for suspicious entries."

—Schneier & Kelsey '99

How do you audit a secure log?

> *“Audit logs are useless unless someone reads them. Hence, we first assume that there is a software program whose job it is to scan all audit logs and look for suspicious entries.”*
>
> **—Schneier & Kelsey ‘99**

Where is that program?

How do you audit a secure log?

“Audit logs are useless unless someone reads them. Hence, we first assume that there is a software program whose job it is to scan all audit logs and look for suspicious entries.”

—Schneier & Kelsey ‘99

Where is that program?

“suspicious” is domain-specific

How do you audit a secure log?

"Audit logs are useless unless someone reads them. Hence, we first assume that there is a software program whose job it is to scan all audit logs and look for suspicious entries."

—Schneier & Kelsey '99

Where is that program?

"suspicious" is domain-specific

QUERIFIER: an audit log analysis tool

How do you audit a secure log?

“Audit logs are useless unless someone reads them. Hence, we first assume that there is a software program whose job it is to scan all audit logs and look for suspicious entries.”

—Schneier & Kelsey ‘99

Where is that program?

“suspicious” is domain-specific

QUERIFIER: an audit log analysis tool

Predicate logic for expressing rules over secure logs

How do you audit a secure log?

"Audit logs are useless unless someone reads them. Hence, we first assume that there is a software program whose job it is to scan all audit logs and look for suspicious entries."

—Schneier & Kelsey '99

Where is that program?

"suspicious" is domain-specific

QUERIFIER: an audit log analysis tool

Predicate logic for expressing rules over secure logs

Key predicate: "precedes" — requires graph search

How do you audit a secure log?

"Audit logs are useless unless someone reads them. Hence, we first assume that there is a software program whose job it is to scan all audit logs and look for suspicious entries."

—Schneier & Kelsey '99

Where is that program?

"suspicious" is domain-specific

QUERIFIER: an audit log analysis tool

Predicate logic for expressing rules over secure logs

Key predicate: "precedes" — requires graph search

Querifier runs on a complete log ("OK" / "Violation")

How do you audit a secure log?

“Audit logs are useless unless someone reads them. Hence, we first assume that there is a software program whose job it is to scan all audit logs and look for suspicious entries.”

—Schneier & Kelsey ‘99

Where is that program?

“suspicious” is domain-specific

QUERIFIER: an audit log analysis tool

Predicate logic for expressing rules over secure logs

Key predicate: “precedes” — requires graph search

Querifier runs on a complete log (“OK” / “Violation”)

or iteratively on a growing log (“OK so far” / “Violation”)

D. Sandler, K. Derr, S. Crosby, and D. S. Wallach. **Finding the evidence in tamper-evident logs.** In Proceedings of the 3rd International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE’08).

Ballots.

Privacy

Privacy

Secure log of votes could be a problem

When decrypted for tallying, votes are exposed in **order**

An observer could match them with voters

Loss of privacy → bribery & coercion*

Privacy

Secure log of votes could be a problem

When decrypted for tallying, votes are exposed in **order**

An observer could match them with voters

Loss of privacy → bribery & coercion*

Anonymity through clever ballot ordering

re-encryption mixnets

lexicographic sorting

Privacy

Secure log of votes could be a problem

When decrypted for tallying, votes are exposed in **order**

An observer could match them with voters

Loss of privacy → bribery & coercion*

Anonymity through clever ballot ordering

re-encryption mixnets

lexicographic sorting

These would still require the ballots to be removed from the ordered audit logs

Ballots in VoteBox

Ballots in VoteBox

logically, a cast ballot is a vector of counters

one per *candidate*

Ballots in VoteBox

logically, a cast ballot is a vector of counters

one per *candidate*

***e.g.,* for one race with three candidates:**

$$\text{ballot} = (a, b, c) \qquad\qquad a, b, c \in \{0, 1\}$$

Ballots in VoteBox

logically, a cast ballot is a vector of counters

one per *candidate*

***e.g.*, for one race with three candidates:**

$$\text{ballot} = (a, b, c) \qquad a, b, c \in \{0, 1\}$$

ballots may therefore be summed

$$\text{tally} = \sum \text{ballot}_i = \left(\sum a_i, \sum b_i, \sum c_i\right)$$

Encryption

Encryption

Ballots should be sealed

protected from prying eyes

once cast, they should be readable only by the parties trusted to count them

Encryption

Ballots should be sealed

protected from prying eyes

once cast, they should be readable only by the parties trusted to count them

But how do we count them?

Remember, we don't want to decrypt them in order

Diffie-Hellman (1976)

$$
\begin{aligned}
\text{Alice} &: \quad \textit{random}\ a \in \mathbb{Z}_p^* \\
\text{Bob} &: \quad \textit{random}\ b \in \mathbb{Z}_p^* \\
\text{Public} &: \quad \textit{generator}\ g \in \mathbb{Z}_p^* \\
A \to B &: \quad g^a \\
B \to A &: \quad g^b \\
\text{Alice} &: \quad \text{computes } (g^b)^a = g^{ab} \\
\text{Bob} &: \quad \text{computes } (g^a)^b = g^{ab} \\
\text{Eve} &: \quad \text{knows } g^a, g^b, \text{cannot compute } g^{ab}
\end{aligned}
$$

Elgamal encryption (1984)

Non-deterministic cryptosystem (different *r* every time)

$$E(g^a, r, M) = \langle g^r, (g^a)^r M \rangle$$

$$D(g^r, g^{ar} M) = \frac{g^{ar} M}{(g^r)^a}$$

$$= M$$

g	group generator
M	plaintext (message)
r	random (chosen at encryption time)
a	(private) decryption key
g^a	(public) encryption key

Homomorphic property

Anybody can combine two ciphertexts to get a new one.

$$\begin{aligned} E(M_1) \oplus E(M_2) &= \;< g^{r_1}, (g^a)^{r_1} M_1 > \oplus < g^{r_2}, (g^a)^{r_2} M_2 > \\ &= \;< g^{r_1} g^{r_2}, (g^a)^{r_1} M_1 (g^a)^{r_2} M_2 > \\ &= \;g^{r_1+r_2}, g^{a(r_1+r_2)} M_1 M_2 \\ &= \;E(M_1 M_2) \end{aligned}$$

g	group generator
M	plaintext (message)
r	random (chosen at encryption time)
a	(private) decryption key
g^a	(public) encryption key

Homomorphic vote tallying

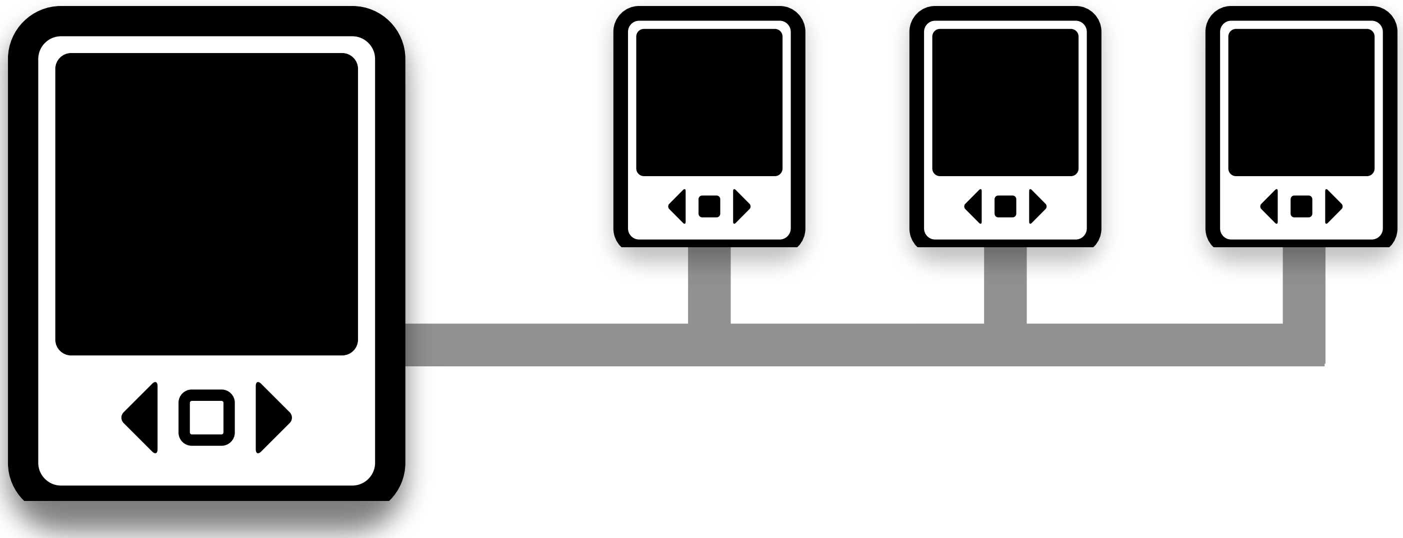
Change messages to counters, additive in exponent of *g*.
"Exponential Elgamal"

$$
\begin{aligned}
E(v_1) \oplus E(v_2) &= \; < g^{r_1}, (g^a)^{r_1} g^{v_1} > \oplus < g^{r_2}, (g^a)^{r_2} g^{v_2} > \\
&= \; < g^{r_1 + r_2}, g^{a(r_1 + r_2)} g^{v_1 + v_2} > \\
&= \; E(v_1 + v_2)
\end{aligned}
$$

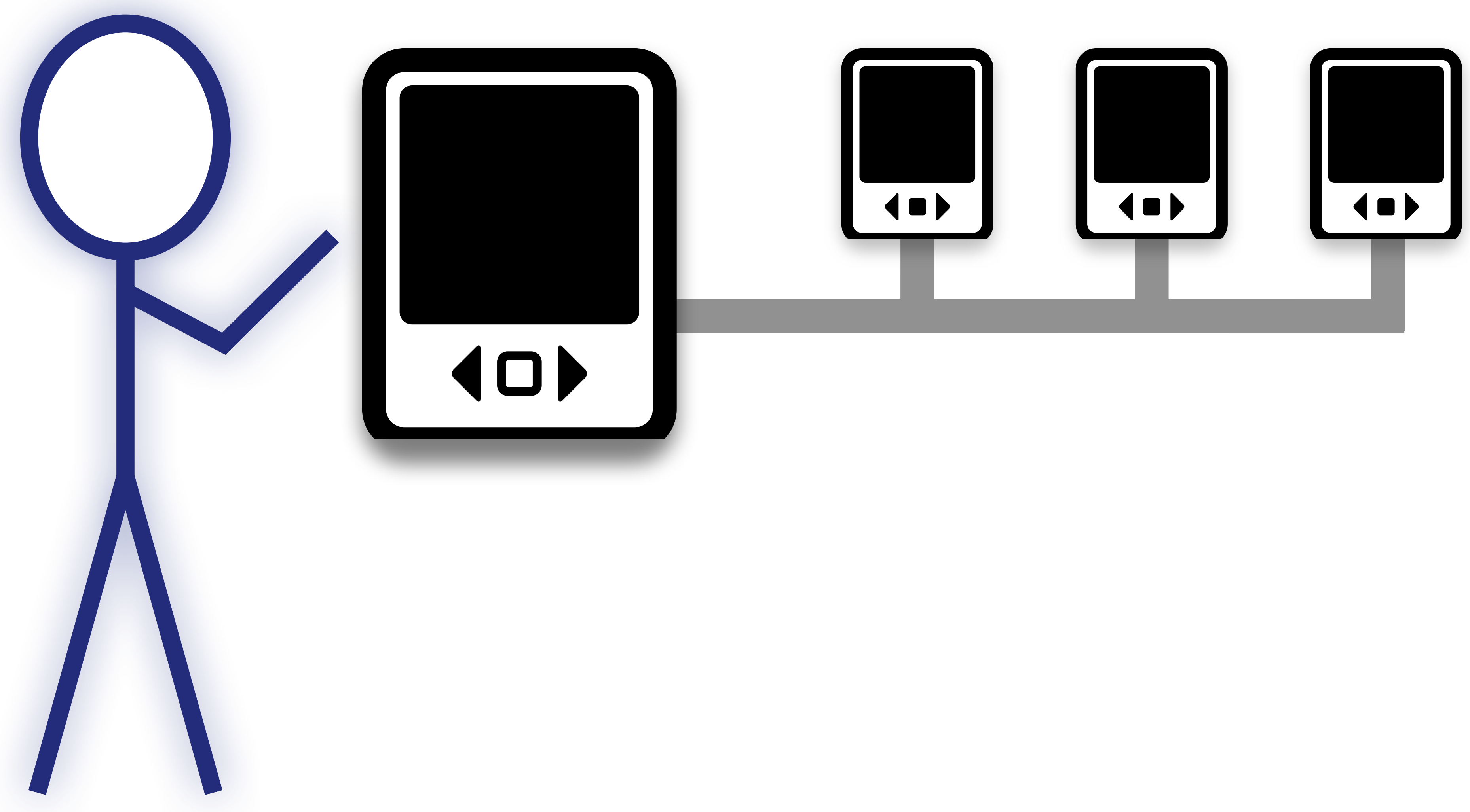
g	group generator
v	plaintext (counters)
r	random (chosen at encryption time)
a	(private) decryption key
g^a	(public) encryption key

How can I be sure my vote is faithfully captured by the voting machine?

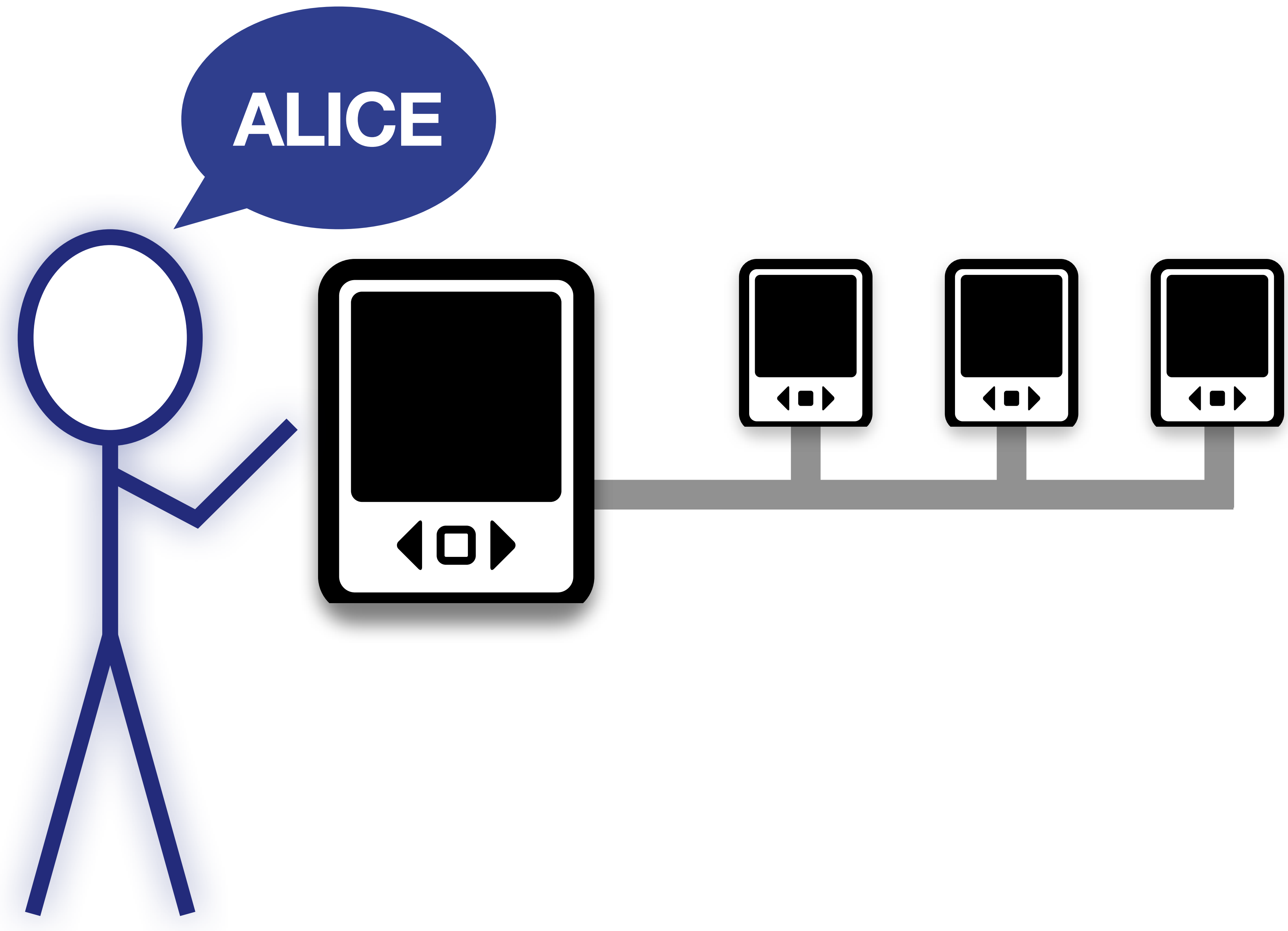
polling place

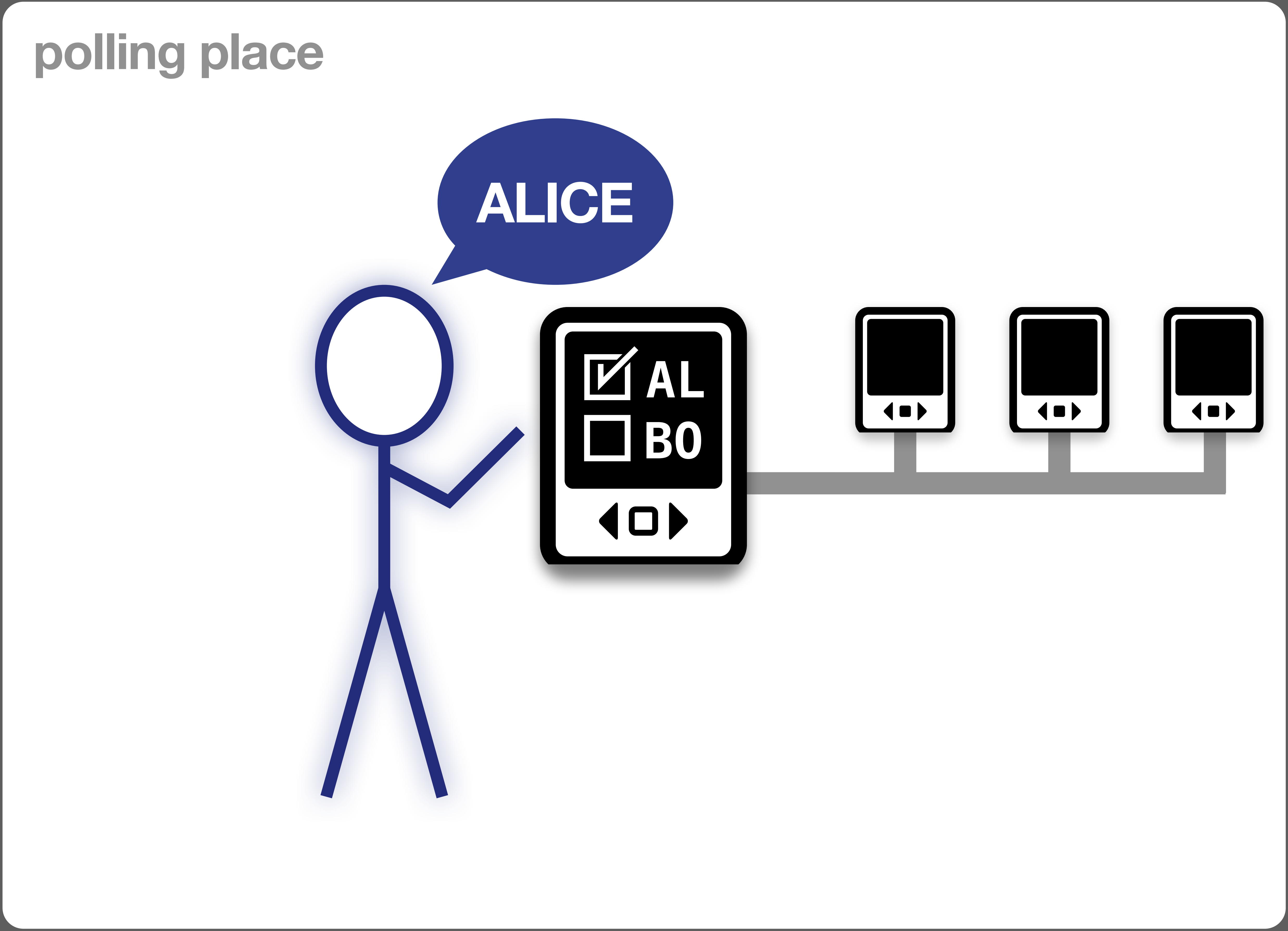


polling place

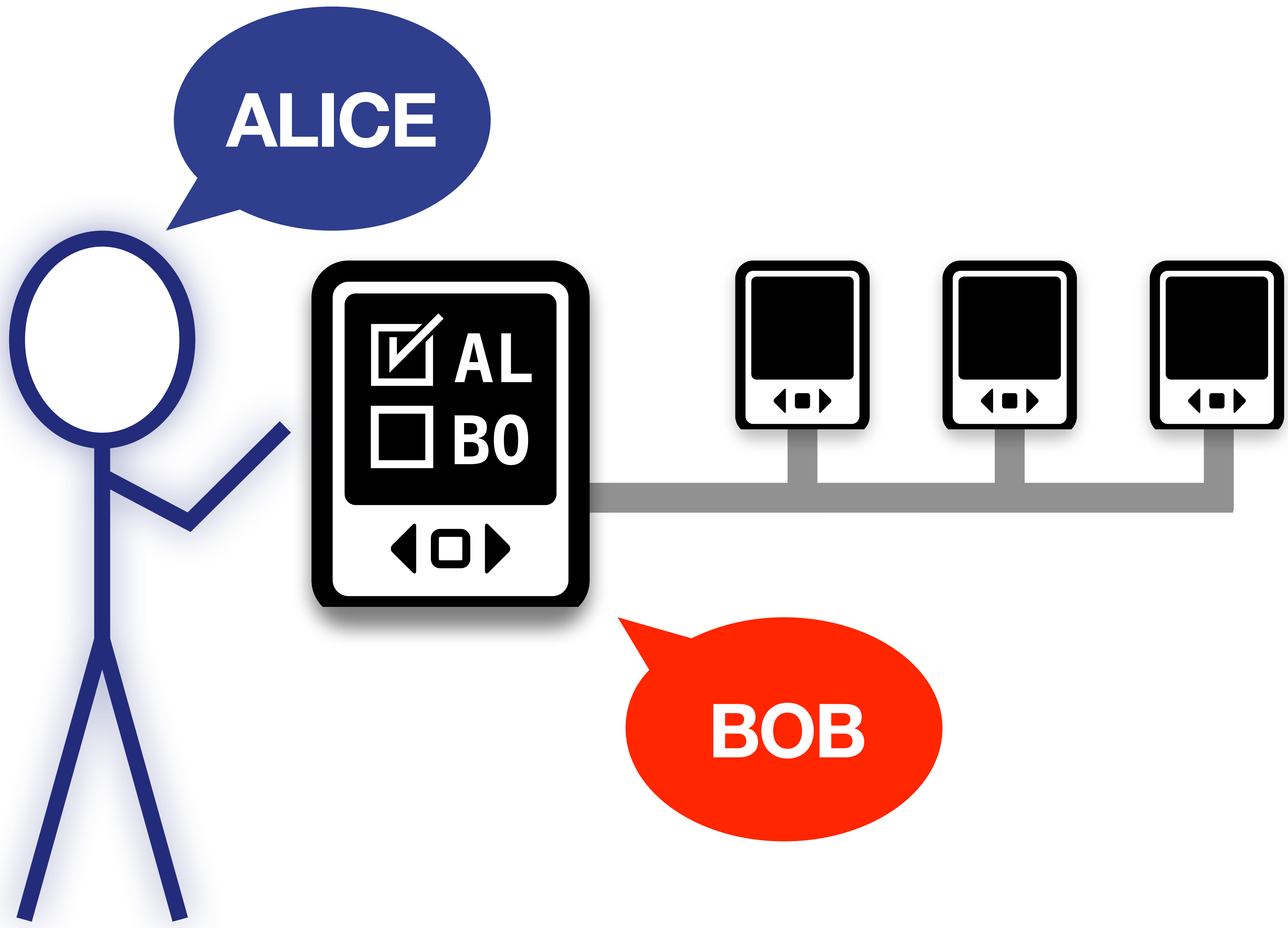


polling place

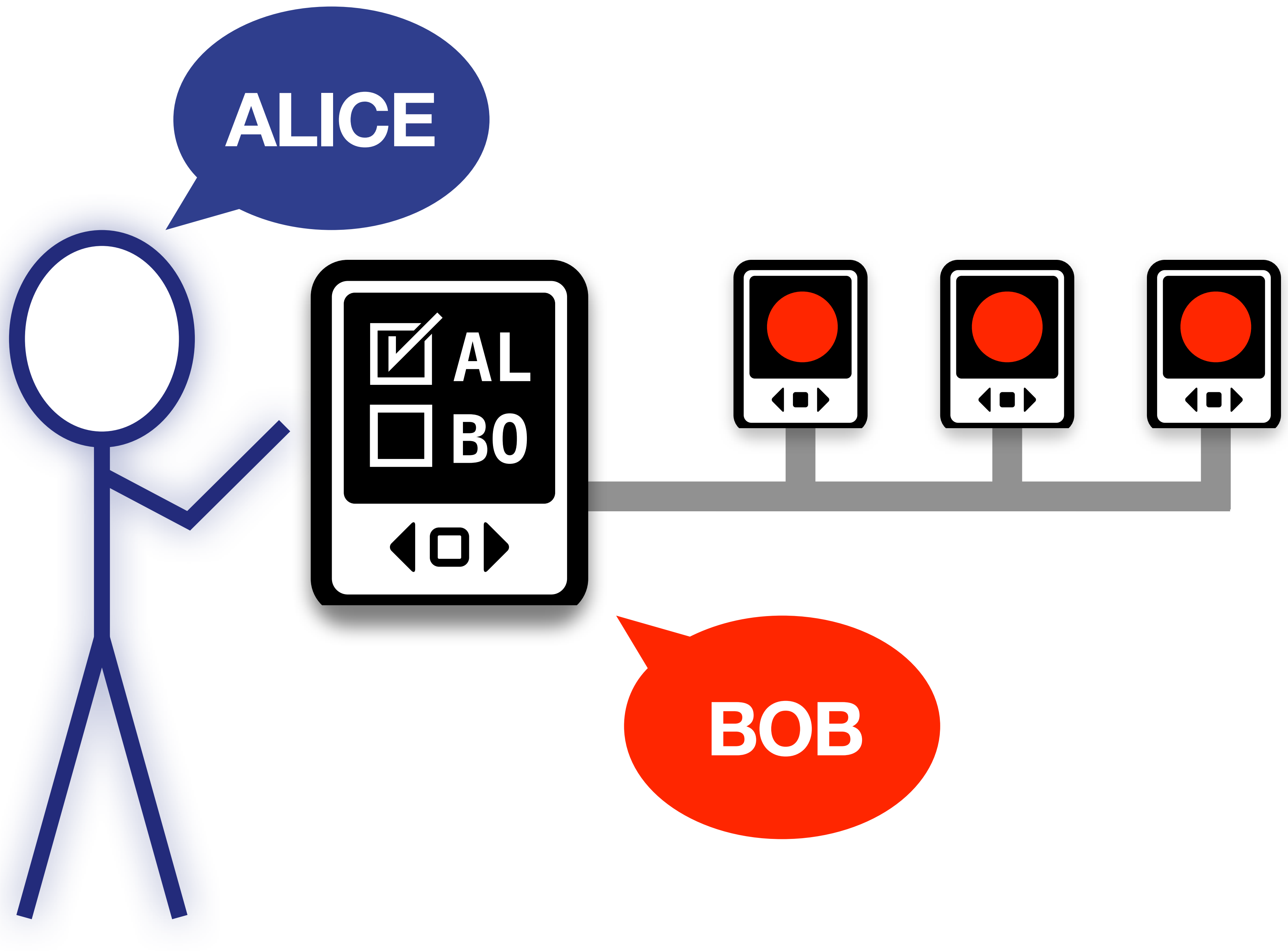



polling place
ALICE
AL
BO

polling place



polling place



this doesn’t work:

“logic & accuracy testing”

VoteBox's approach:

ballot challenge

ballot challenge

ballot challenge

a technique due to [Benaloh '07]

ballot challenge

a technique due to [Benaloh '07]

at the end, instead of casting your ballot:

force the machine to **show it to you**

ballot challenge

a technique due to [Benaloh '07]

at the end, instead of casting your ballot:

force the machine to **show it to you**

this happens on election day

no artificial testing conditions (viz., "L&A tests")

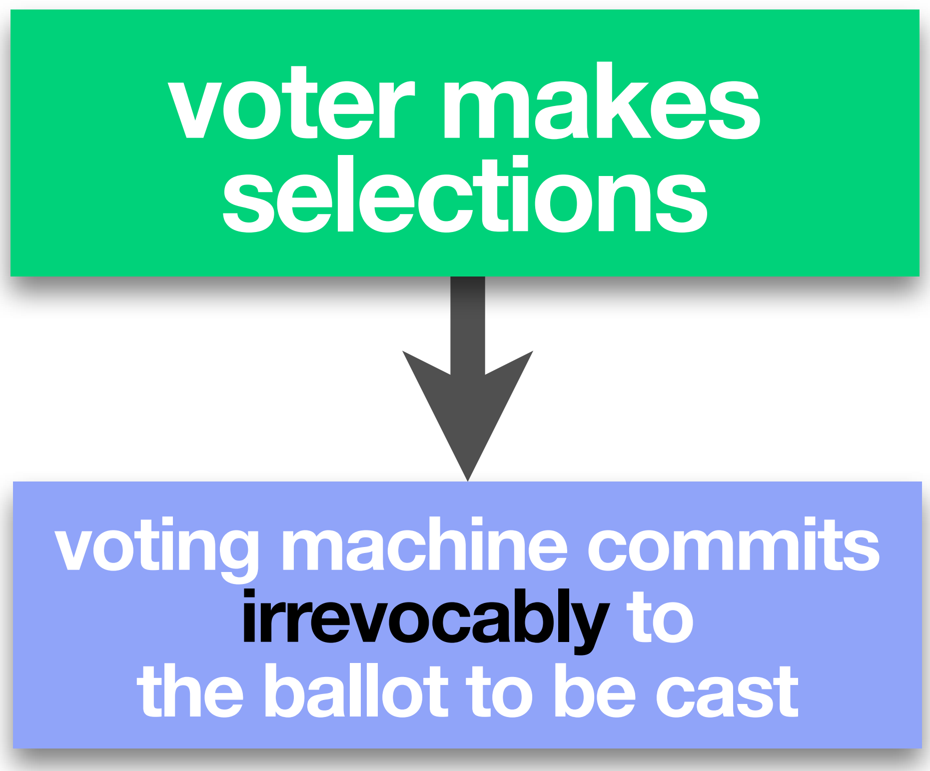
the voting machine cannot distinguish this from a real vote until the challenge

ballot challenge

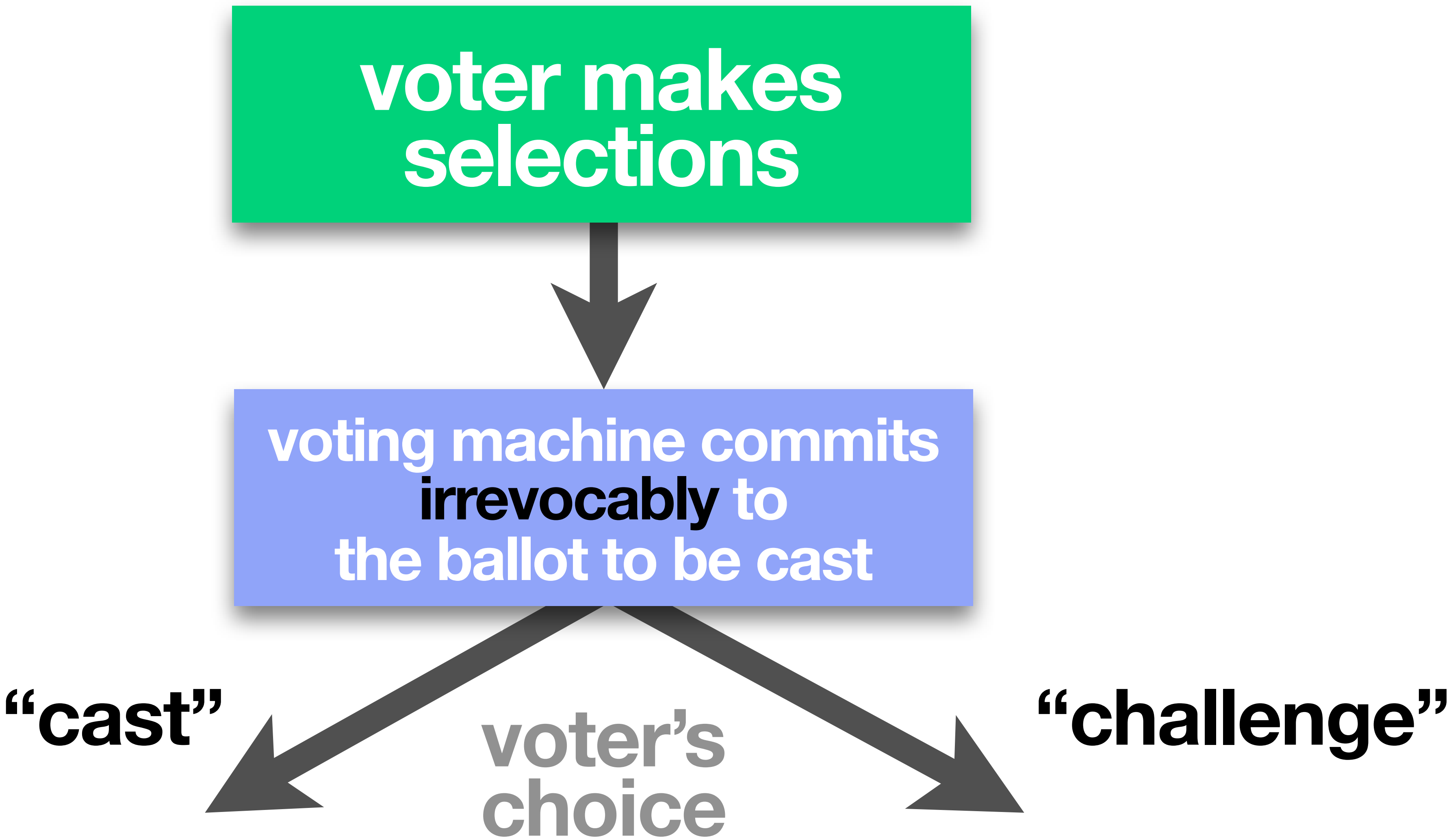
ballot challenge

voter makes selections

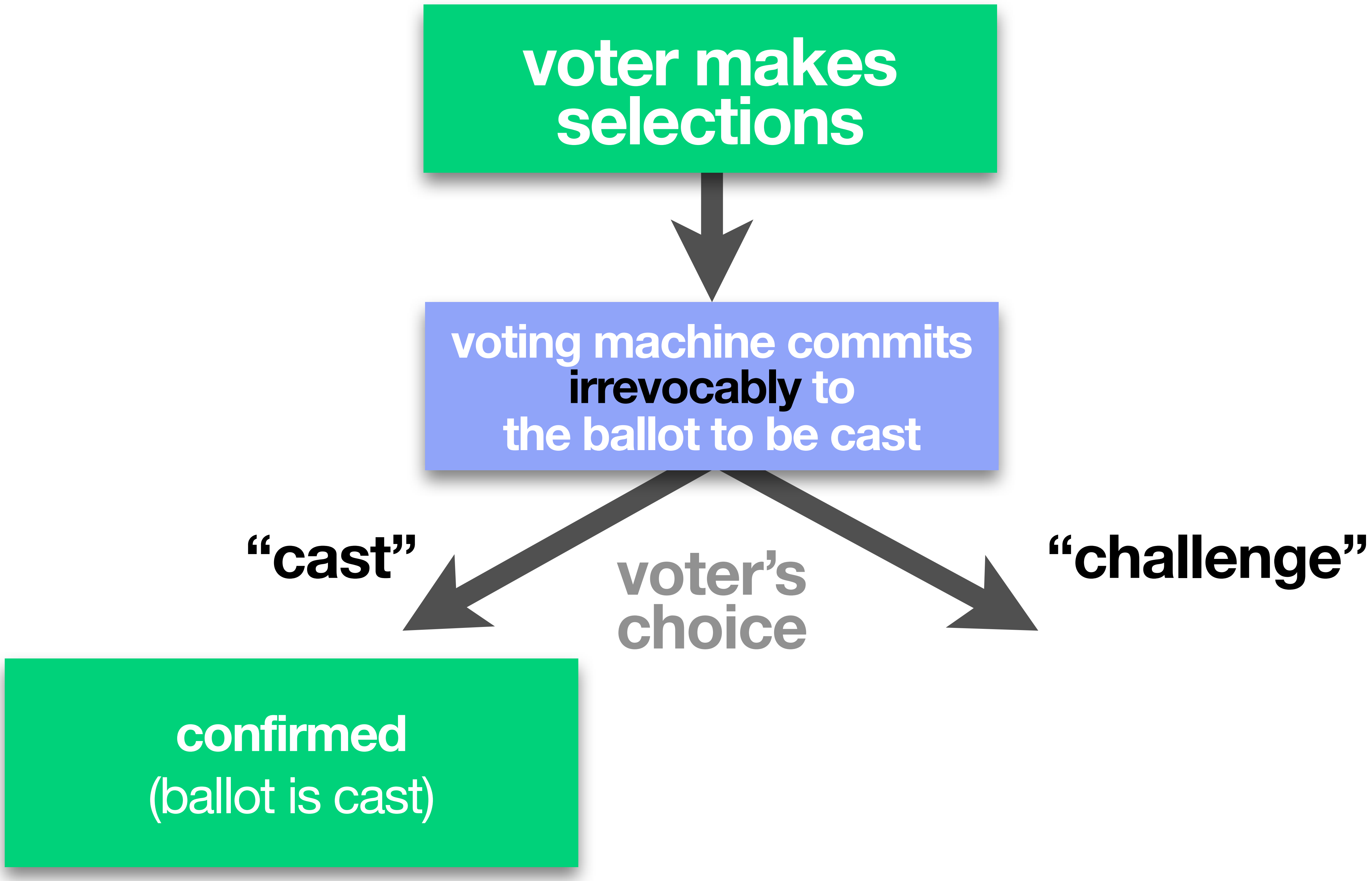
ballot challenge



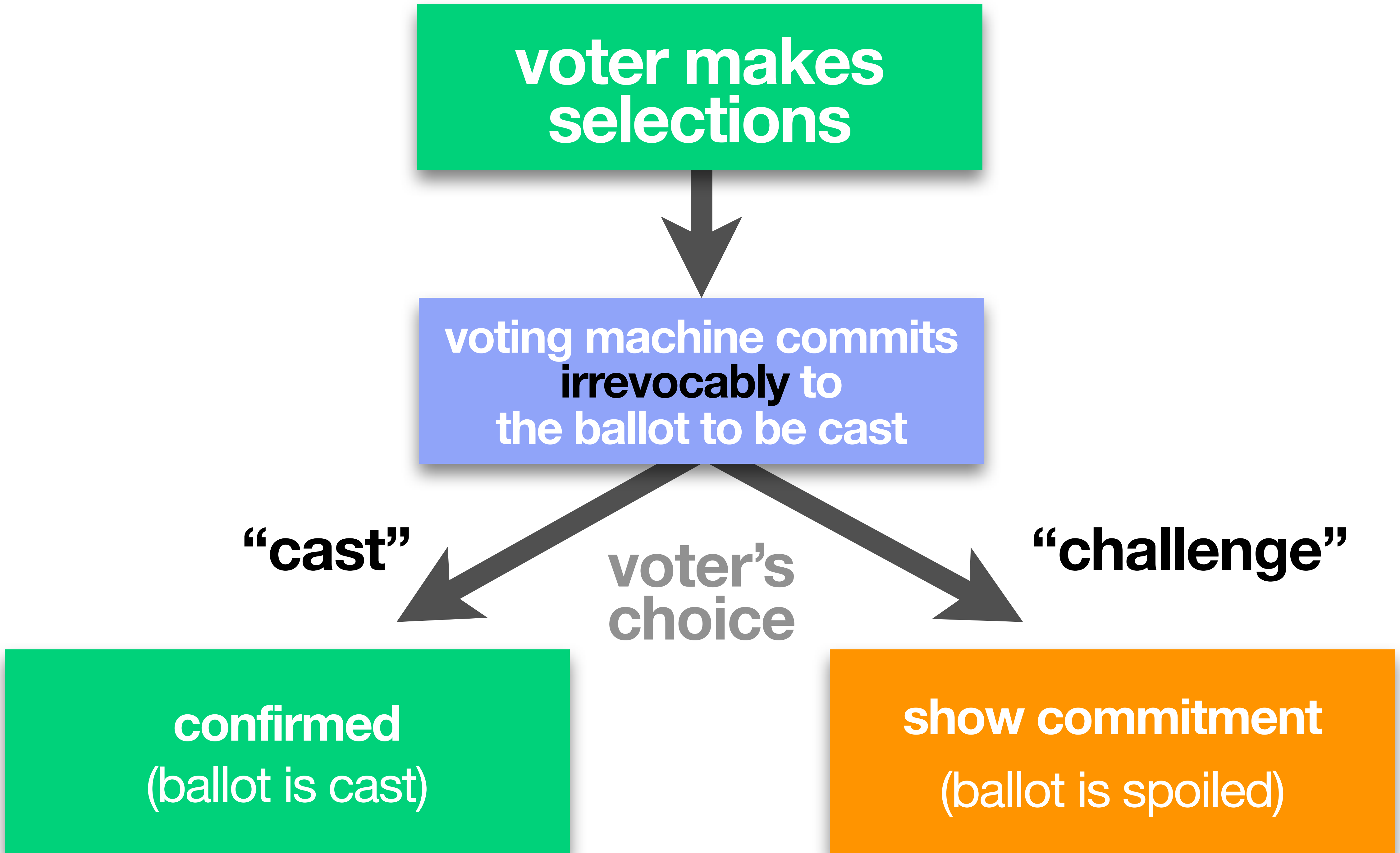
ballot challenge



ballot challenge



ballot challenge



ballot commitment

ballot commitment

What is the commitment?

How do we force the machine to produce proof of what it’s about to cast on the voter’s behalf?

ballot commitment

What is the commitment?

How do we force the machine to produce proof of what it's about to cast on the voter's behalf?

Benaloh's proposal

print the encrypted ballot behind an opaque shield

You can't see the contents, but you can see the page

the computer cannot "un-print" the ballot

ballot commitment

What is the commitment?

How do we force the machine to produce proof of what it's about to cast on the voter's behalf?

Benaloh's proposal

print the encrypted ballot behind an opaque shield

You can't see the contents, but you can see the page

the computer cannot "un-print" the ballot

How do you test the commitment?

ballot commitment

What is the commitment?

How do we force the machine to produce proof of what it's about to cast on the voter's behalf?

Benaloh's proposal

print the encrypted ballot behind an opaque shield

You can't see the contents, but you can see the page

the computer cannot "un-print" the ballot

How do you test the commitment?

Decrypt it.

But decryption requires the private key for tabulating the whole election!

Elgamal reminder

Two ways to decrypt:

$$
\begin{aligned}
E(g^{a}, r, M) &= \langle g^{r}, (g^{a})^{r} M \rangle \\
D(g^{r}, g^{ar}M, a) &= \frac{g^{ar}M}{(g^{r})^{a}} \\
D(g^{r}, g^{ar}M, r) &= \frac{g^{ar}M}{(g^{a})^{r}} \\
&= M
\end{aligned}
$$

g	group generator
M	plaintext (message)
r	random (chosen at encryption time)
a	(private) decryption key
g^{a}	(public) encryption key

challenging the machine

challenging the machine

When challenged, the machine must reveal *r*

We can then decrypt this ballot (only) and see if it's what we expected to see

In Benaloh, the encrypted ballot is on paper

An **irrevocable** output medium

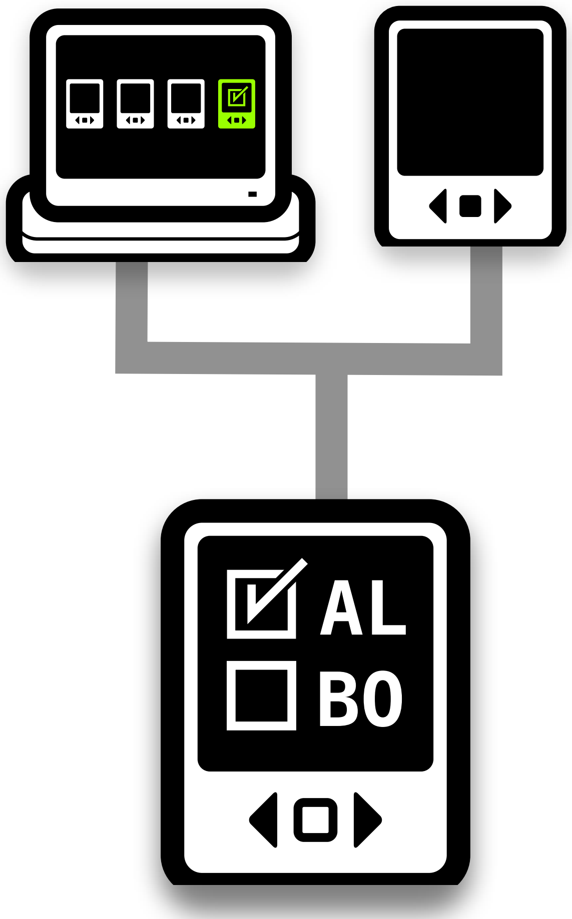
decrypting requires additional equipment

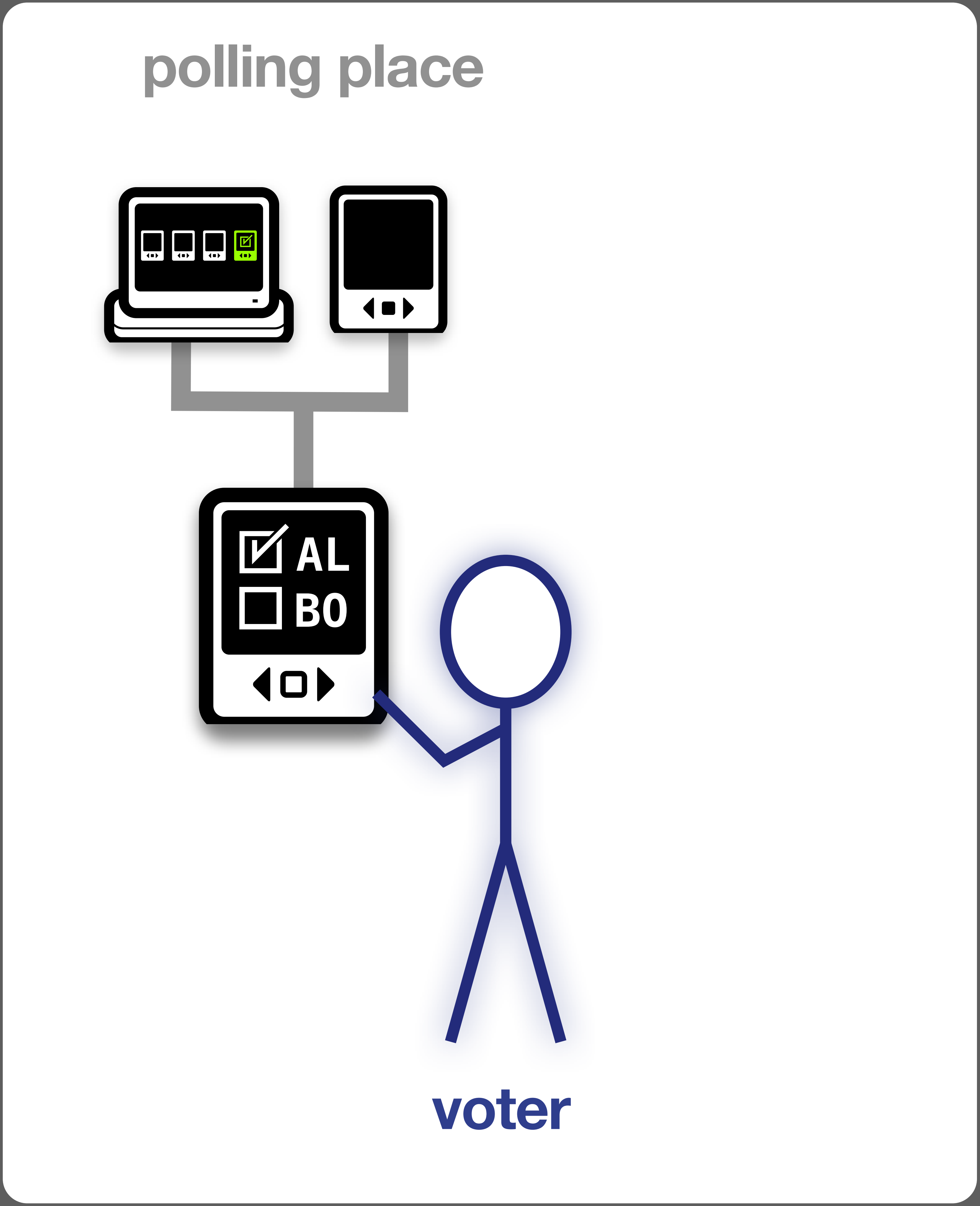
VoteBox happens to have its own irrevocable publishing system

One that doesn't run out of ink or paper

Auditorium.

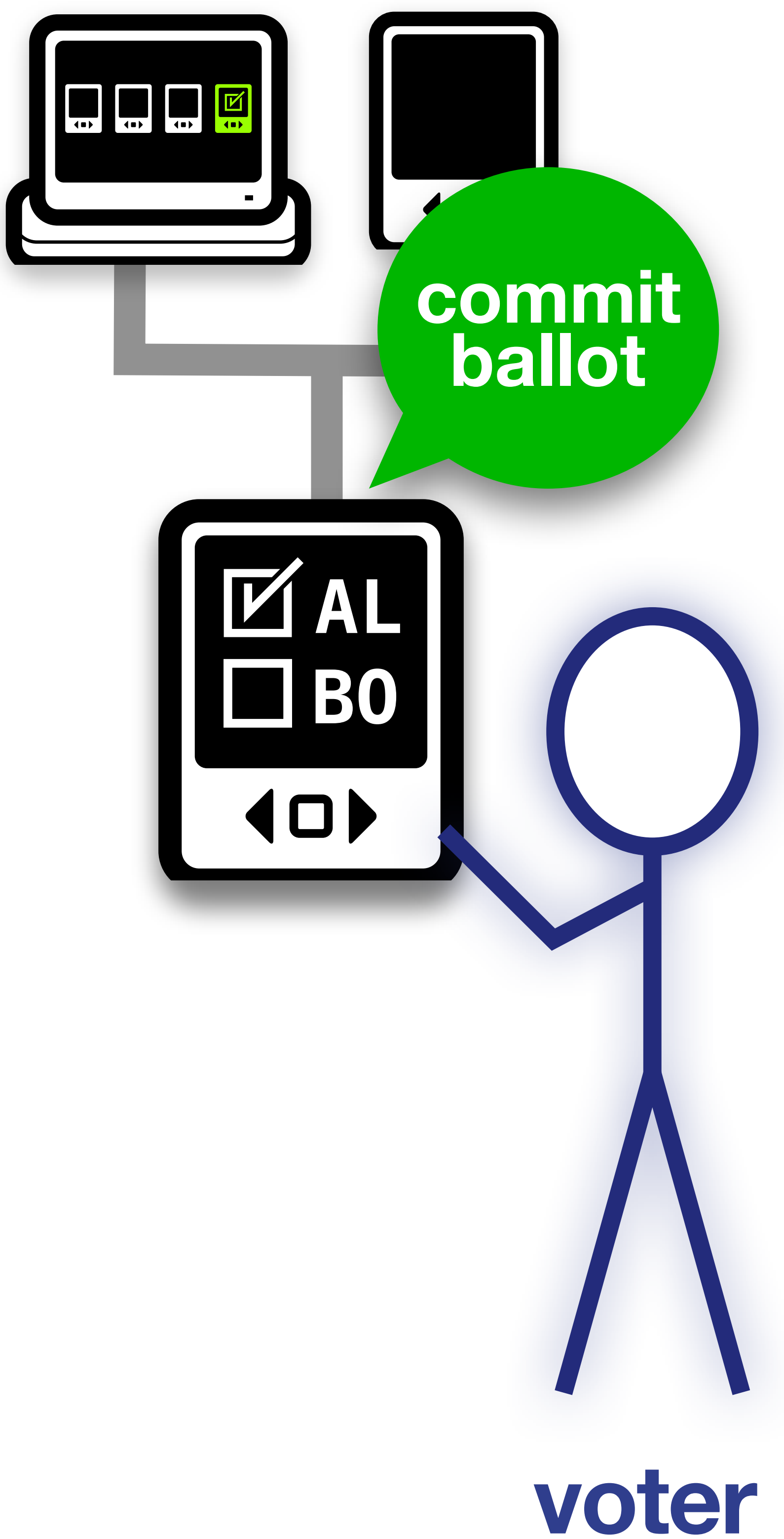
polling place



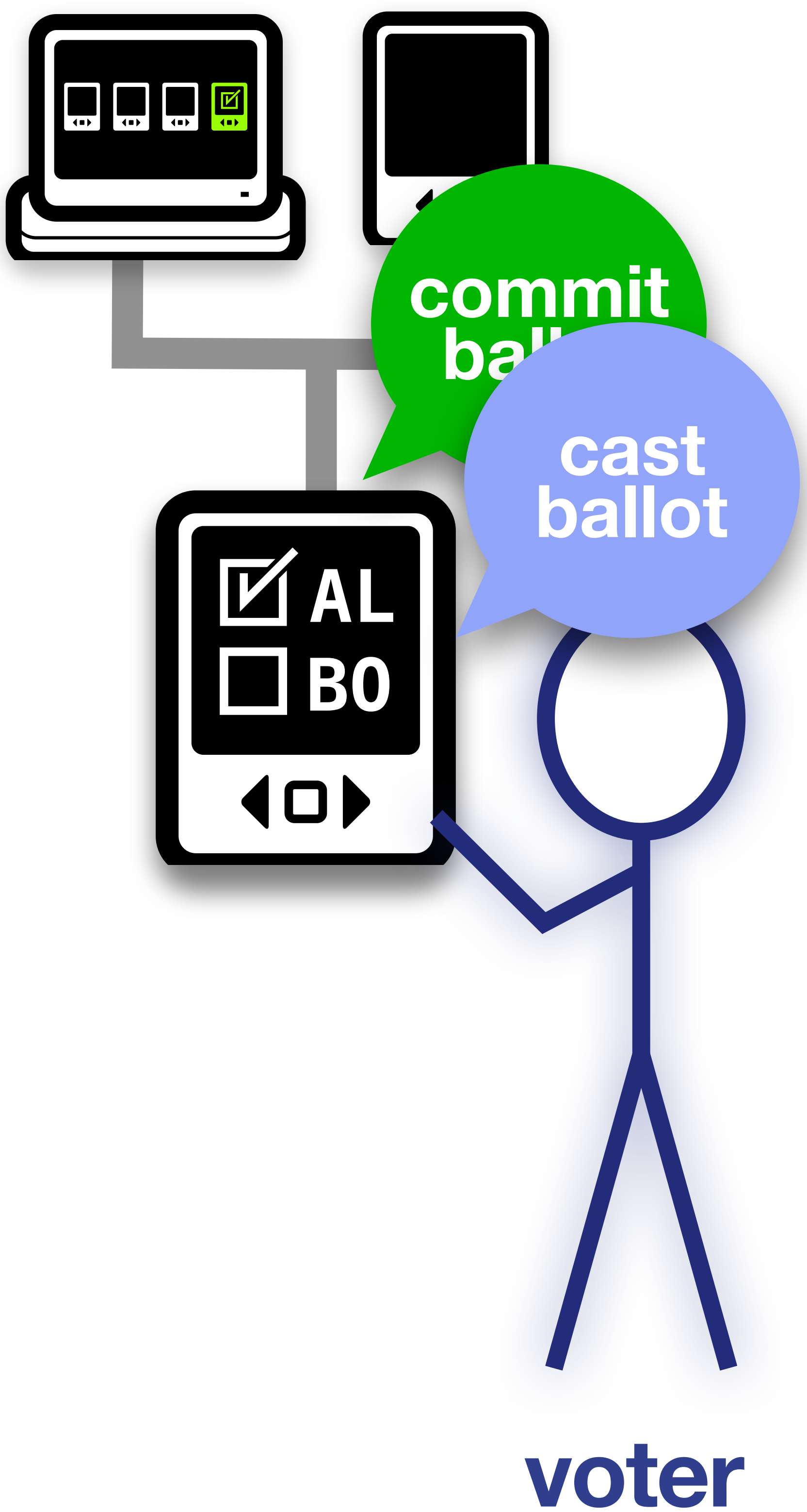


polling place
AL
BO
voter

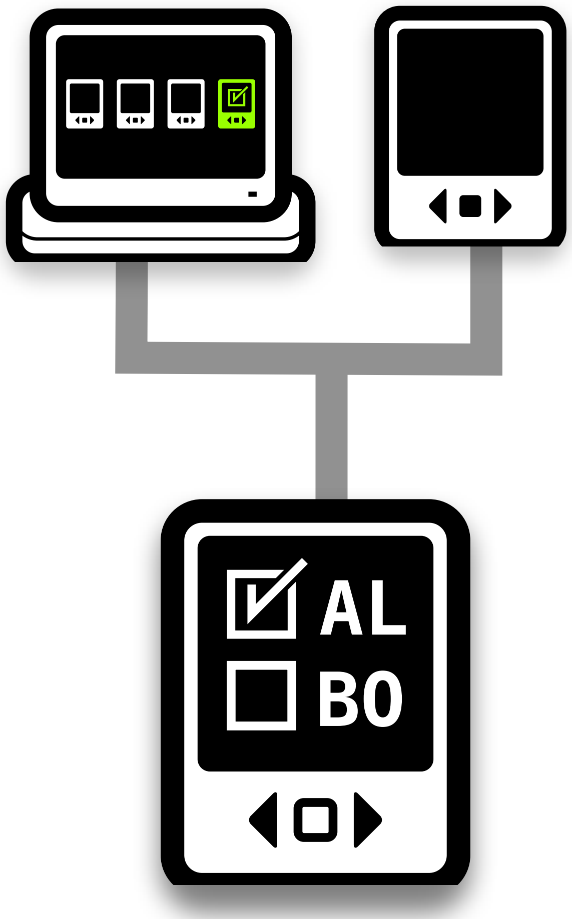
polling place

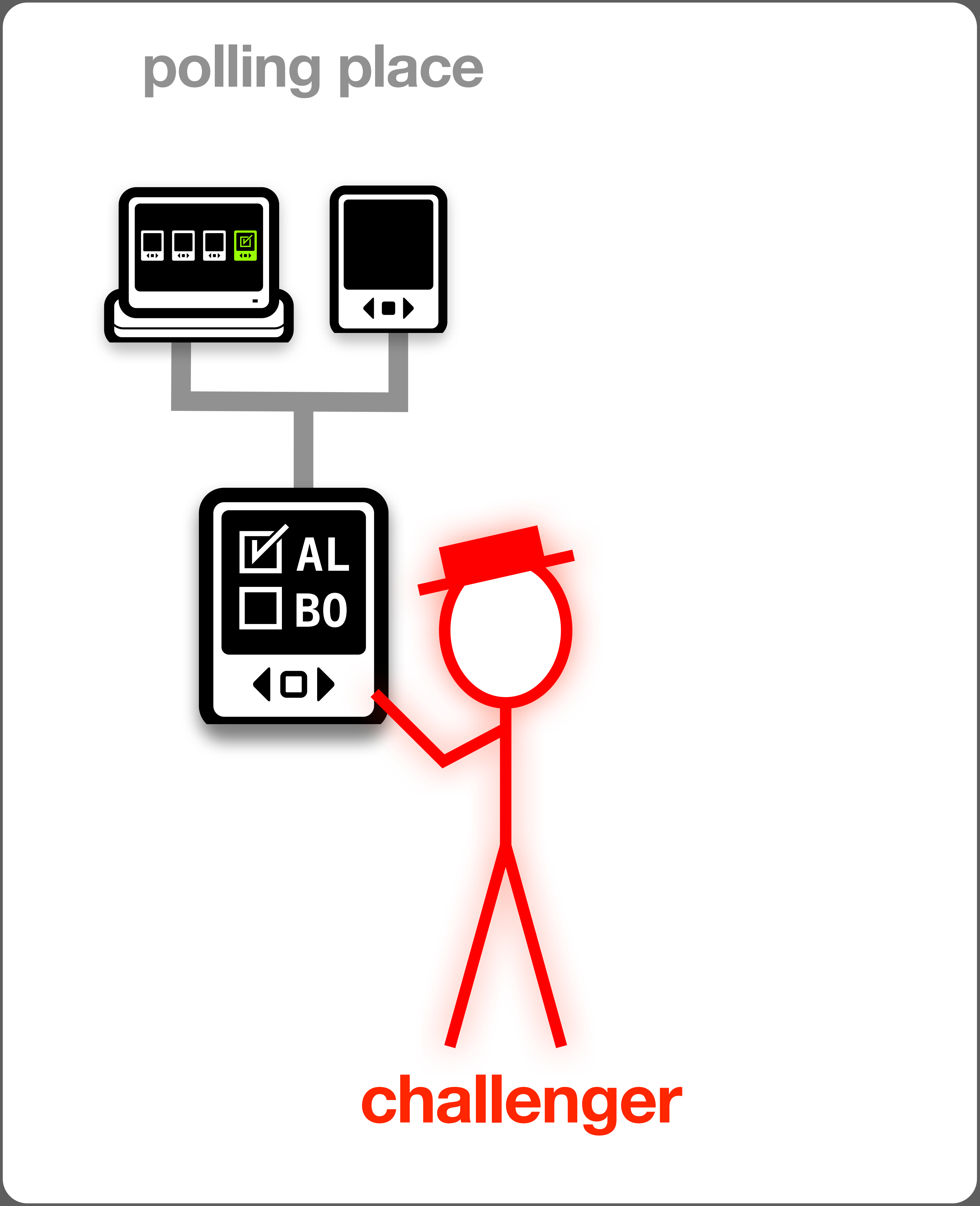


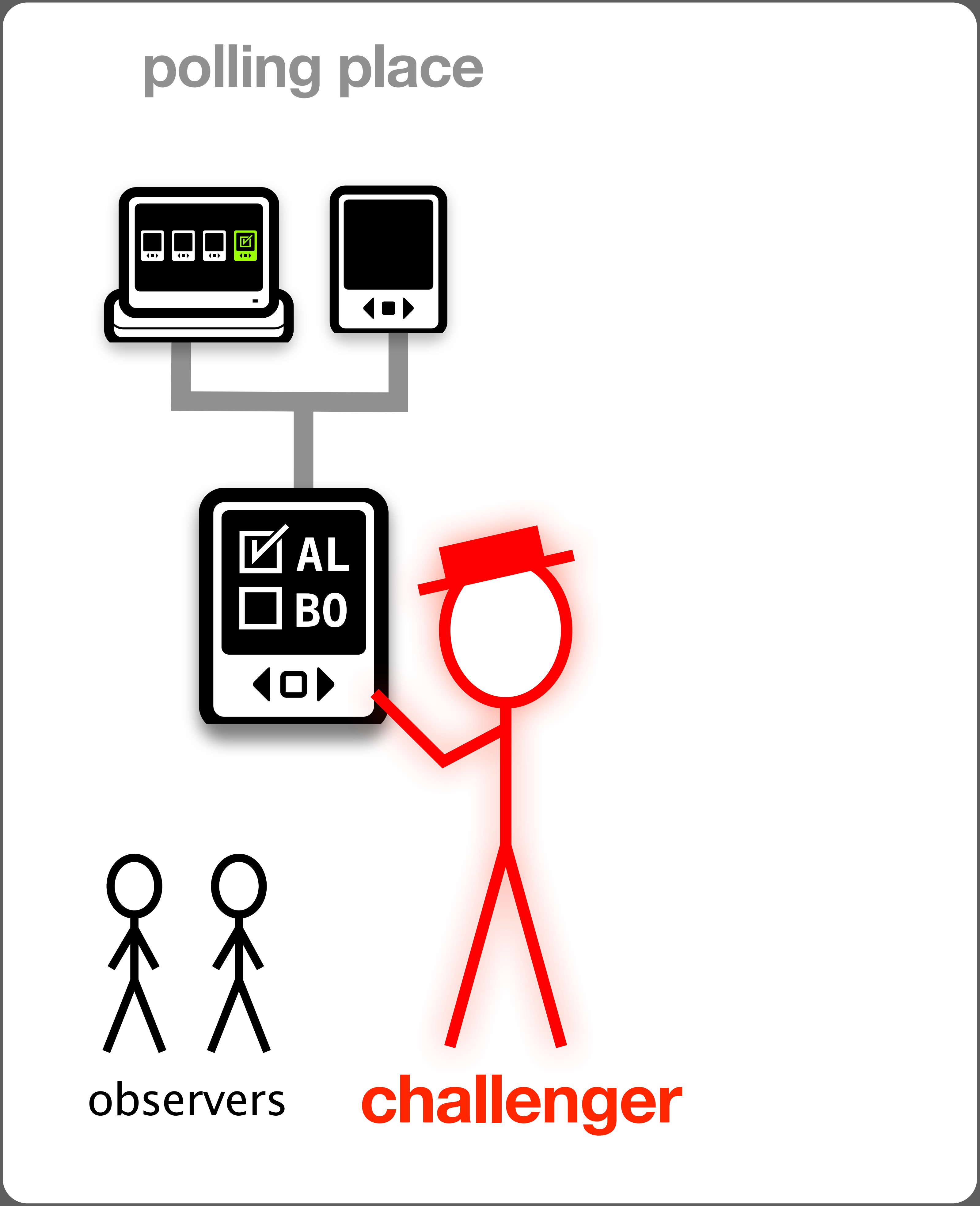
polling place

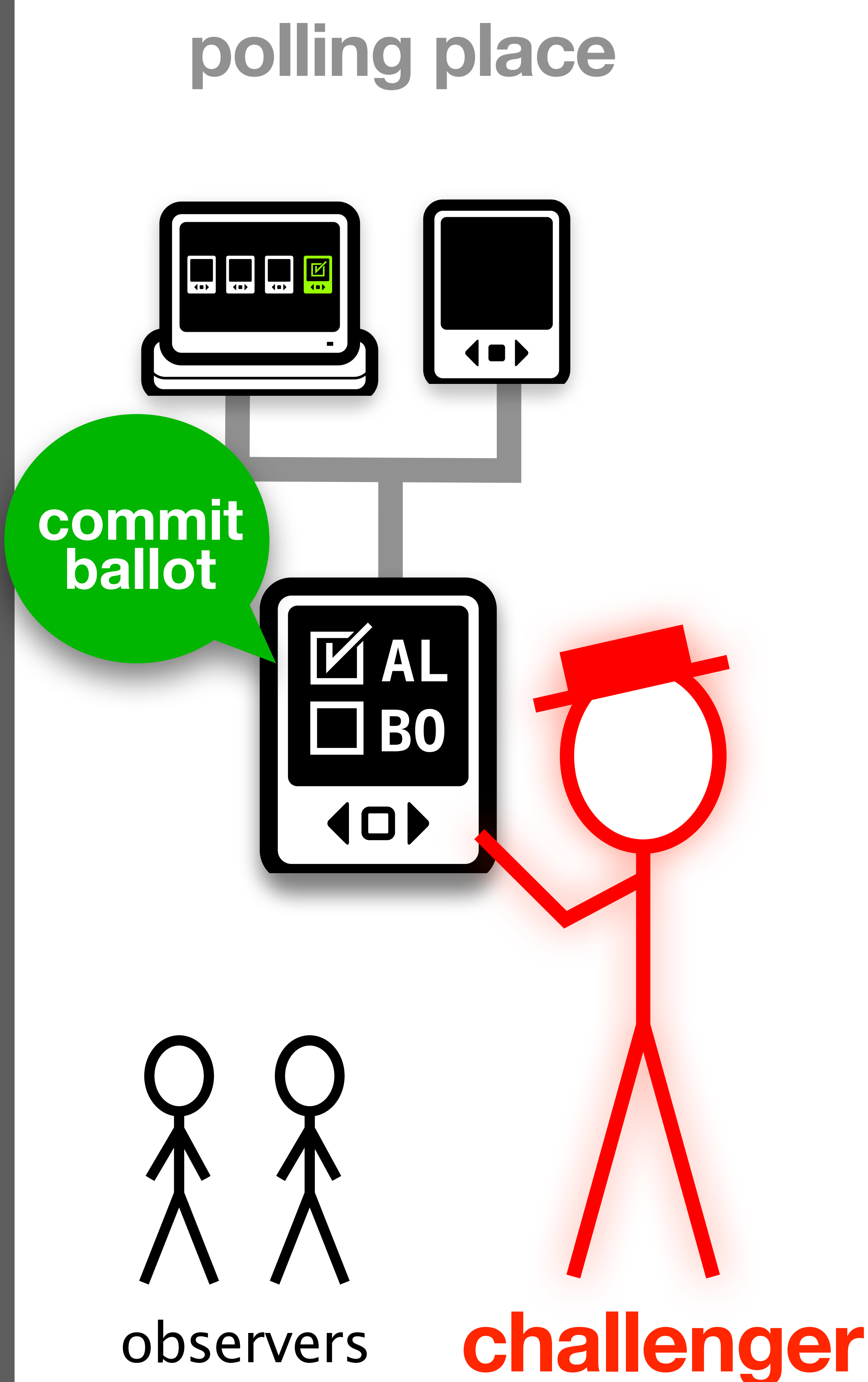


polling place

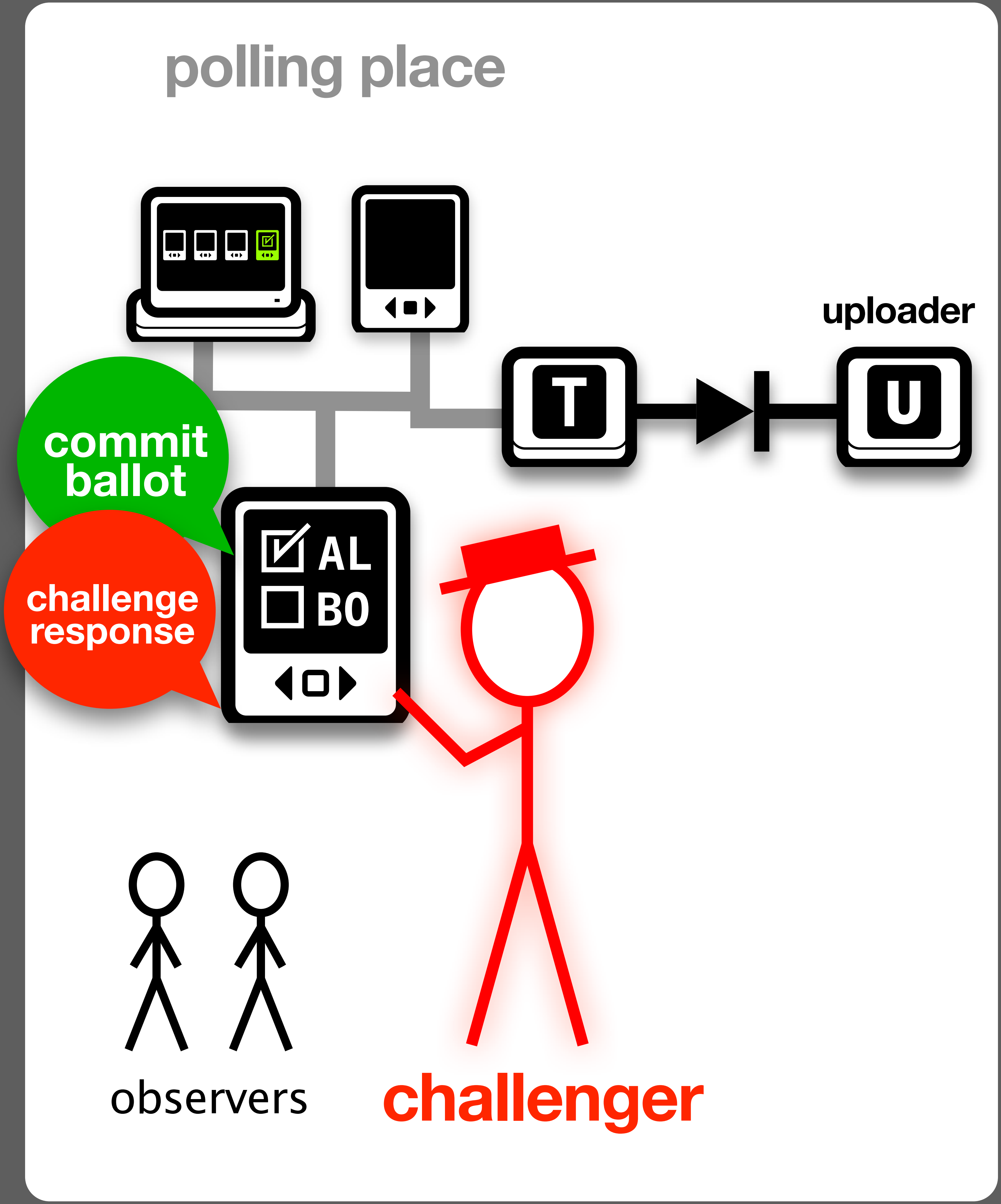


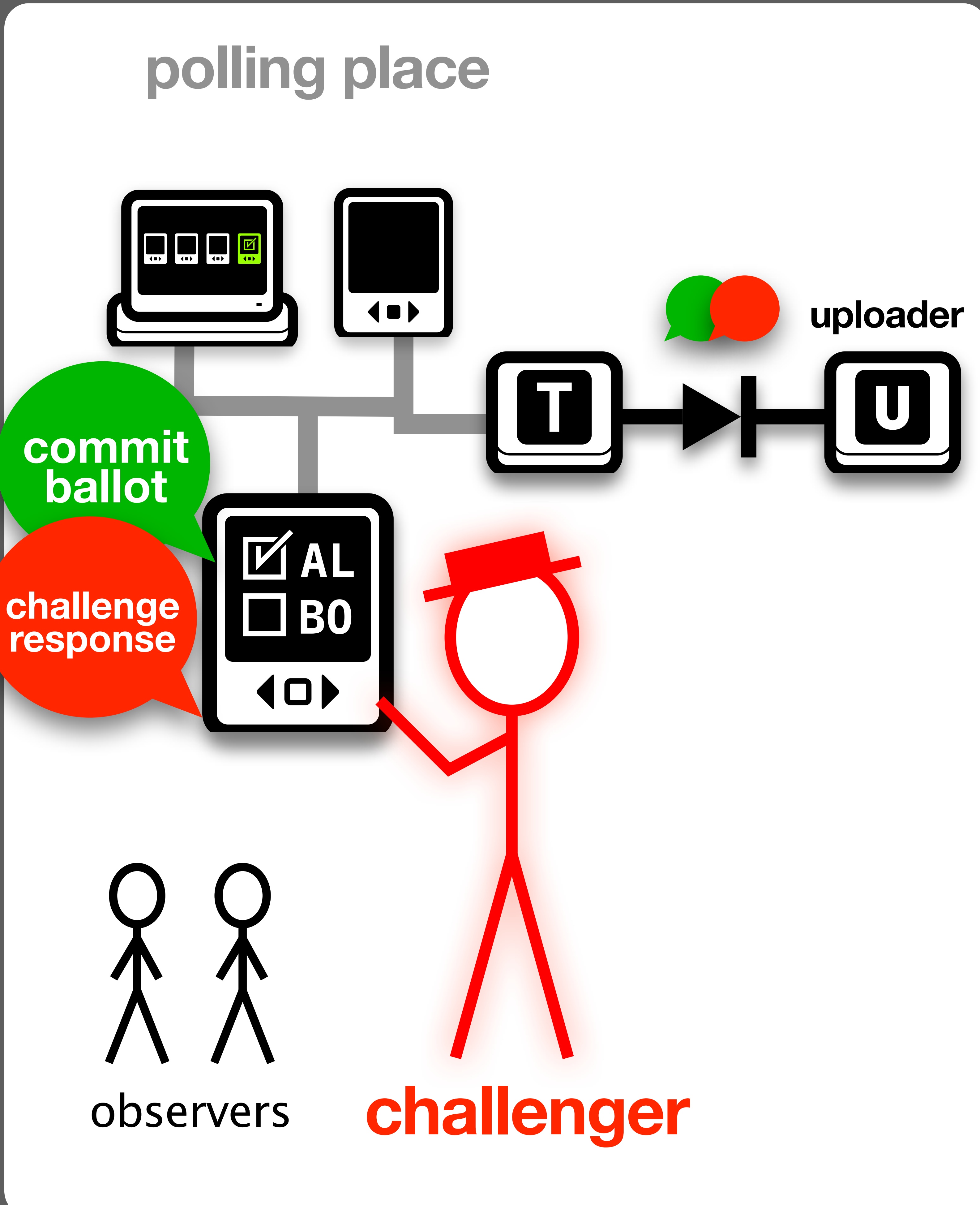

polling place
AL
BO
challenger

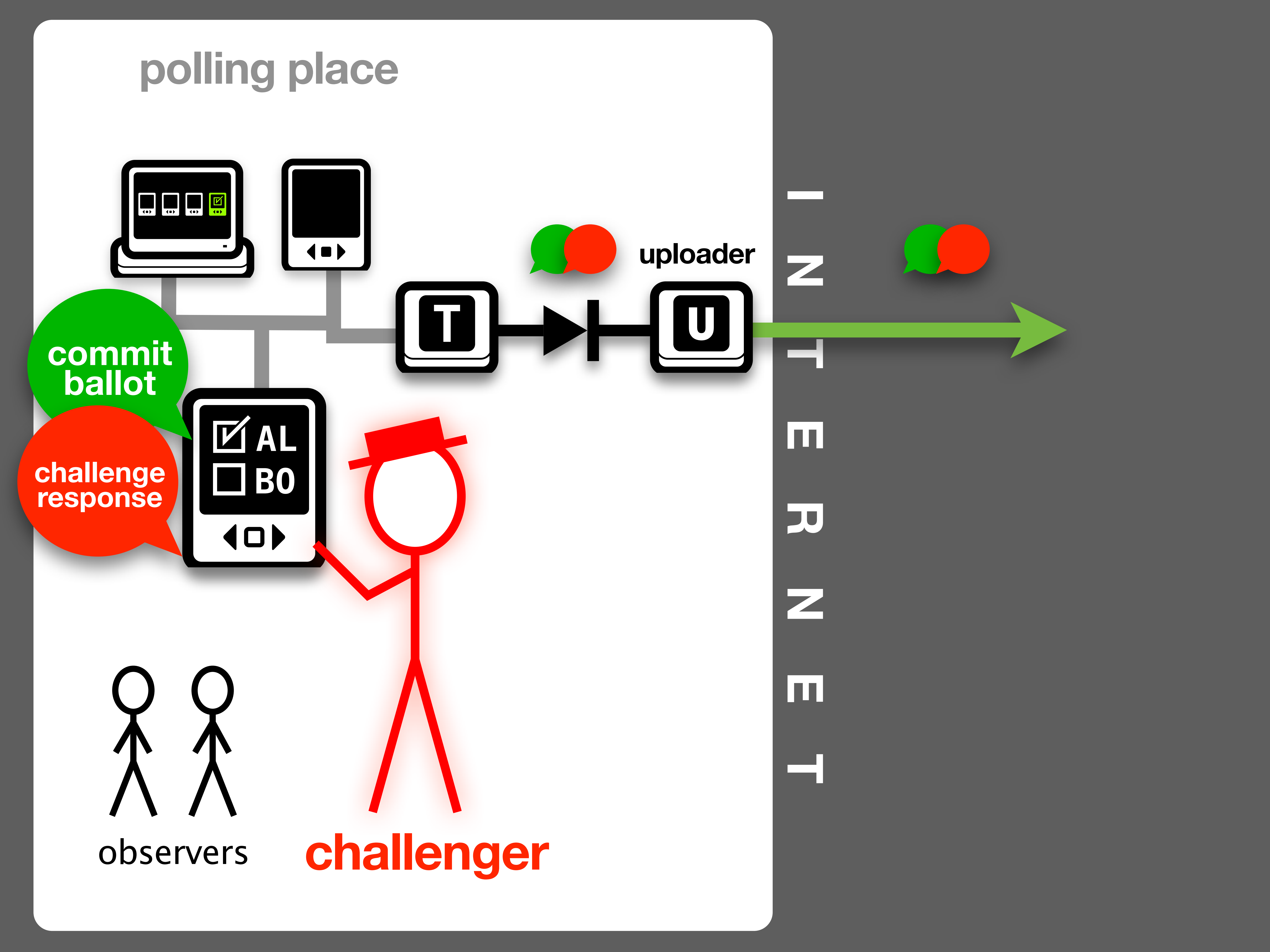

polling place
AL
BO
observers
challenger


polling place
commit ballot
AL
BO
observers
challenger

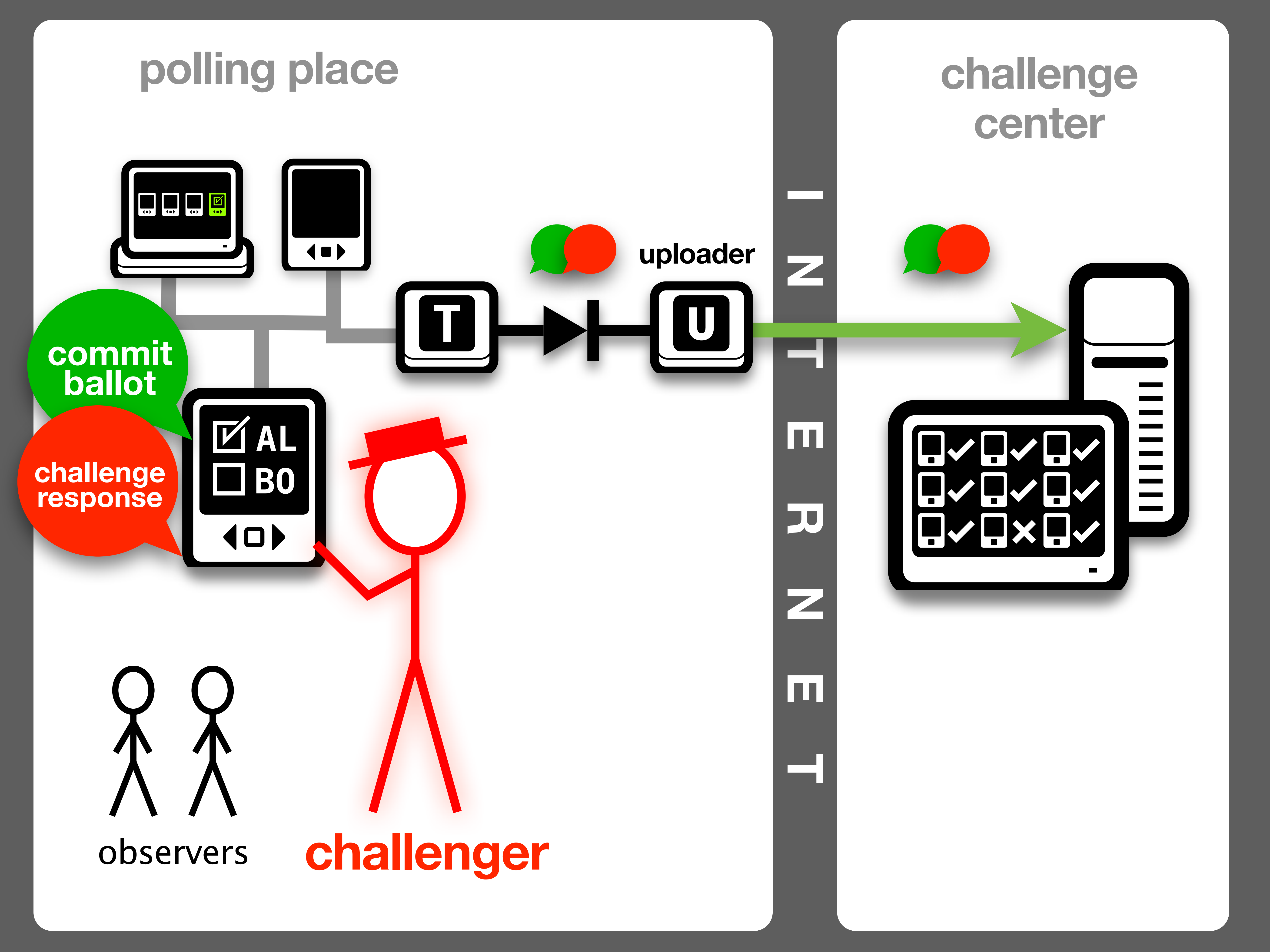

polling place
commit ballot
challenge response
AL
BO
observers
challenger


polling place
uploader
T
U
commit ballot
challenge response
AL
BO
observers
challenger

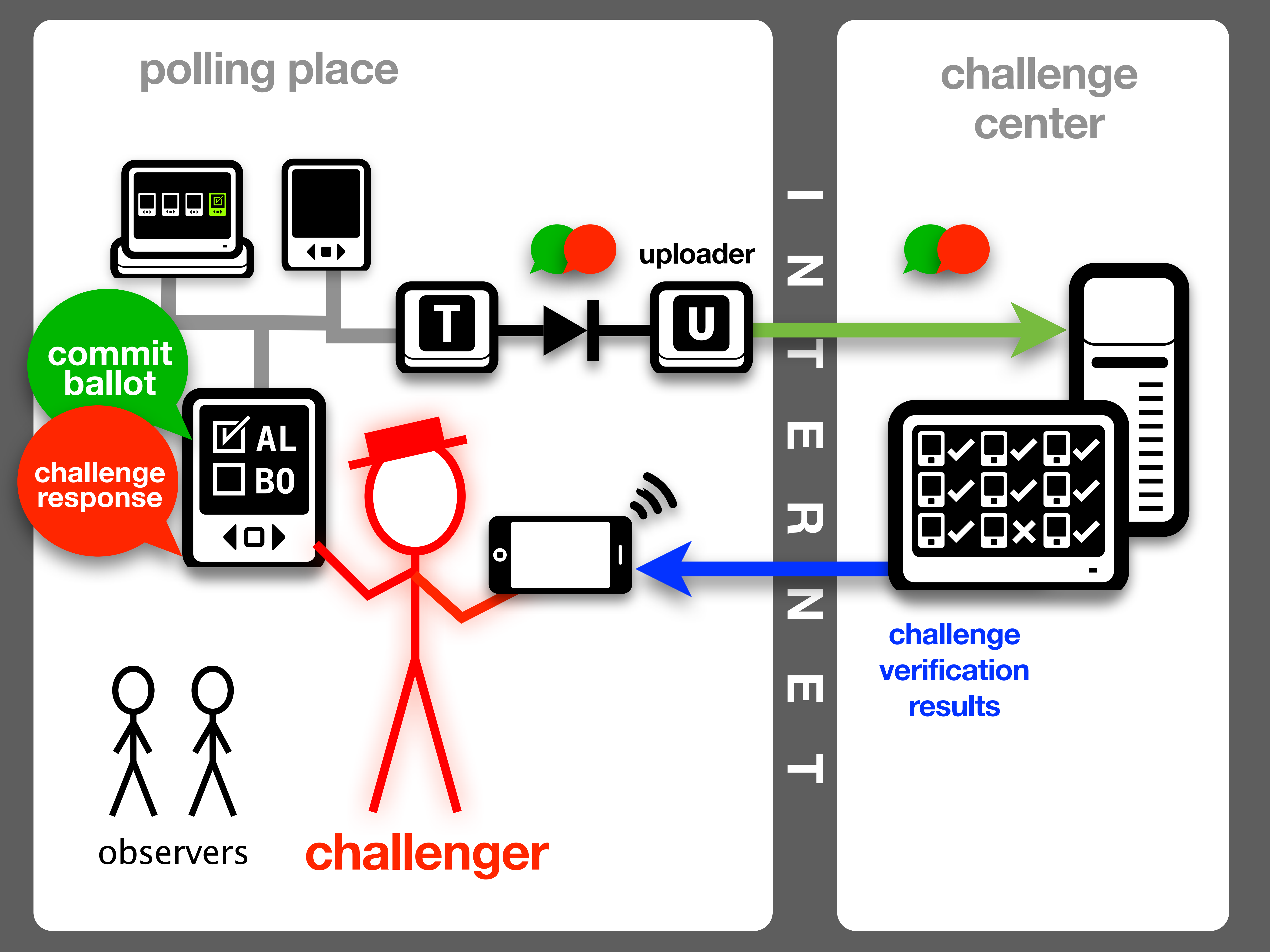

polling place
uploader
T
U
commit ballot
challenge response
AL
BO
observers
challenger

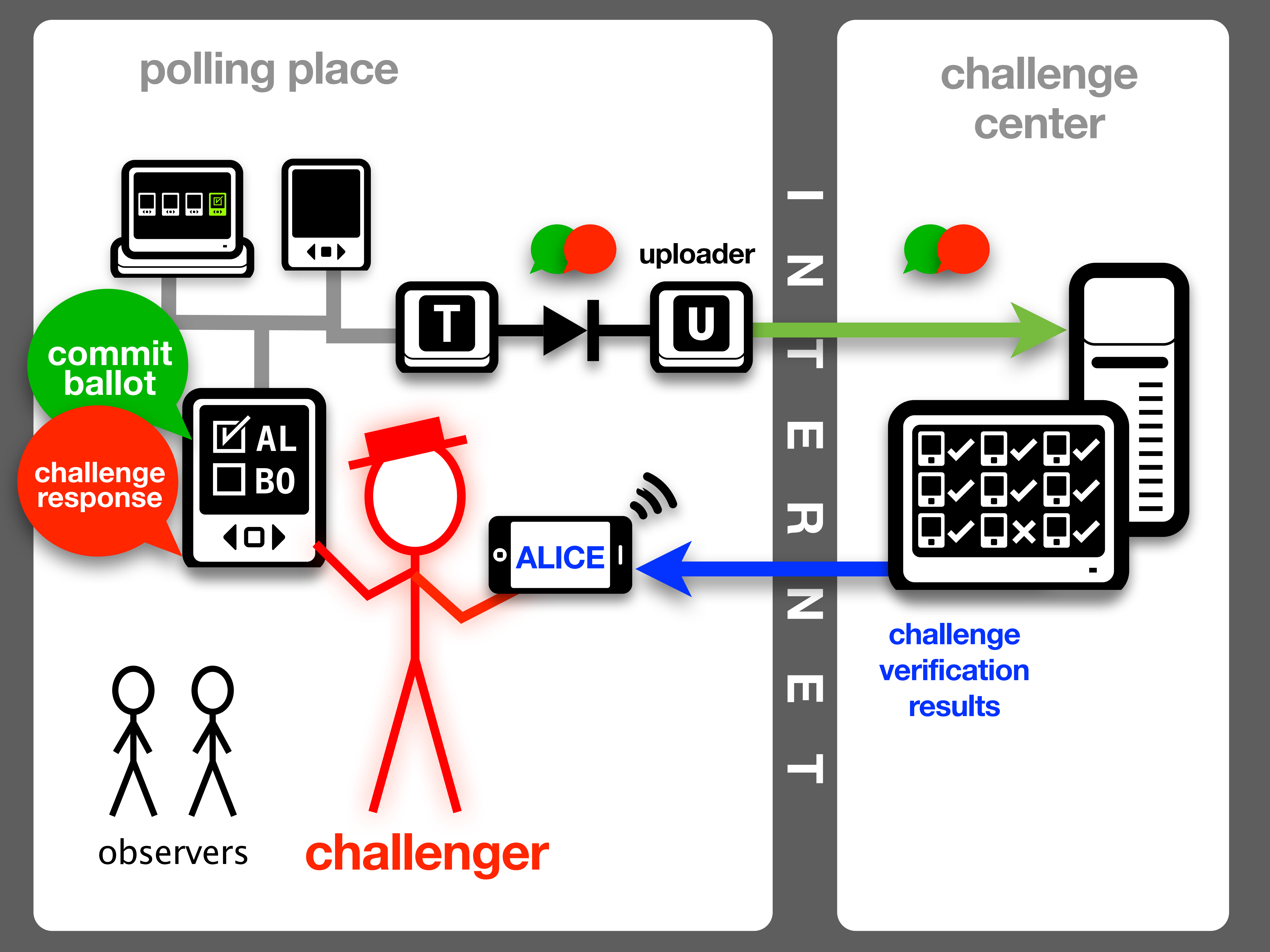


polling place
commit ballot
challenge response
AL
BO
T
U
uploader
INTERNET
observers
challenger



polling place
challenge center
commit ballot
challenge response
AL
BO
T
U
uploader
INTERNET
observers
challenger


polling place
challenge center
uploader
T
U
INTERNET
commit ballot
challenge response
AL
BO
challenge verification results
observers
challenger



polling place
challenge center
uploader
T
U
INTERNET
commit ballot
challenge response
AL
BO
ALICE
challenge verification results
observers
challenger

Challenges in Auditorium

Challenges in Auditorium

When challenged,

Challenges in Auditorium

When challenged,

a VoteBox must **announce** ***r*** **on the network**

Challenges in Auditorium

When challenged,

a VoteBox must **announce *r* on the network**

Irrevocable thanks to the properties of Auditorium

Challenges in Auditorium

When challenged,

a VoteBox must **announce *r* on the network**

Irrevocable thanks to the properties of Auditorium

We still need help decrypting the commitment, even given ***r***

Challenges in Auditorium

When challenged,

a VoteBox must **announce *r* on the network**

Irrevocable thanks to the properties of Auditorium

We still need help decrypting the commitment, even given ***r***

If we are careful, we can send challenges offsite

Challenges in Auditorium

When challenged,

a VoteBox must **announce *r* on the network**

Irrevocable thanks to the properties of Auditorium

We still need help decrypting the commitment, even given ***r***

If we are careful, we can send challenges offsite

Allow a third party to assist in verifying the challenge

Challenges in Auditorium

When challenged,

a VoteBox must **announce *r* on the network**

Irrevocable thanks to the properties of Auditorium

We still need help decrypting the commitment, even given ***r***

If we are careful, we can send challenges offsite

Allow a third party to assist in verifying the challenge

Trusted by the challenger!

Ballot challenges:

Ballot challenges:
cast-as-intended verification

Ballot challenges:
cast-as-intended verification
preserving privacy

Ballot challenges:
cast-as-intended verification
preserving privacy
without artificial test conditions.

UI.

skip

pre-rendered user interfaces

pre-rendered user interfaces

very restricted UI functions

pre-rendered user interfaces

very restricted UI functions

`next_event()` → keyboard or (x, y) input

pre-rendered user interfaces

very restricted UI functions

`next_event()` → keyboard or (x, y) input

`blit(bitmap, x, y)`

pre-rendered user interfaces

very restricted UI functions

`next_event``()` → keyboard or (x, y) input

`blit``(bitmap, x, y)`

what's not here?

pre-rendered user interfaces

very restricted UI functions

`next_event``()` → keyboard or (x, y) input

`blit``(bitmap, x, y)`

what's not here?

windowing system; widgets; fonts & text rendering

pre-rendered user interfaces

very restricted UI functions

`next_event()` → keyboard or (x, y) input

`blit``(bitmap, x, y)`

what's not here?

windowing system; widgets; fonts & text rendering

result

pre-rendered user interfaces

very restricted UI functions

`next_event()` → keyboard or (x, y) input

`blit(bitmap, x, y)`

what's not here?

windowing system; widgets; fonts & text rendering

result

less code to inspect, certify, and trust

pre-rendered user interfaces

very restricted UI functions

`next_event()` → keyboard or (x, y) input

`blit(bitmap, x, y)`

what's not here?

windowing system; widgets; fonts & text rendering

result

less code to inspect, certify, and trust

inspiration: Pvote

pioneering work on PRUI in e-voting

[Yee, EVT '06 & '07]

STEP 1
Read Instructions

You are now on
STEP 2
Make your choices

STEP 3
Review your choices

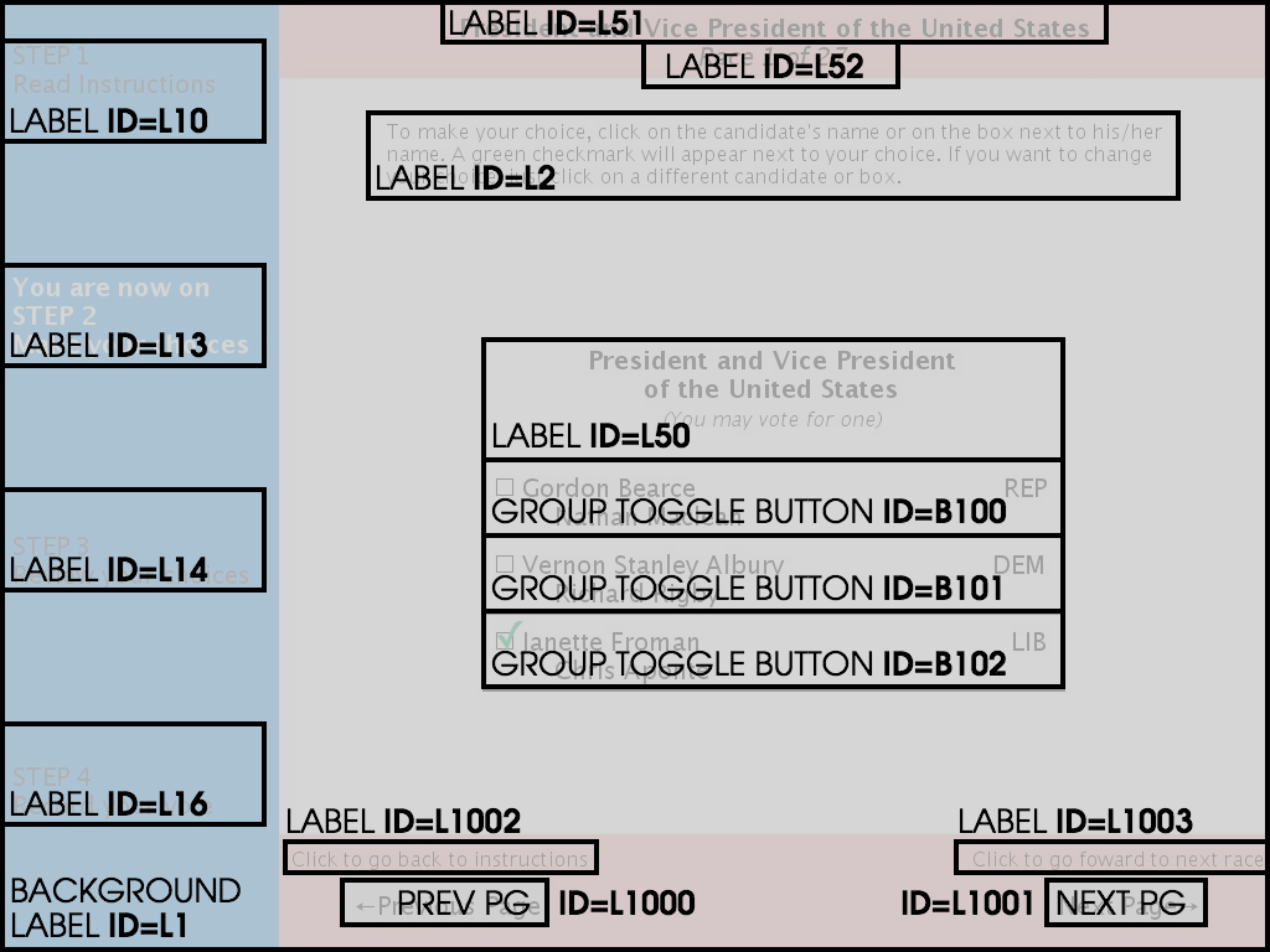
STEP 4
Record your vote

To make your choice, click on the candidate's name or on the box next to his/her name. A green checkmark will appear next to your choice. If you want to change your choice, just click on a different candidate or box.

President and Vice President of the United States *(You may vote for one)*	
☐ Gordon Bearce Nathan Maclean	REP
☐ Vernon Stanley Albury Richard Rigby	DEM
☑ Janette Froman Chris Aponte	LIB

Click to go back to instructions
←Previous Page

Click to go foward to next race
Next Page→


LABEL ID=L51
Vice President of the United States
LABEL ID=L52
STEP 1
Read Instructions
LABEL ID=L10
To make your choice, click on the candidate's name or on the box next to his/her name. A green checkmark will appear next to your choice. If you want to change
click on a different candidate or box.
LABEL ID=L2
You are now on
STEP 2
LABEL ID=L13
President and Vice President
of the United States
(You may vote for one)
LABEL ID=L50
Gordon Bearce
REP
GROUP TOGGLE BUTTON ID=B100
Vernon Stanley Albury
DEM
GROUP TOGGLE BUTTON ID=B101
Janette Froman
LIB
GROUP TOGGLE BUTTON ID=B102
STEP 3
LABEL ID=L14
STEP 4
LABEL ID=L16
LABEL ID=L1002
Click to go back to instructions
LABEL ID=L1003
Click to go foward to next race
PREV PG
ID=L1000
ID=L1001
NEXT PG
BACKGROUND
LABEL ID=L1

VoteBox ballot creator

GUI tool for creating pre-rendered ballots

this is where the complexity went

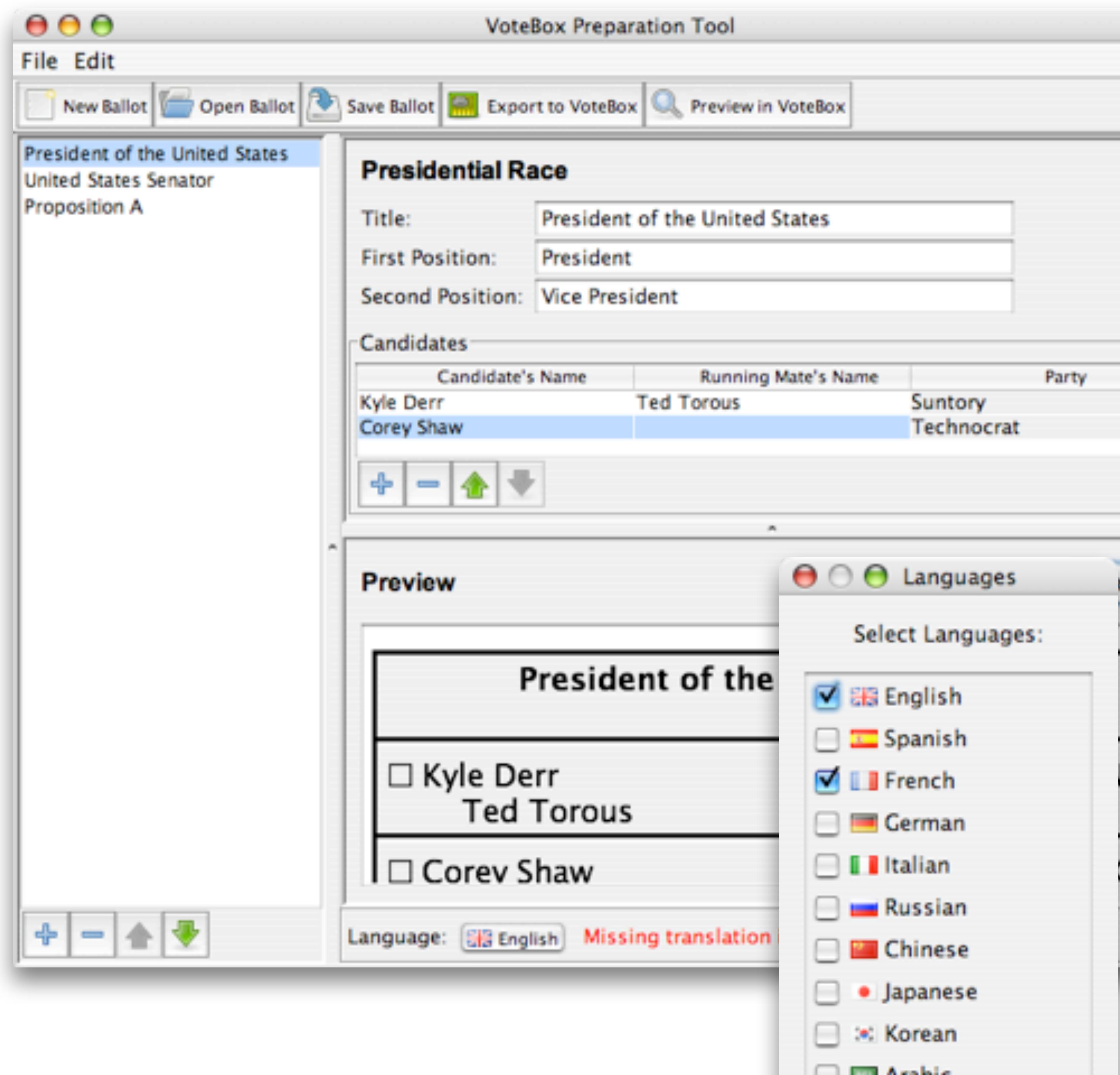
Not in the TCB

we don't need to trust this software

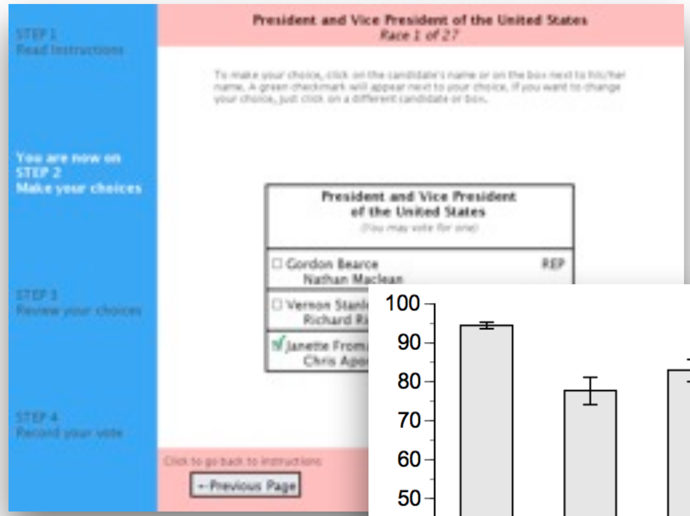
sufficient to verify that the output ballot is correct

Flexibility

New ballot designs do not require changes to VoteBox—only the ballot creator



HCI research



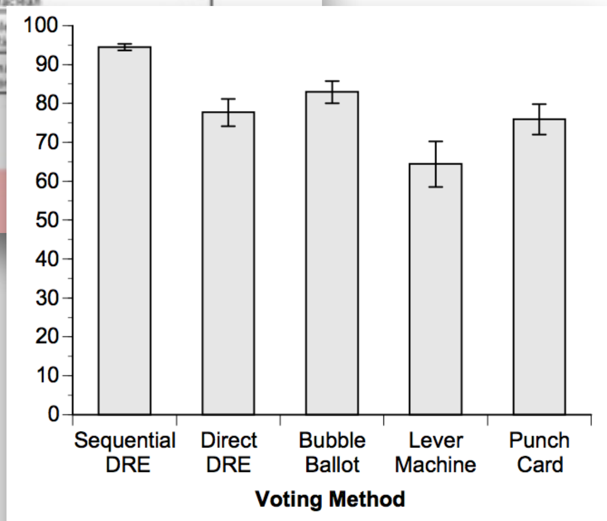
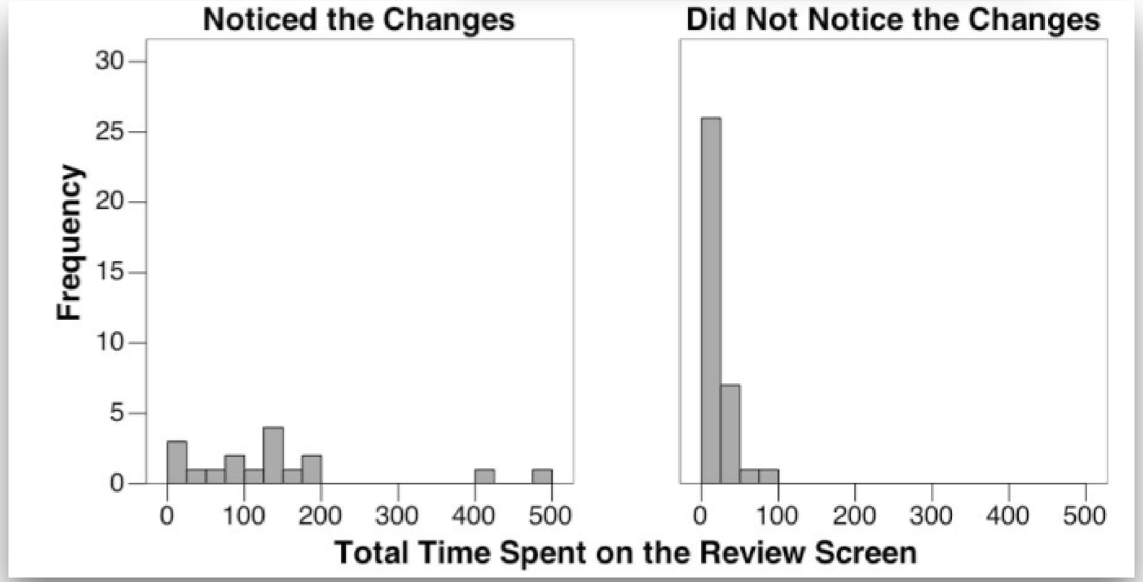


Figure 5. Mean SUS ratings by voting method.

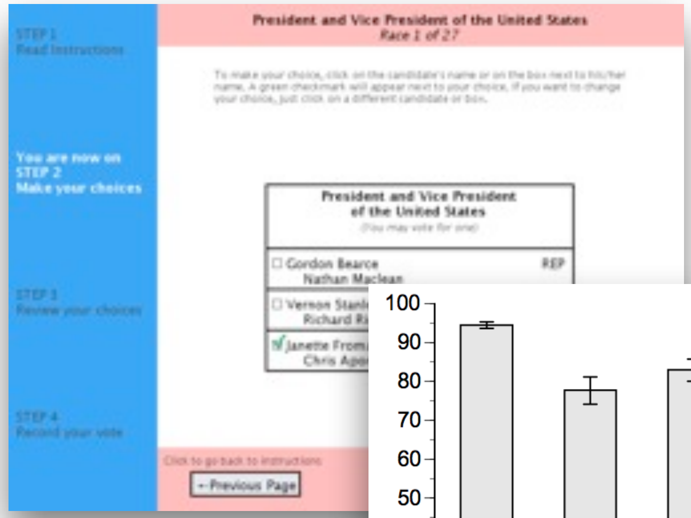


HCI research

VoteBox is a platform for human factors research & experimentation

VoteBox's ballot designed jointly with Rice CHIL

special VoteBox-HF build includes extensive instrumentation for HCI work



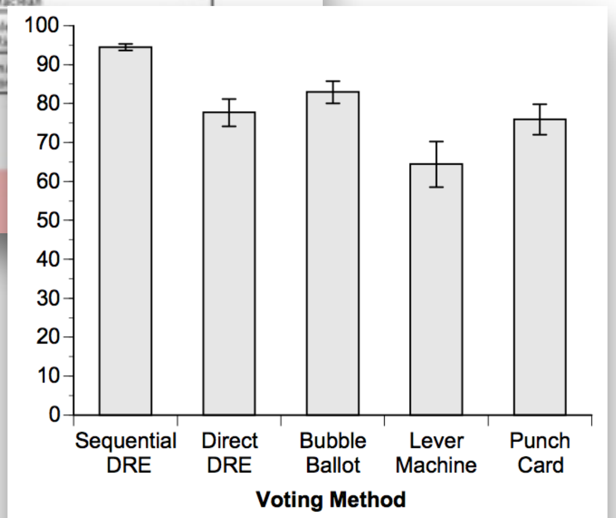
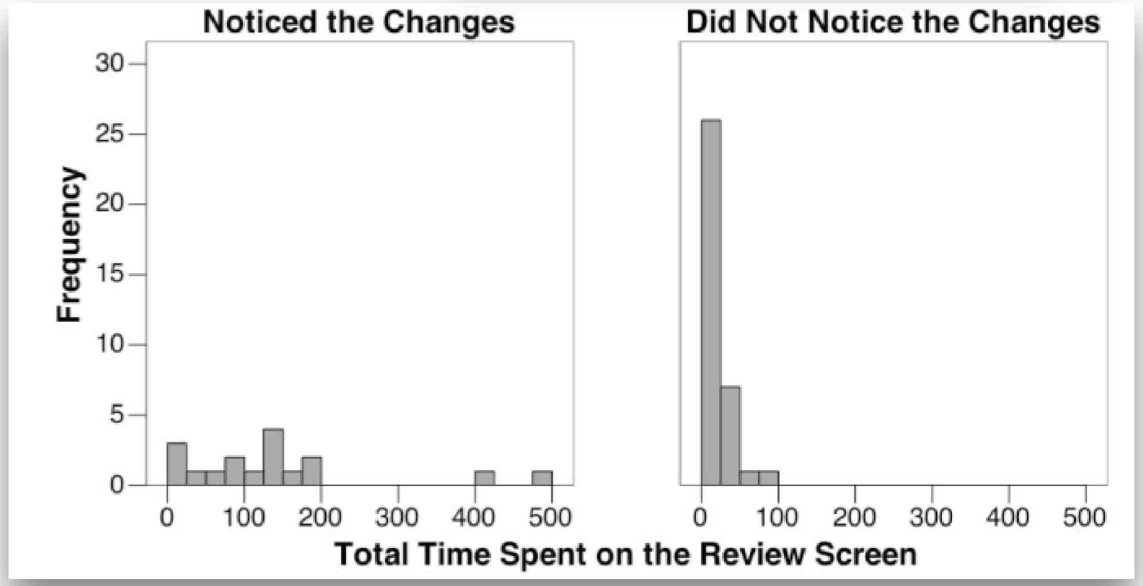


Figure 5. Mean SUS ratings by voting method.



HCI research

VoteBox is a platform for human factors research & experimentation

VoteBox's ballot designed jointly with Rice CHIL

special VoteBox-HF build includes extensive instrumentation for HCI work

Questions answered include:

"Do users prefer DREs?"

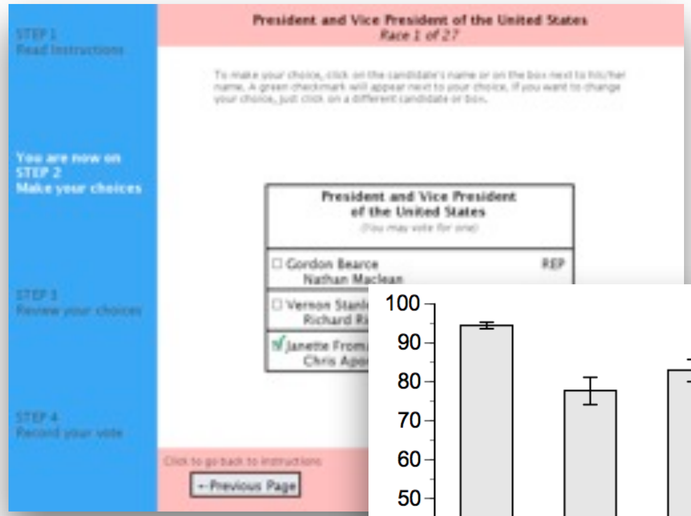
"Do DREs improve performance?"

"Do voters notice if DREs malfunction?"

Software engineering implications

Instrumentation is "evil" code from a security standpoint

Compile-time processing to exclude all HCI code from normal VoteBox builds



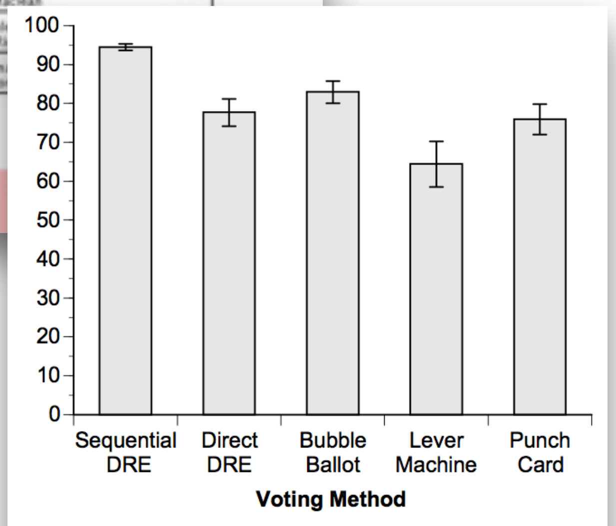
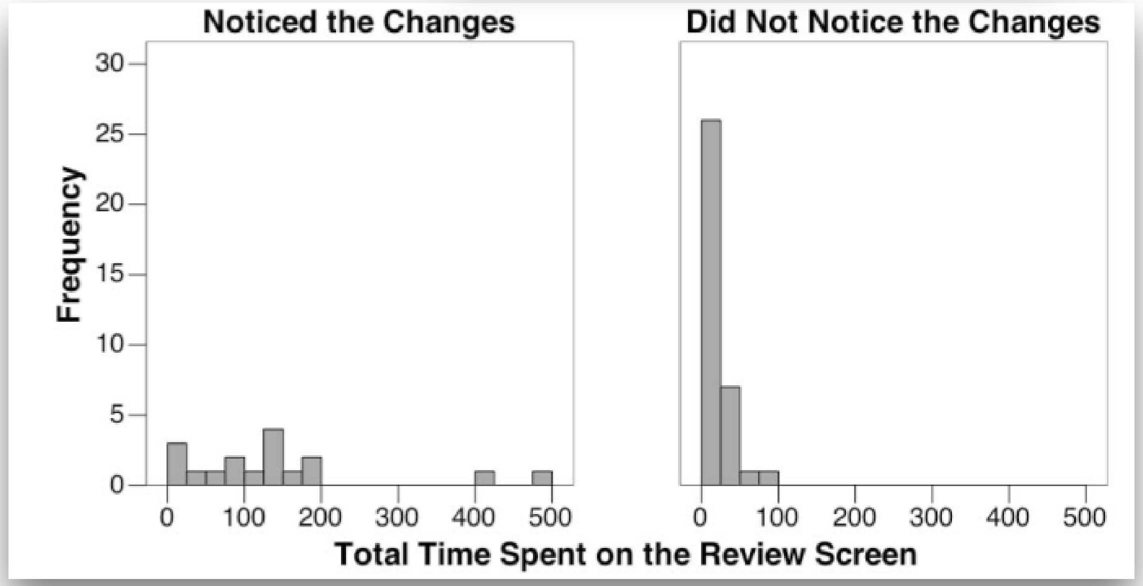


Figure 5. Mean SUS ratings by voting method.



Extensions.

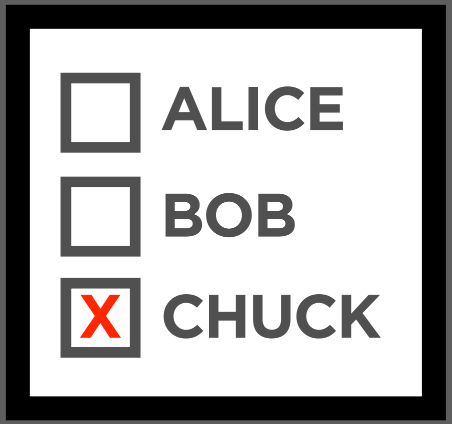
skip

internet voting
from home is
a bad idea

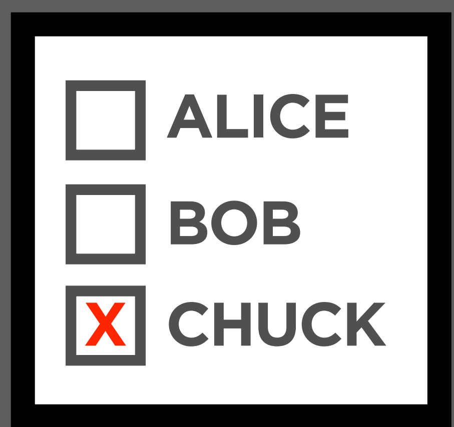
remote voting
can be
a good idea

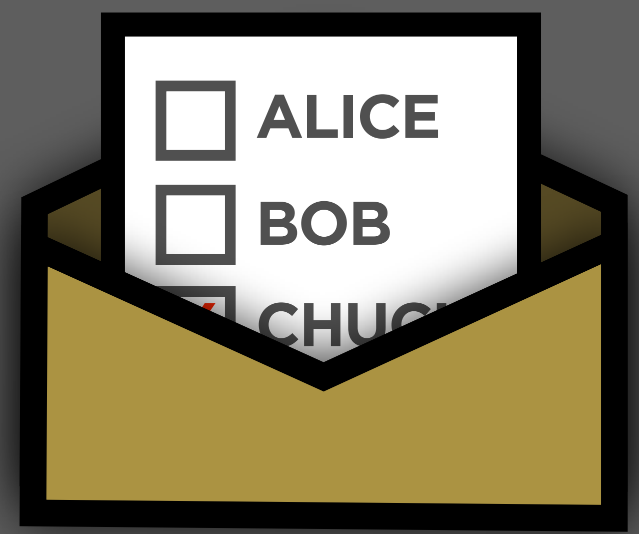
“postal voting”

“postal voting”

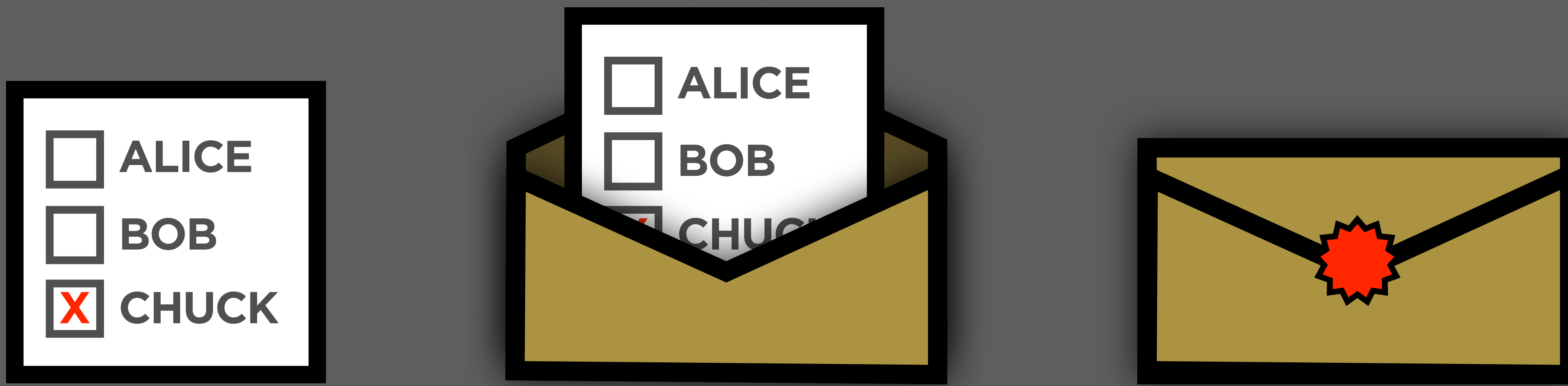


“postal voting”

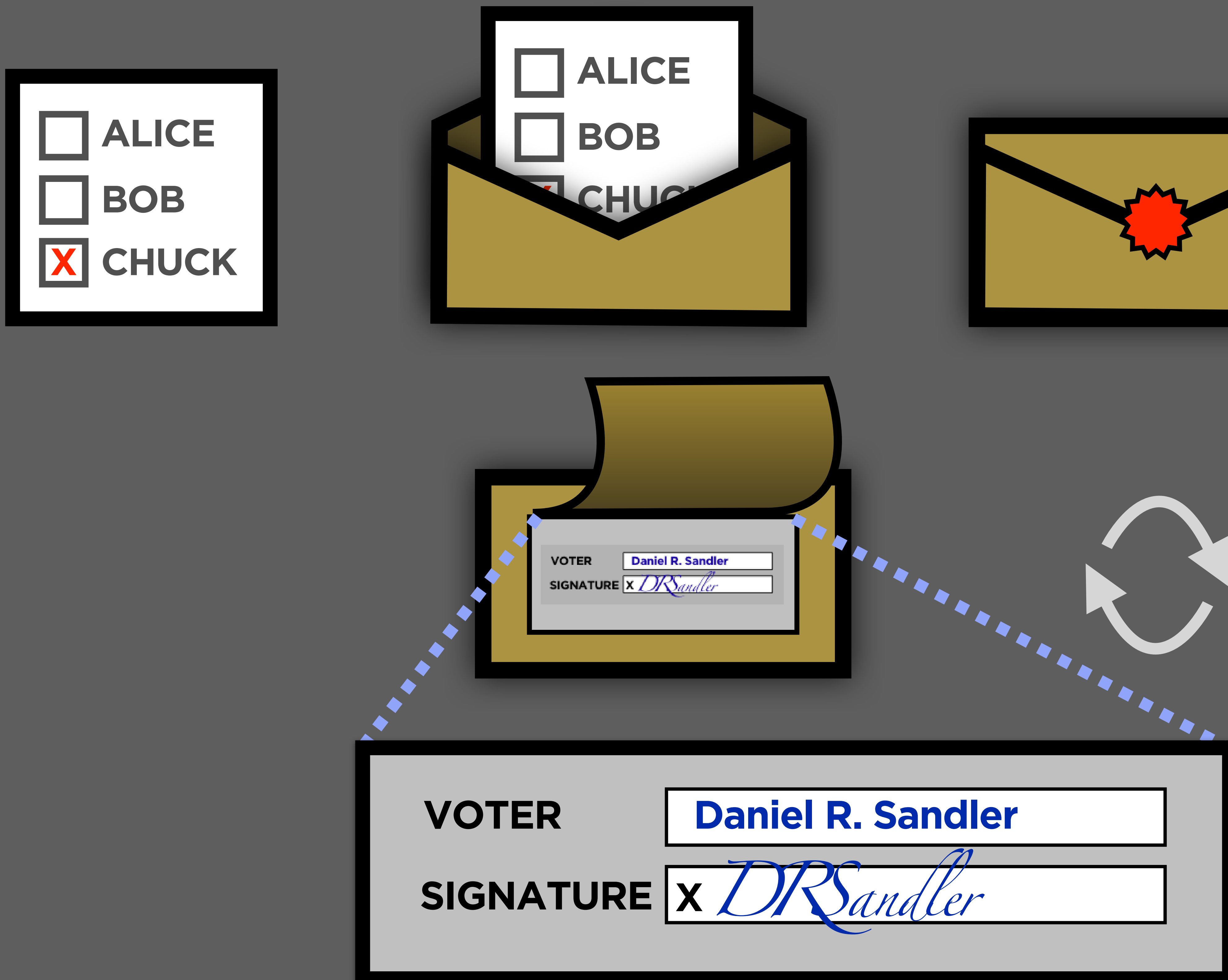




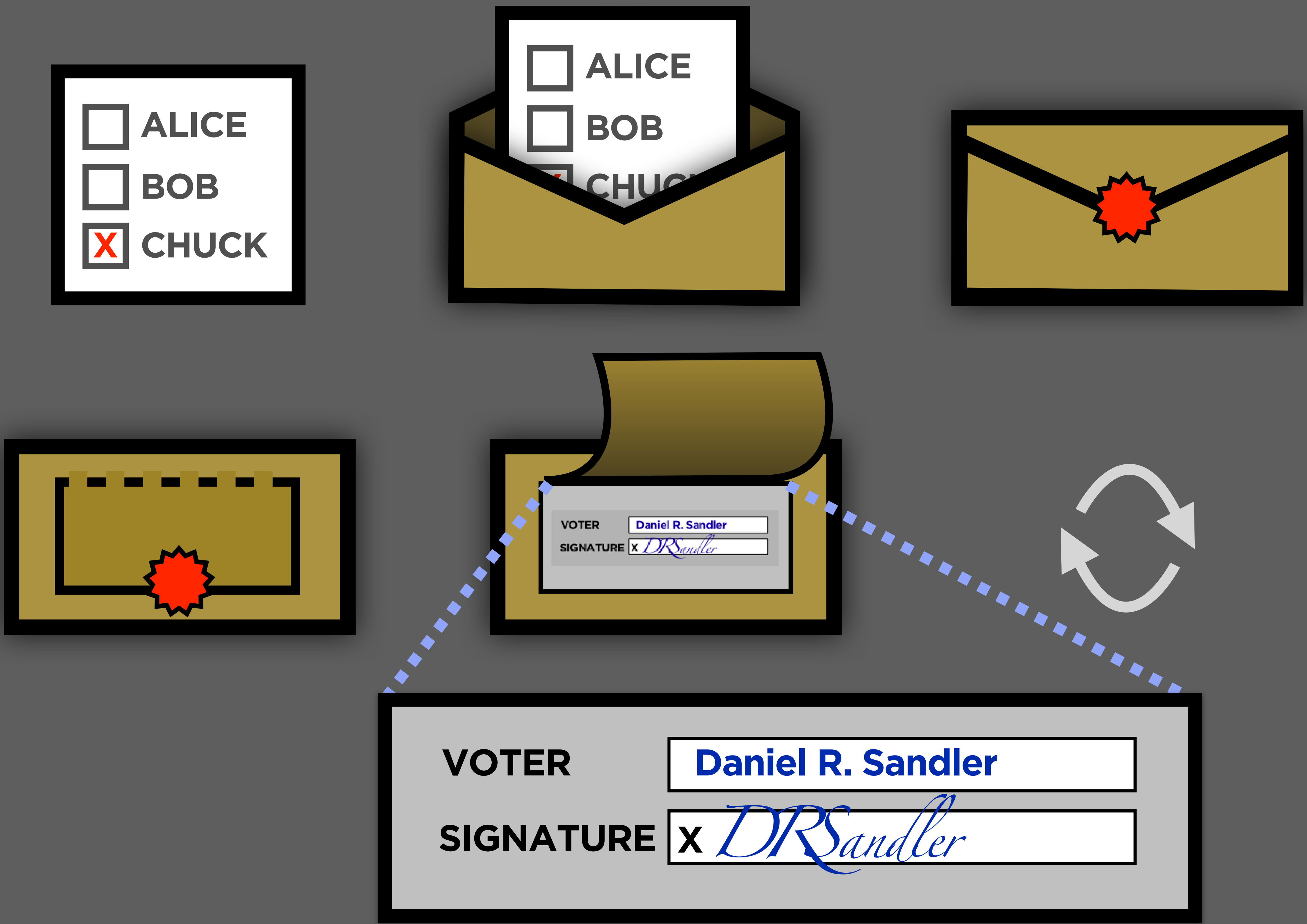
“postal voting”



“postal voting”



"postal voting"



we can do this with VoteBox

Conventional:	postal system
Replace with:	Auditorium network

Conventional:	sealed envelopes
Replace with:	encryption

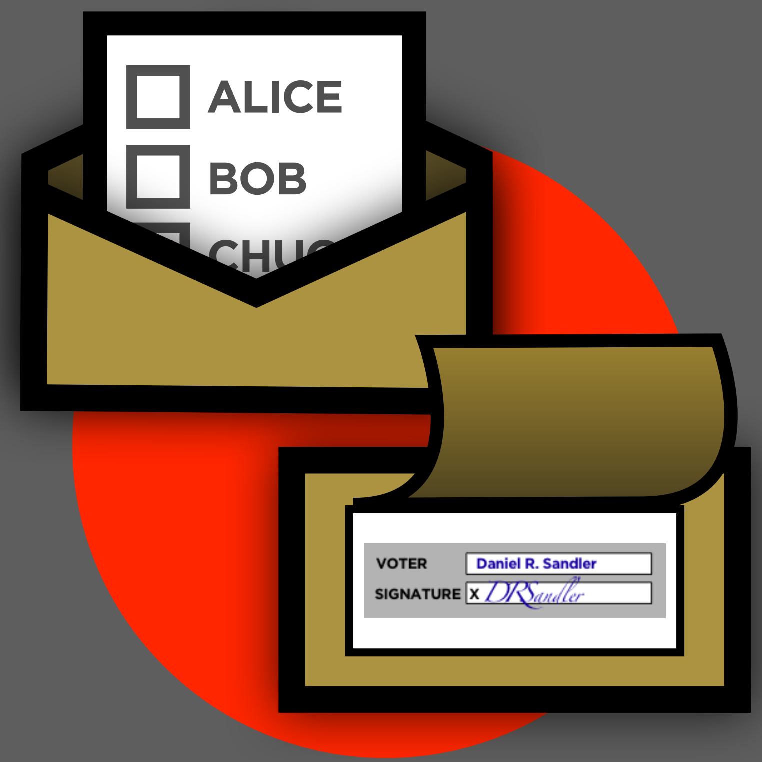

ID
S


ID
S
?
?

=

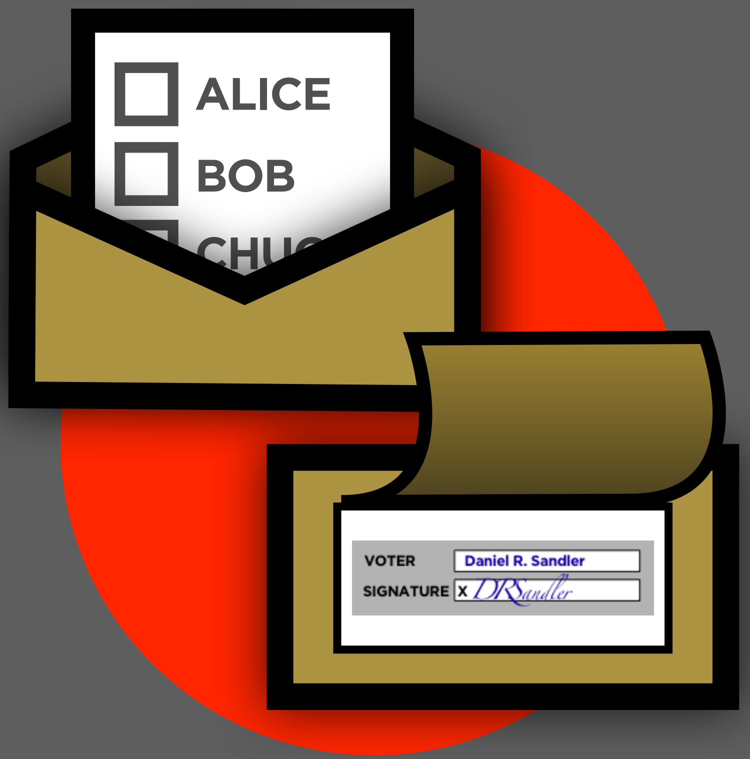

ID
S
?
?

=


ALICE
BOB
VOTER
Daniel R. Sandler
SIGNATURE
X


ID
S
? ?

>


ALICE
BOB
VOTER
Daniel R. Sandler
SIGNATURE
X

Benefits of the networked remote polling place

Fast

Ballot types from home precinct

Cast ballots back to home precinct

Robust

Post and networks both lossy

…but networks can retransmit

More secure

Choices cannot be observed while in transit

Crypto protects vote secrecy (even from officials)

3. Conclusion

why?

lots of research on
individual pieces
of the e-voting problem

VoteBox integrates these techniques in a **single system.**

D. R. Sandler, K. Derr, D. S. Wallach. **VoteBox: A tamper-evident, verifiable electronic voting system.** In USENIX Security 2008.

VoteBox integrates these techniques in a **single system.**

Auditorium (Sandler et al.)
robustness, tamper-evidence

D. R. Sandler, K. Derr, D. S. Wallach. **VoteBox: A tamper-evident, verifiable electronic voting system.** In USENIX Security 2008.

VoteBox integrates these techniques in a **single system.**

Auditorium (Sandler et al.)
robustness, tamper-evidence

Ballot challenge (new adaptation of Benaloh)
verifiability

D. R. Sandler, K. Derr, D. S. Wallach. **VoteBox: A tamper-evident, verifiable electronic voting system.** In USENIX Security 2008.

VoteBox integrates these techniques in a **single system.**

Auditorium (Sandler et al.)
robustness, tamper-evidence

Ballot challenge (new adaptation of Benaloh)
verifiability

Other ingredients
PRUI; HCI instrumentation

D. R. Sandler, K. Derr, D. S. Wallach. **VoteBox: A tamper-evident, verifiable electronic voting system.** In USENIX Security 2008.

VoteBox integrates these techniques in a **single system.**

Auditorium (Sandler et al.)
robustness, tamper-evidence

Ballot challenge (new adaptation of Benaloh)
verifiability

Other ingredients
PRUI; HCI instrumentation

Techniques suitable for integration with today's systems

D. R. Sandler, K. Derr, D. S. Wallach. **VoteBox: A tamper-evident, verifiable electronic voting system.** In USENIX Security 2008.

THREAT LEVEL

PRIVACY, SECURITY, POLITICS AND CRIME ONLINE

« High Court Asks Obama to Weigh In on Copyright Case; Conflict of Interest Brews | Main | Video: Sneaky New ATM Skimmer Found in Pennsylvania »

Voting Machine Audit Logs Raise More Questions about Lost Votes in CA Election

By Kim Zetter January 13, 2009 | 12:00:00 PM Categories: E-Voting, Election '08

Computer audit logs showing what occurred on a vote tabulation system that lost ballots in the November election are raising more questions not only about how the votes were lost, but also about the general reliability of voting system audit logs to record what occurs during an election and to ensure the integrity of results.

The logs, which Threat Level obtained through a public records request from Humboldt County, California, are produced by the Global Election Management System, the tabulation software, also known as GEMS, that counts the votes cast on all voting machines -- touch-screen and optical-scan machines -- made by Premier Election Solutions (formerly called Diebold Election Systems).

The logs are at the core of an investigation that the California secretary of state's office is conducting to determine why the GEMS tabulation system deleted 197 ballots from the tallies of one precinct in Humboldt County during the November 4 general election. But instead of providing transparency into what occurred on the system, the GEMS logs have so far only baffled state investigators. Deputy Secretary of State Lowell Finley has referred to the logs as "'Greek' to anyone other than a programmer."

http://blog.wired.com/27bstroke6/2009/01/diebold-audit-l.html

Voting Macl
Votes in CA

By Kim Zetter Janua

Computer audit logs sh
system that lost ballots i
questions not only abou
general reliability of vot
during an election and t

The logs, which Threat l
request from Humboldt
Global Election Manage
known as GEMS, that c
touch-screen and optica
Solutions (formerly call

The logs are at the core
secretary of state's offic
tabulation system delet
in Humboldt County du
instead of providing tra
the GEMS logs have so
Secretary of State Lowe
anyone other than a pro

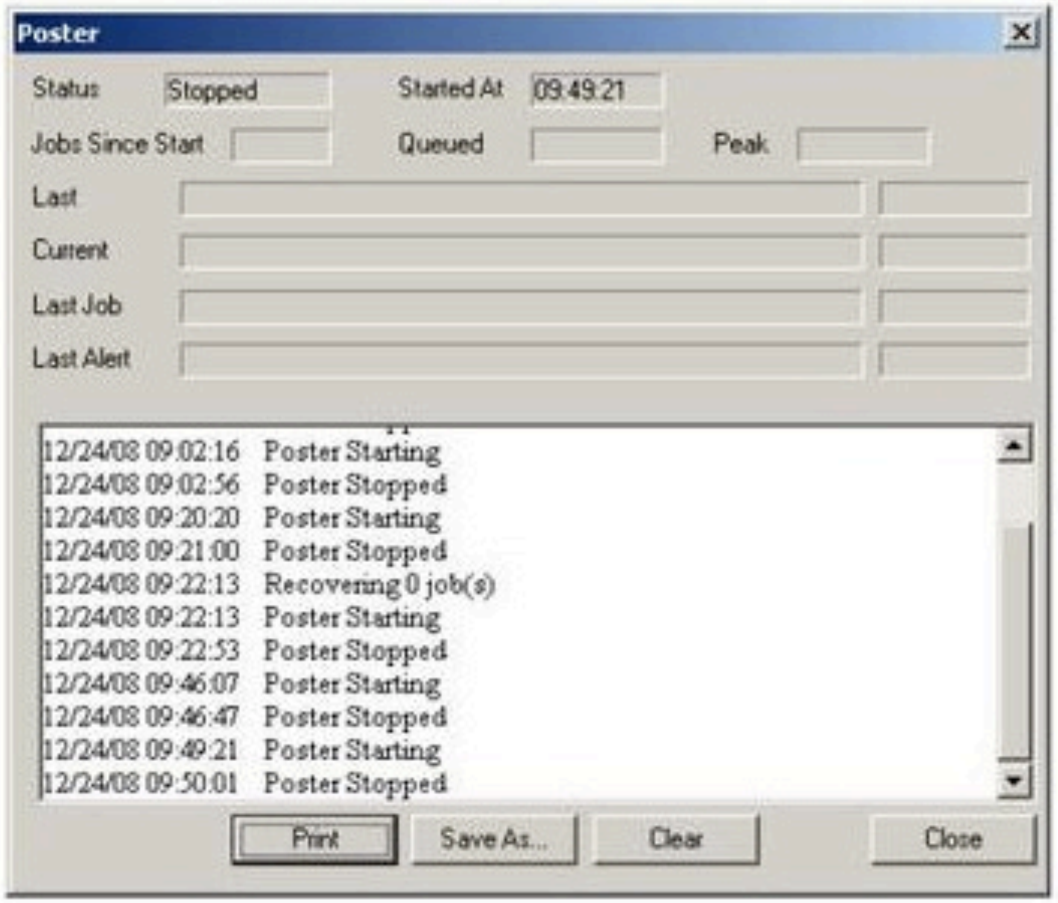
http://blog.w

Report: Diebold Voting System Has 'Delete' Button for Erasing Audit Logs

By Kim Zetter March 03, 2009 | 7:30:17 PM Categories: E-Voting

After three months of investigation, California's secretary of state has released a report examining why a voting system made by Premier Election Solutions (formerly known as Diebold) lost about 200 ballots in Humboldt County during November's presidential election.

But the most startling information in the state's 13-page report (.pdf) is not why the system lost votes, which Wired.com previously covered in detail, but that some versions of Diebold's vote tabulation system, known as the Global Election Management System (Gems), include a button that allows someone to delete audit logs from the system.



Auditing logs are required under the federal voting-system guidelines, which are used to test and qualify voting systems for use in elections. The logs record changes and other events that occur on voting systems to ensure the integrity of elections and help determine what occurred in a system when something goes wrong.

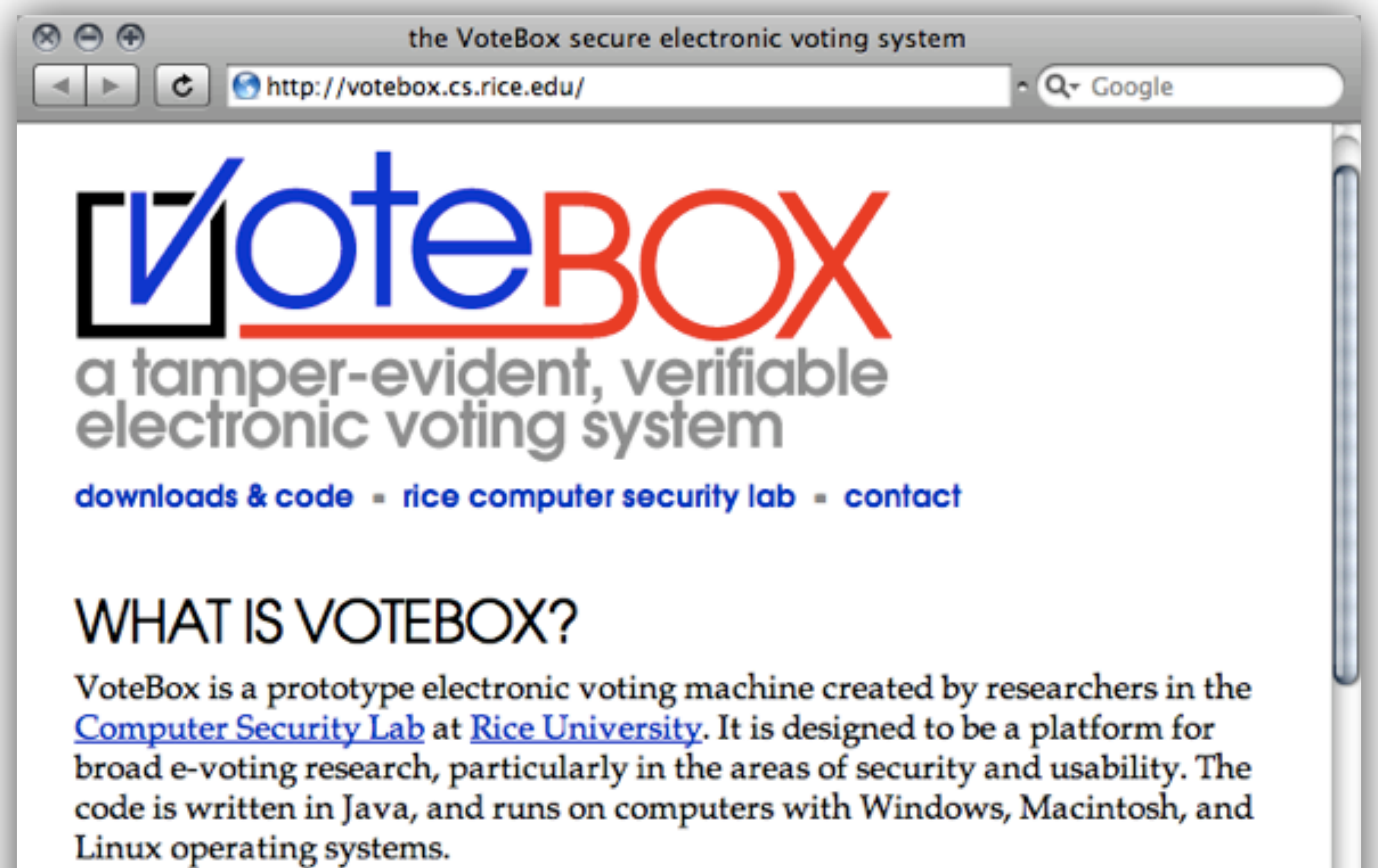
http://blog.wired.com/27bstroke6/2009/03/ca-report-finds.html

platform

VoteBox is open-source

votebox.cs.rice.edu & code.google.com/p/votebox

suitable for further research, HCI experiments, class projects, security analysis



thanks

co-authors

Scott Crosby, Kyle Derr, Daniel Sandler,
Ted Torous

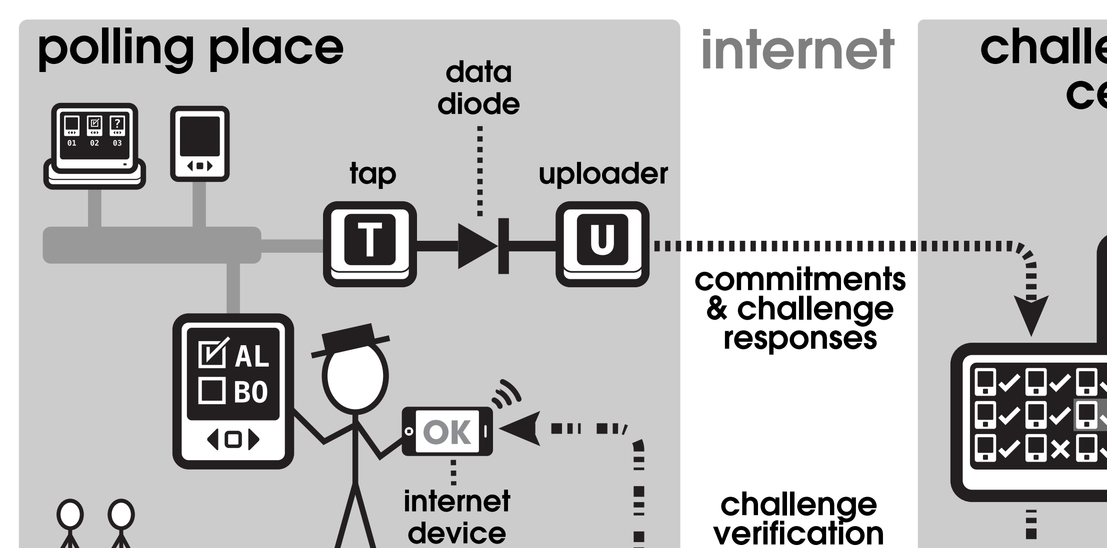
contributors to VoteBox

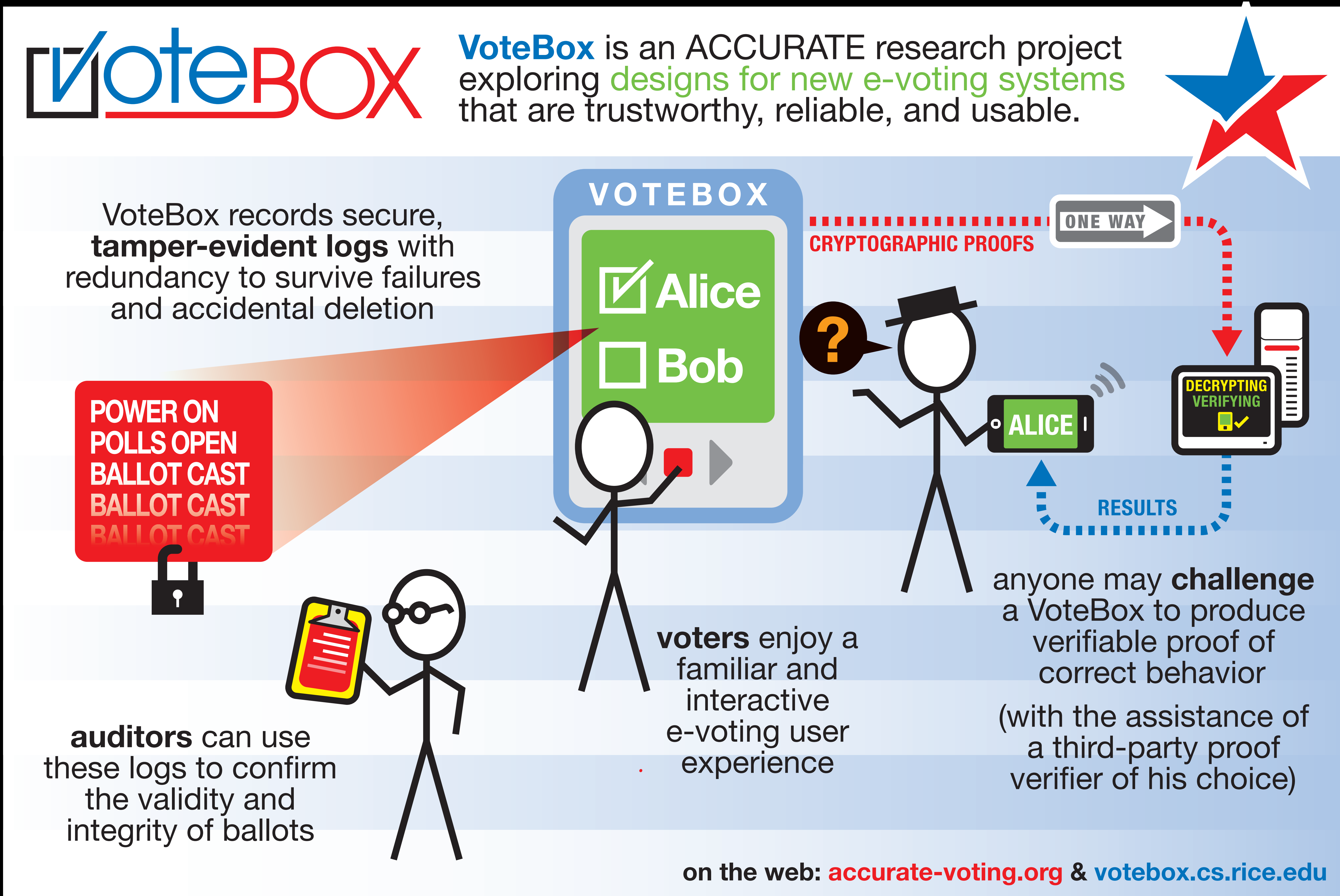
Emily Fortuna, George Mastrogiannis,
Kevin Montrose, Corey Shaw

CHIL

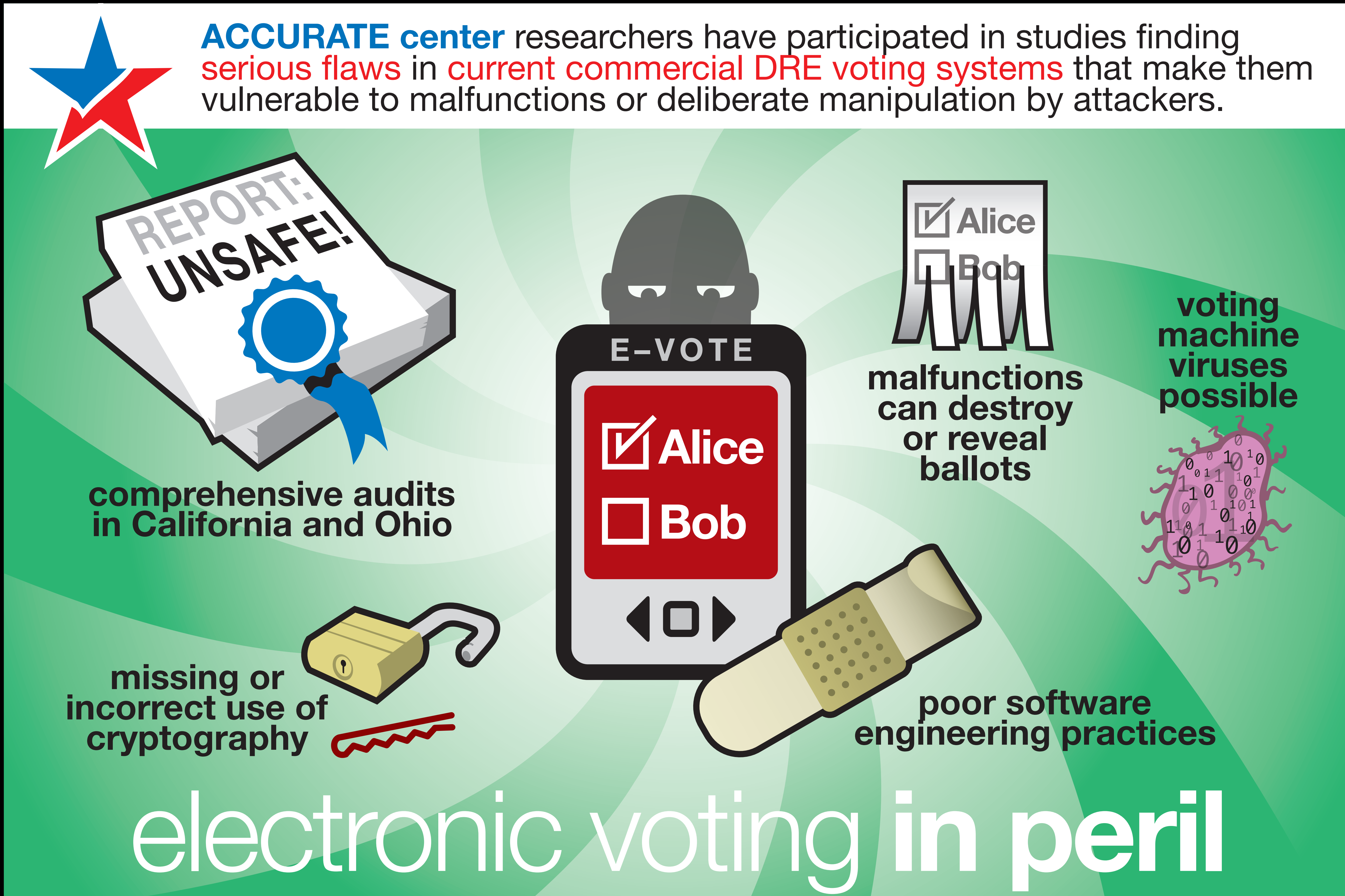
Mike Byrne, Sarah Everett, Kristen Greene

NSF/ACCURATE





NSF “highlights” graphic, 2009



NSF “highlights” graphic, 2009

(assorted backup slides)

Beyond.

Beyond VoteBox

Other systems need assurance, auditability, transparency

Future directions

- **email** *(auditability, document retention)*
- **web 2.0 publishing** *(reliability, openness)*
- **collaborative tools** *(event ordering, change tracking)*
- **gaming** *(ordering, cheat resistance & audit)*

email

entangled mailboxes

apply the tamper-evidence and timeline properties of auditorium to email records that must be highly auditable and recoverable

applications

Sarbanes-Oxley compliance

patents/notarization

Presidential records

status: *planning*

micropublishing

rapid short messaging

e.g. Twitter, Facebook

opt-in/social subscription

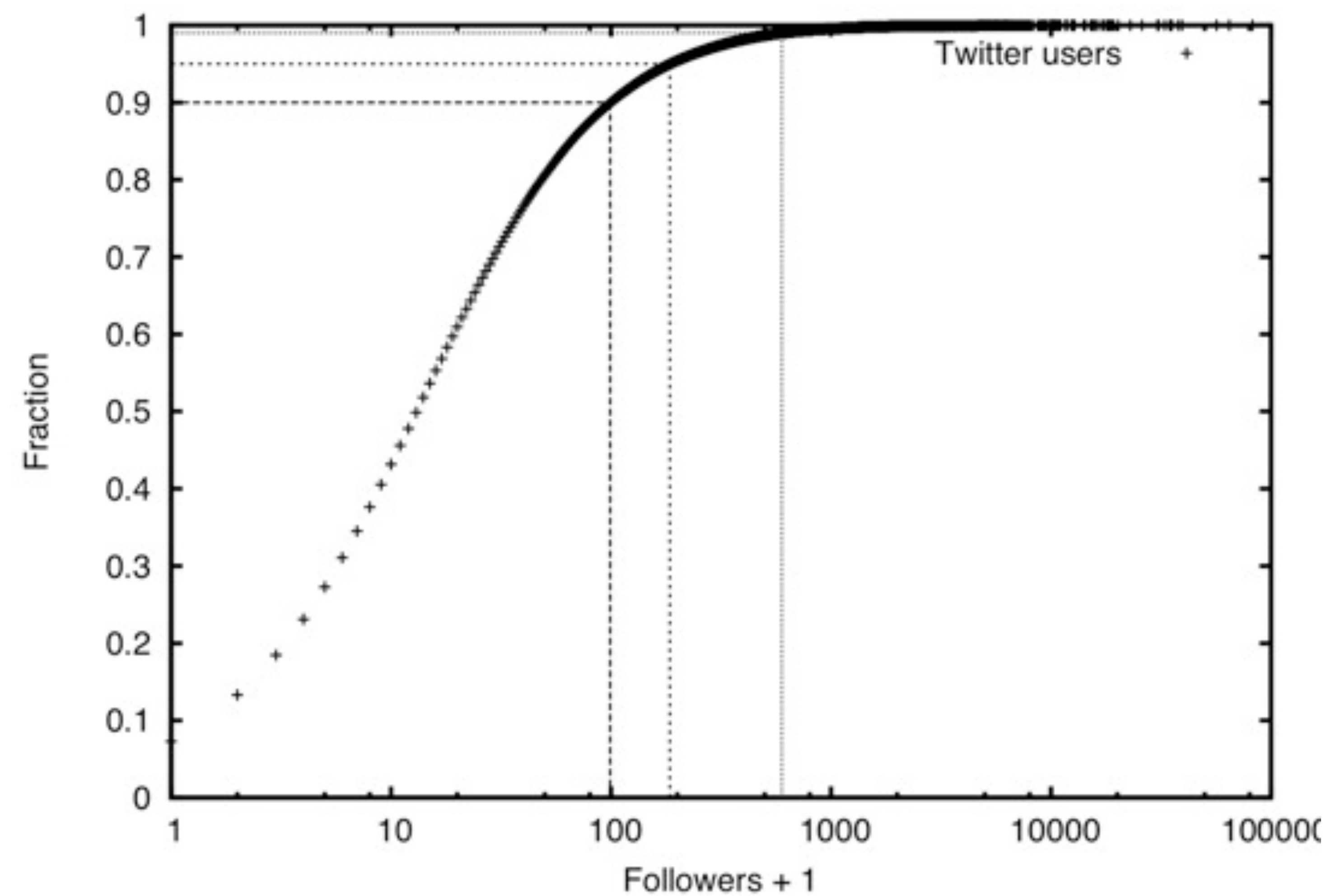
current systems are centralized, isolated, and limited

research opportunity

distributed, *secure* micropublishing

Auditorium-style timeline entanglement

scaling to millions of users



(data from Twitter, collected 2008)

continued…

micropublishing (2)

FETHR

micropublishing API

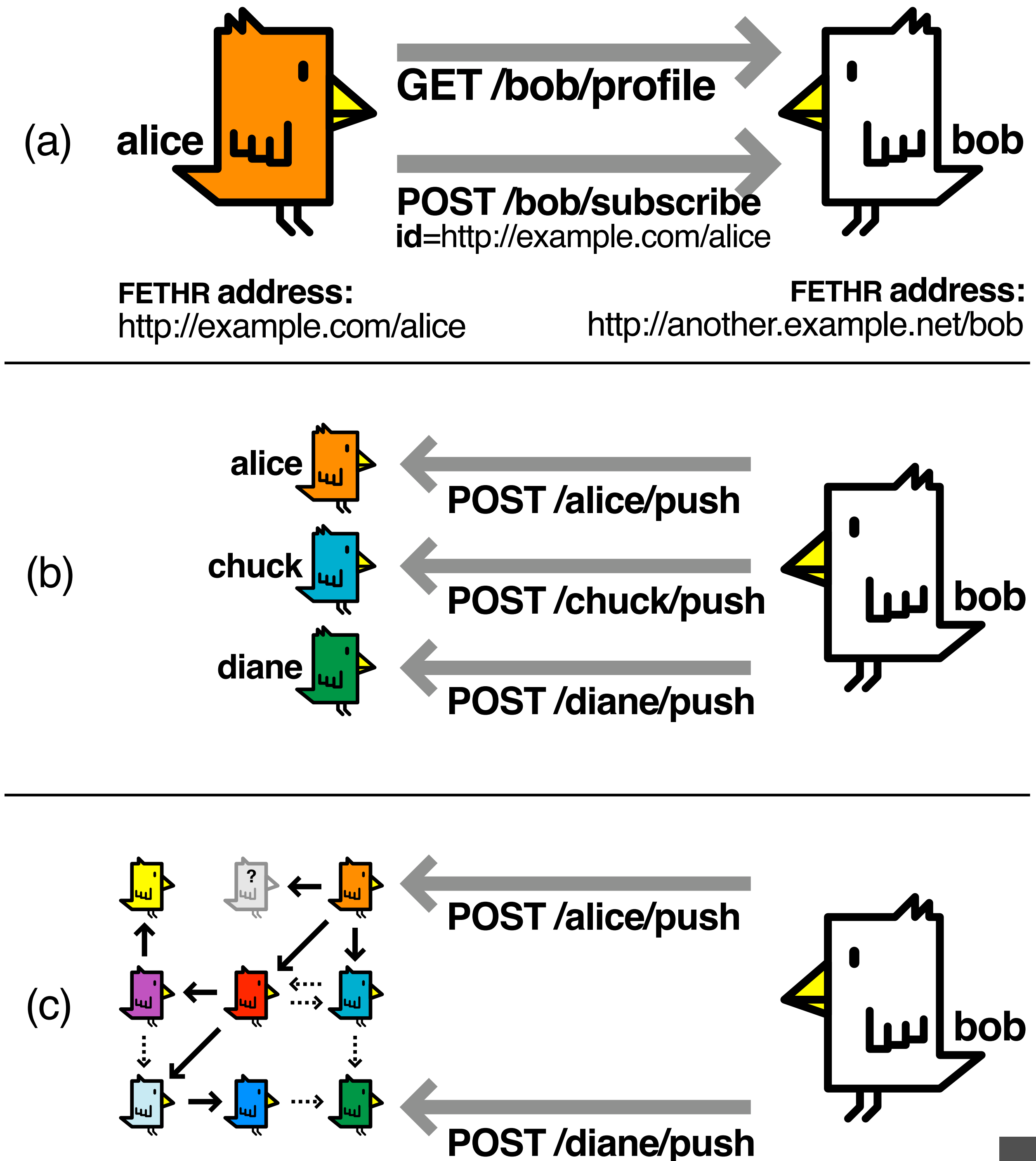
updates pushed to subscribers via HTTP POST

entanglement between publishers

gossip to assist in message distribution

prototype implementation: Birdfeeder (brdfdr.com)

status: *in progress; submitted (IPTPS)*

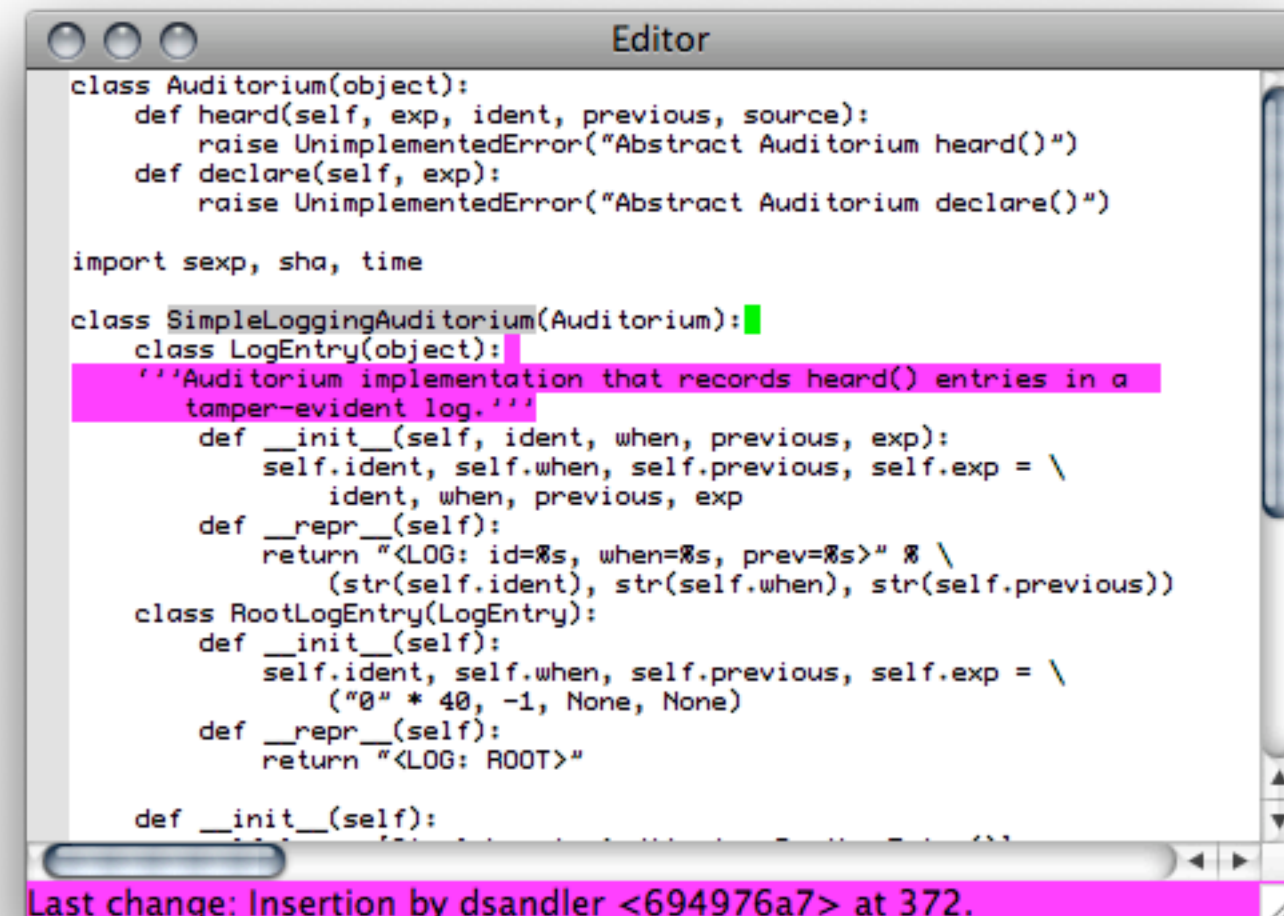


collaborative tools

timeline entanglement to represent sequence of edits or actions

ordering of events corresponds neatly to causality in groupware

status: *prototyped*



networked games

Auditorium-style communication for participants

gossip for decentralization, reliability

hash chains forward & backward (move commitment, history authentication)

secure logs for post-facto audit of suspected cheating

Fancy Cryptography

Violation of encryption semantics?

If I know M_1 and M_2 and $E(M_1) \oplus E(M_2) = E(M_1 M_2)$
then I can find other messages where
I know their encryption!

Solution: Padding

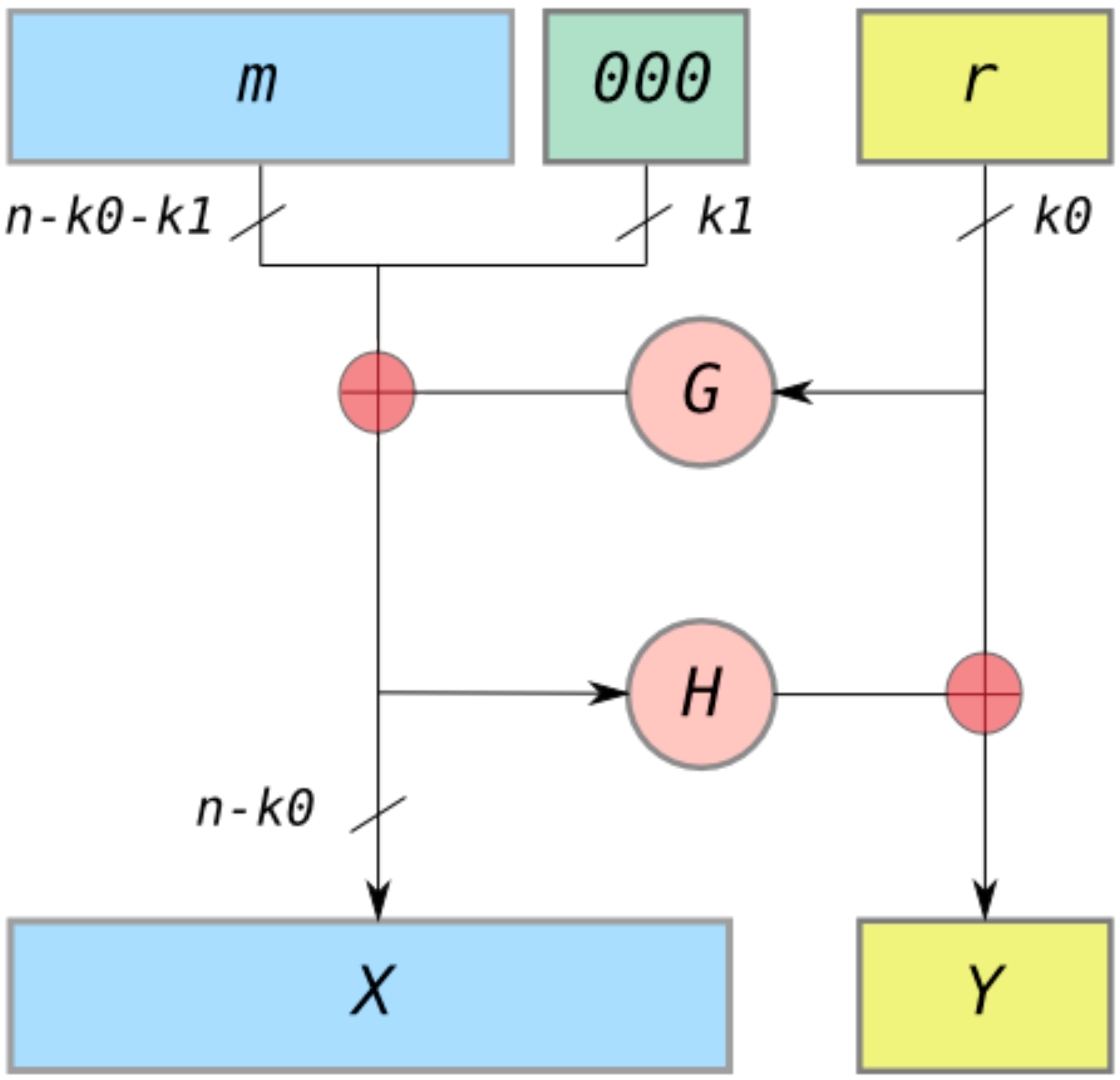
Optimal Asymmetric Encryption Padding (OAEP) - *Belare and Rogaway (1995)*

m - message (plaintext)

r - random number

G, H - cryptographic hash functions

X, Y - the message that gets encrypted



Cool trick: reencryption

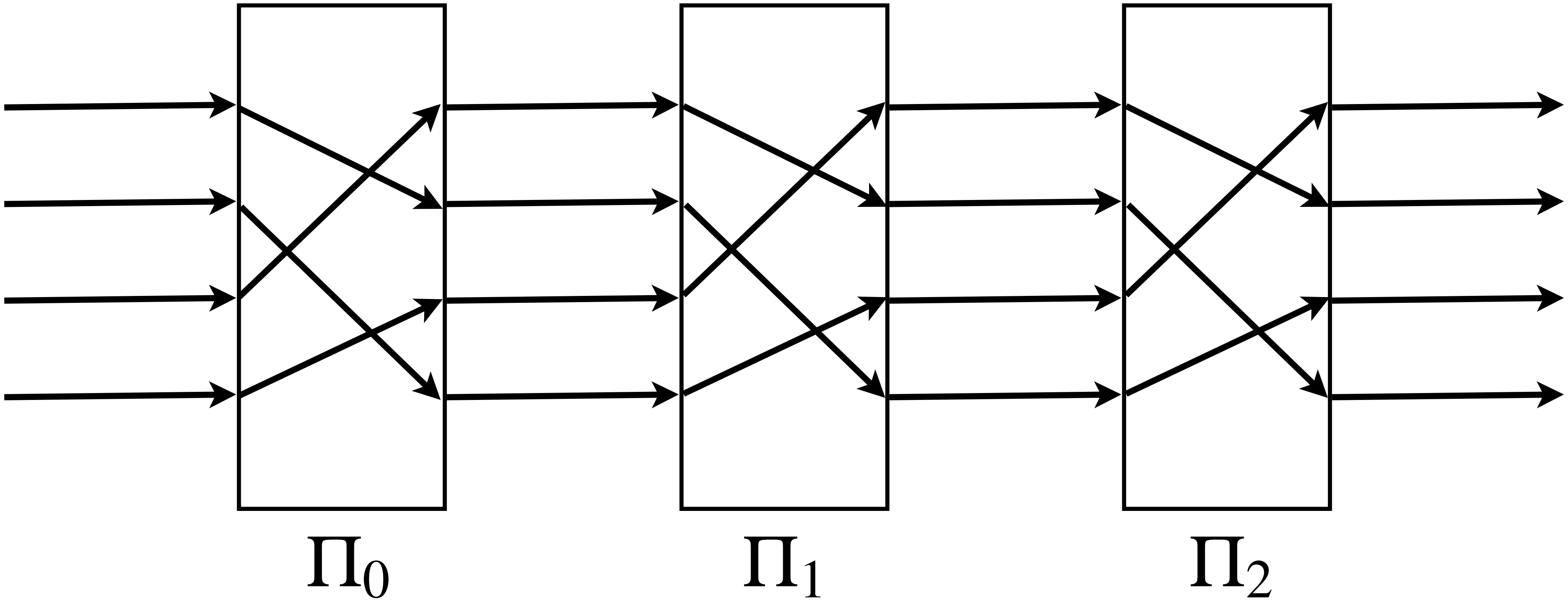
$$E(M) \oplus E(0) = E(M)^*$$

Anybody can "reencrypt" a message.

(New random number introduced from *E*(0).)

Reencryption mixnets

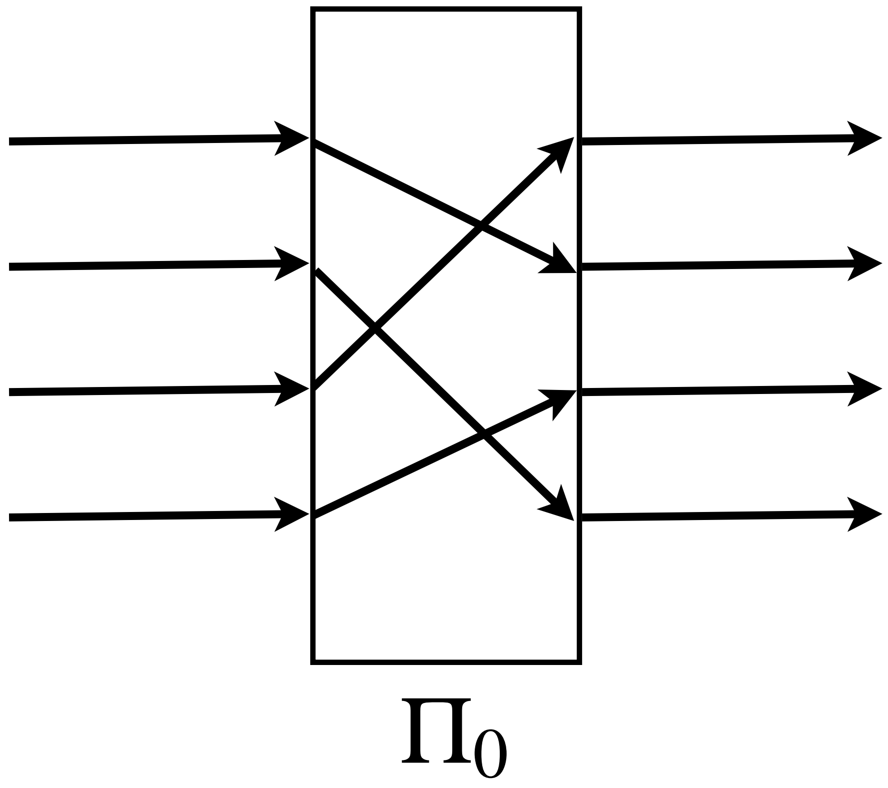
Permutations Π_i, where output is reencrypted.



Each mix permutes/reencrypts.
Must prove output corresponds to input.

Non-solution: reveal the mix

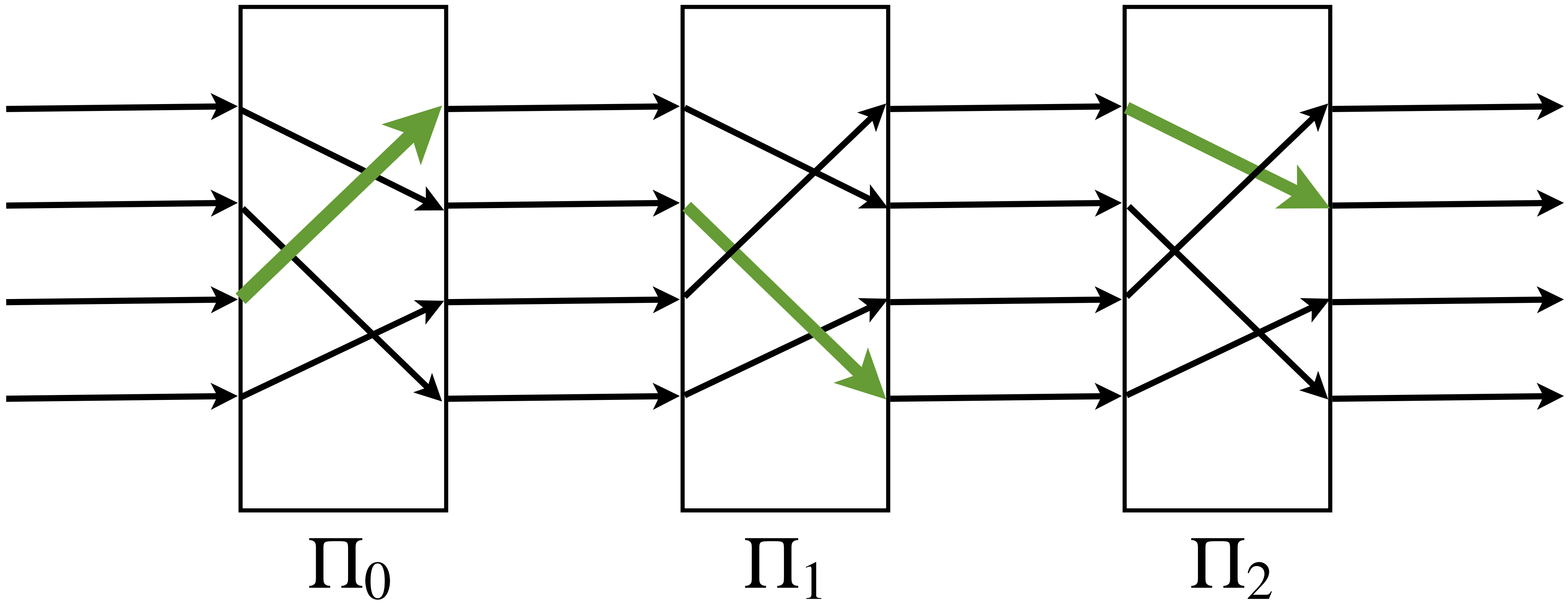
Publish the random numbers and the permutation.



Eliminates benefit of randomization.

Randomized partial checking

Effective across larger mixes.
(Jakobsson, Jules, Rivest '02)



Say we're mixing 1 million ballots, each mix reveals 1%. After five mixes, 99.99% chance that all ballots reencrypted at least once.

Zero-knowledge proofs (ZKP)

want to prove you know something

while revealing nothing

generalized format

prover: commit to something (e.g., reencryption mix output)

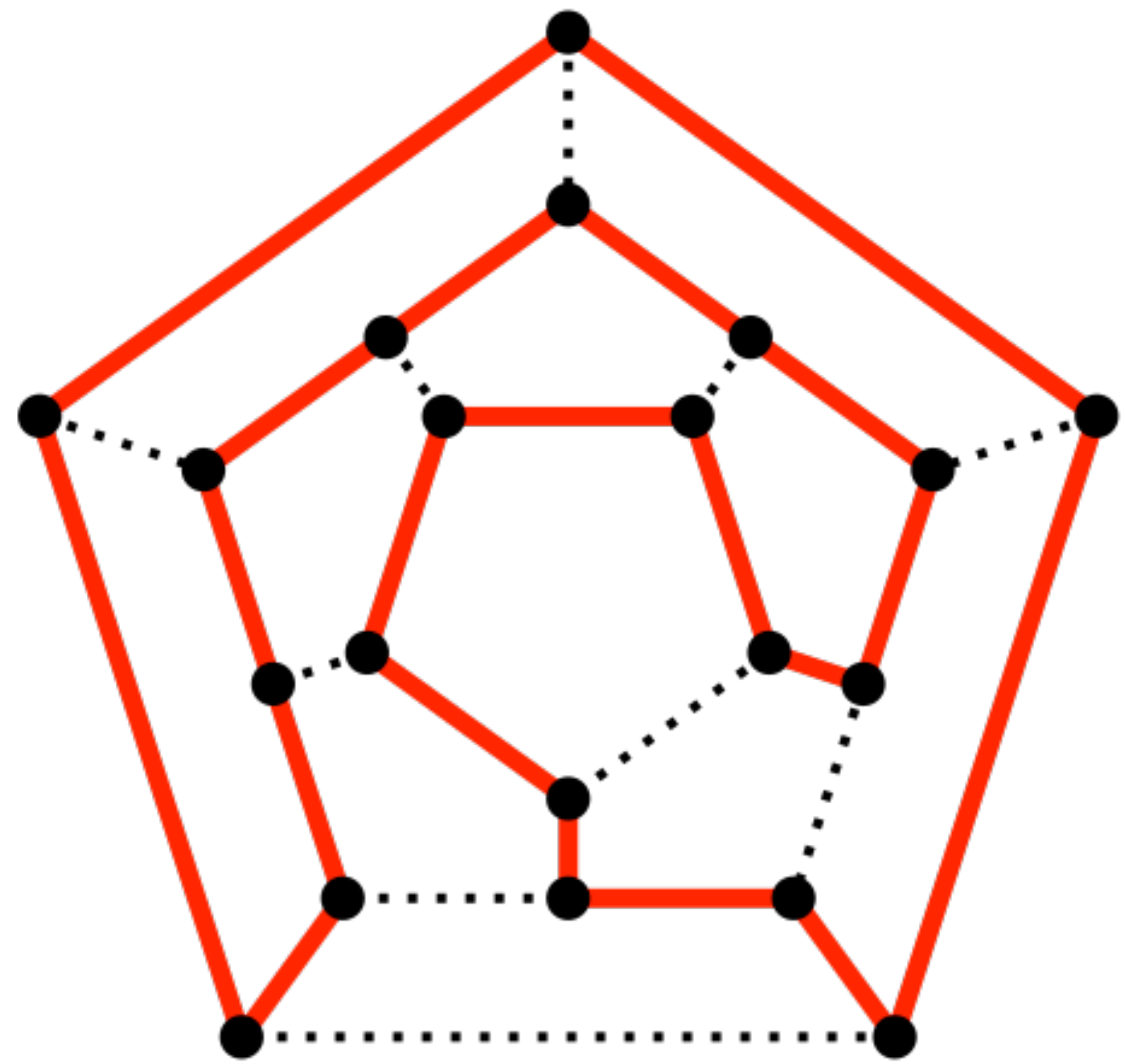
verifier: *challenge* the prover

prover: respond to the challenge

Example: Hamiltonian paths

Prover: "I know a HP over graph *G*." Compute graph isomorphism *H*. Publish *G, H.*

Verifier: Coin toss. Heads: tell me HP over *H*. Tails: tell me isomorphism *G* to *H*.



(Repeat *N* times.)

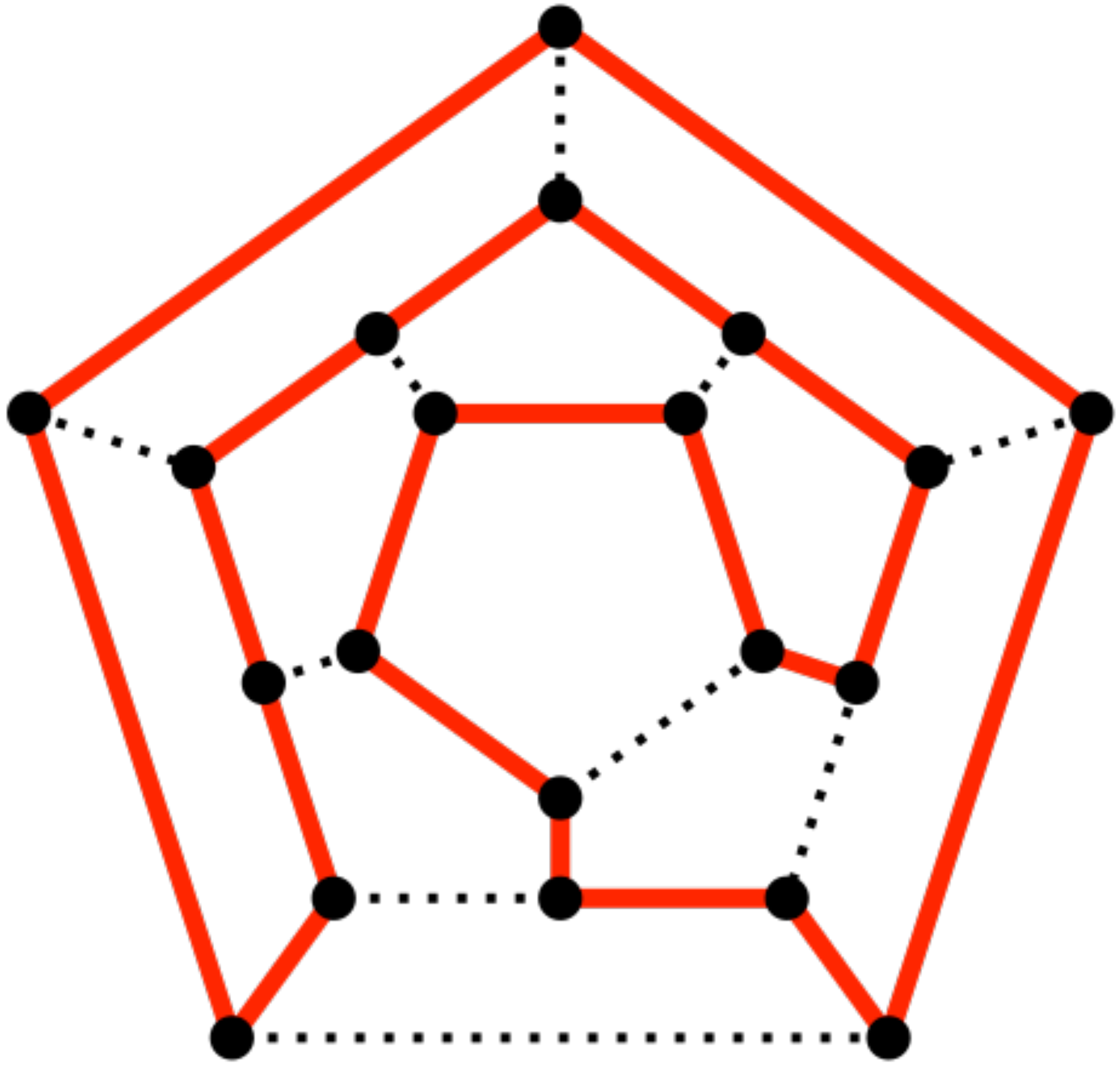
If prover doesn't know HP, verifier catches with high probability.

Non-interactive ZK proofs

Prover: Precompute *N* isomorphisms (H_1 to H_N) and hash them. Hash function yields coin tosses for virtual challenger. Then output the results.

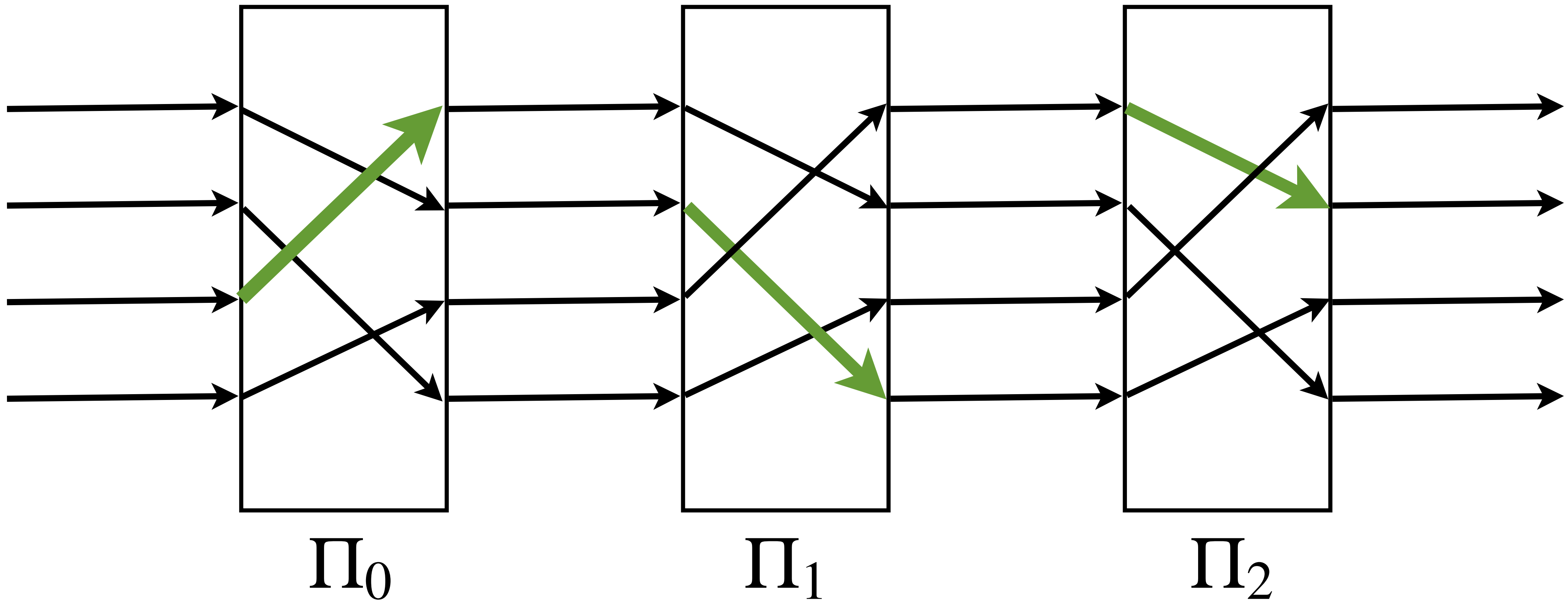
(Assumes good hash functions.)

This is an example of the *Fiat-Shamir heuristic* (1986).



NIZK variant for mixes

Hash the output of the permutation/reencryption. Use those bits to select which edges get revealed.



Say we're mixing 1 million ballots, each mix reveals 1%. After five mixes, 99.99% chance that all ballots reencrypted at least once.

Evil machine: E(bignum)?

Must prove ciphertext corresponds to well-formed plaintext. (Example, prove counters are zero or one.)

We need another ZK tool: Chaum-Pedersen proofs.

Prover knows: $(g, g^x), (h, h^x)$

Wants to prove that these two tuples share x

Chaum-Pedersen proofs (1992)

Goal: demonstrate $(g, g^x), (h, h^x)$

P: choose random $w \in \mathbb{Z}_p^*$, compute $(A = g^w, B = h^w)$

Send (A,B) to V

V: pick a random number c (challenge), send to P

P: compute $R = w + xc$

send R to V

V: Compute

$$\begin{aligned} A(g^x)^c &= g^w g^{xc} \\ &= g^{w+xc} \\ &= g^R \end{aligned}$$

$$\begin{aligned} B(h^x)^c &= h^w h^{xc} \\ &= h^{w+xc} \\ &= h^R \end{aligned}$$

Fake C-P proofs?

Goal: demonstrate $(g, g^x), (h, h^x)$

P: choose random $w \in \mathbb{Z}_p^*$, compute $(A = g^w, B = h^w)$

Send (A,B) to V

V: pick a random number c (challenge), send to P

P: compute $R = w + xc$

send R to V

V: Compute

$$\begin{aligned} A(g^x)^c &= g^w g^{xc} \\ &= g^{w+xc} \\ &= g^R \end{aligned}$$

$$\begin{aligned} B(h^x)^c &= h^w h^{xc} \\ &= h^{w+xc} \\ &= h^R \end{aligned}$$

Fake C-P proofs?

Goal: demonstrate $(g, g^x), (h, h^x)$

P: choose random $w \in \mathbb{Z}_p^*$, compute $(A = \cancel{g^w}, B = \cancel{h^w})$

Send (A,B) to V

V: pick a random number c (challenge), send to P

P: compute $R = w + xc$

send R to V

V: Compute

$$\begin{aligned} A(g^x)^c &= g^w g^{xc} \\ &= g^{w+xc} \\ &= g^R \end{aligned}$$

$$\begin{aligned} B(h^x)^c &= h^w h^{xc} \\ &= h^{w+xc} \\ &= h^R \end{aligned}$$

Fake C-P proofs?

Goal: demonstrate $(g, g^x), (h, h^x)$

P: choose random $w \in \mathbb{Z}_n^*$, compute $(A = g^w, B = h^w)$

Send (A, B) to

P choses fake c, R: then $A = g^R (g^{xc})^{-1}$.

V: pick a random number c (challenge), send to P

P: compute $R \cancel{= w + xc}$

send R to V

$$A(g^x)^c$$

V: Compute

$$\begin{aligned} A(g^x)^c &= g^w g^{xc} \\ &= g^{w+xc} \\ &= g^R \end{aligned}$$

$$\begin{aligned} B(h^x)^c &= h^w h^{xc} \\ &= h^{w+xc} \\ &= h^R \end{aligned}$$

Fake C-P proofs?

Goal: demonstrate $(g, g^x), (h, h^x)$

P: choose random $w \in \mathbb{Z}_p^*$, compute $(A = g^w, B = h^w)$

Send (A,B) t

P choses fake c, R: then $A = g^R (g^{xc})^{-1}$.

V: pick a random number c (challenge), send to P

P: compute $R = w + xc$

send R to V

Observer can compute $A(g^x)^c$...

V: Compute

$$\begin{aligned} A(g^x)^c &= g^w g^{xc} \\ &= g^{w+xc} \\ &= g^R \end{aligned} \qquad \begin{aligned} B(h^x)^c &= h^w h^{xc} \\ &= h^{w+xc} \\ &= h^R \end{aligned}$$

Fake C-P proofs?

Goal: demonstrate $(g, g^x), (h, h^x)$

P: choose random $w \in \mathbb{Z}_p^*$, compute $(A = g^w, B = h^w)$

Send (A,B) to

P choses fake c, R: then $A = g^R (g^{xc})^{-1}$.

V: pick a random number c (challenge), send to P

P: compute $R = w + xc$

send R to V

Observer can compute $A(g^x)^c$...

V: Compute

$$\begin{aligned} A(g^x)^c &= g^w g^{xc} \\ &= g^{w+xc} \\ &= g^R \end{aligned}$$

$$\begin{aligned} B(h^x)^c &= h^w h^{xc} \\ &= h^{w+xc} \\ &= h^R \end{aligned}$$

ZK protocols only work when "live" (or use Fiat-Shamir heuristic for non-interactive)

C-P for vote testing

Can I prove a vote is zero or one? First, how about proving it's zero using C-P.

Want to verify $\langle g^r, g^{ar} g^v \rangle$ for a specific value of v?

Do C-P protocol where $(g, g^x), (h, h^x)$ becomes

$$(g, g^r), \left(g^a, \frac{g^{ar} g^v}{g^v} \right)$$

We could do this for any value of v

Challenge is to do $v = 0$ and $v = 1$ at the same time.

Cramer-Damgård-Schoenmakers (1996)

Can run two Chaum-Pedersen (or any two ZK proofs like this) simultaneously, one “real” and one “simulated”.

First, fake a proof (e.g., for $v = 1$) in advance.

Then, announce the first message for both protocols. Challenger sends c, prover announced a split c_0, c_1 where $c_0 + c_1 = c$, then executes both ZK protocols.

Verifier cannot tell which one was real vs. simulated, but knows that **one** of them was real.

Crypto summary

At the end of the day, **any** election observer can now:

- verify every single ballot for being "well-formed"
  (valid Elgamal tuple, encrypted zero-or-one, etc.)
- add together all the ballots (homomorphically)
- verify a proof of the tally (Chaum-Pedersen again)
  (only the election authority can generate this)

But we have no idea if the original ciphertext corresponded to the **intent of the voter** (versus evil machine flipping votes).