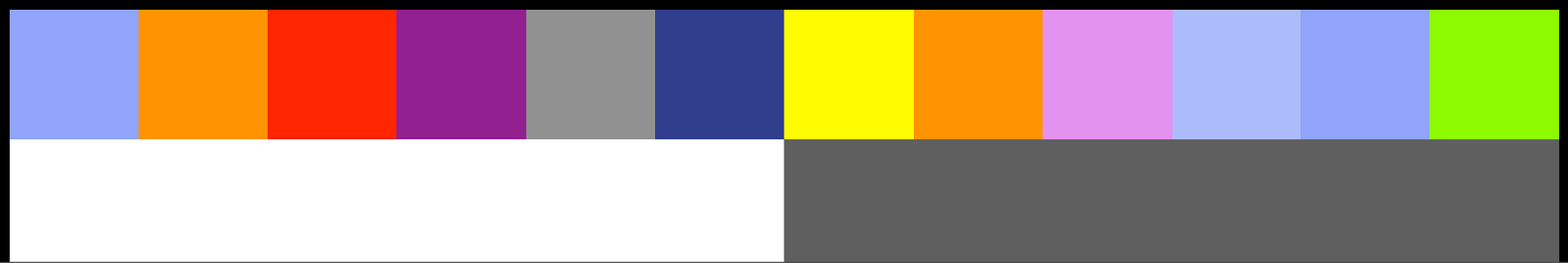

COMPUTER SECURITY LAB
RICE UNIVERSITY



Crypto and e-Voting: Homomorphisms, Zero-Knowledge Proofs

Dan S. Wallach
Rice University

Back to basics: Diffie-Hellman & Elgamal Crypto

Modular arithmetic 101

We're working in $\mathbb{Z}_p^*$, the integers in [1, *p*)

$$2+3 = 5 \pmod 7$$
$$2+4 = 6 \pmod 7$$
$$2+5 = 0 \pmod 7 \longleftarrow \textbf{Forbidden!}$$
$$2*3 = 6 \pmod 7$$
$$2*4 = 1 \pmod 7$$
$$6*6 = 1 \pmod 7$$

Note: $\mathbb{Z}_p^*$ is closed under multiplication but not addition.

Modular arithmetic 101

In $\mathbb{Z}_p^*$, we want to find *generators* such that

$$g^1, g^2, \ldots, g^{p-1}$$

cover all the elements in the group.

Example, for *p*=7:

g=2 is not a generator, but *g*=3 is.

Discrete logarithms

Back to the regular integers, say I give you a very big number $q = 5^{84375912432595 43}$

and ask you to take $\log_5 q$

Logarithms, over integers, are tractable. But what about in $\mathbb{Z}_p^*$**?**

No known efficient solution to DLog problem.

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)

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

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

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

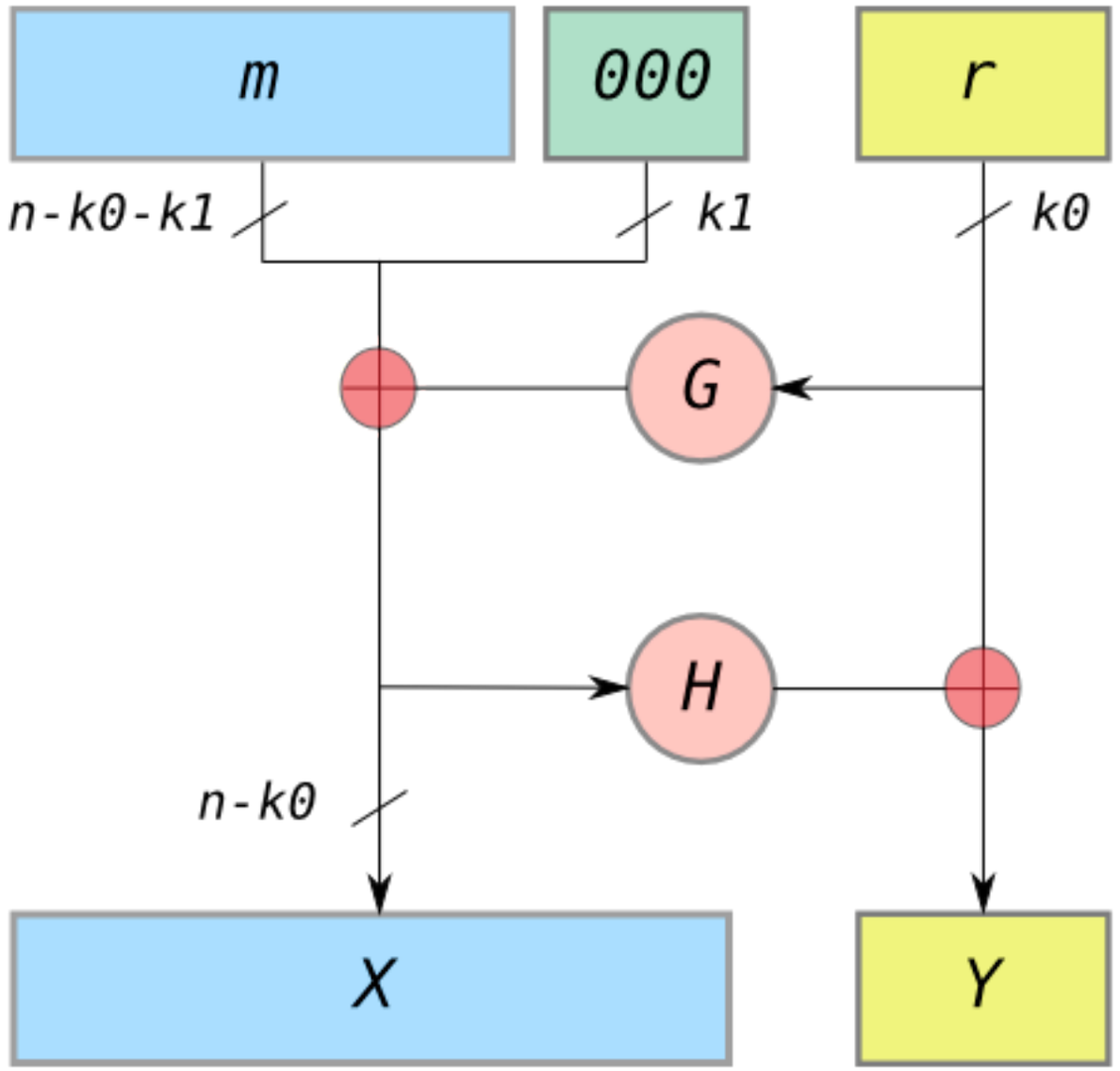
Optimal Asymmetric Encryption Padding (OAEP) - *Belare and Rogaway (1995)*

m - message (plaintext)

r - random number

G, H - cryptographic hash functions

X, Y - the message that gets encrypted



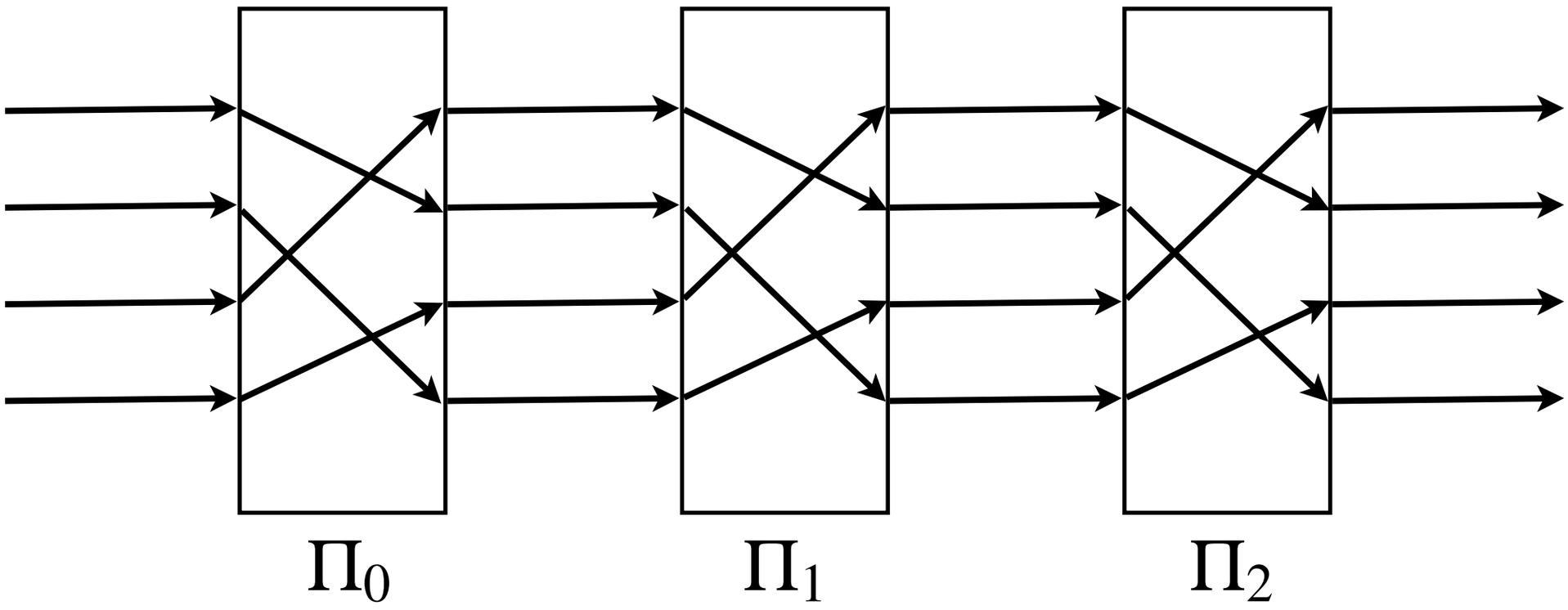
Cool trick: reencryption

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

Anybody can "reencrypt" a message.
(New random number introduced from $E(0)$.)

Reencryption mixnets

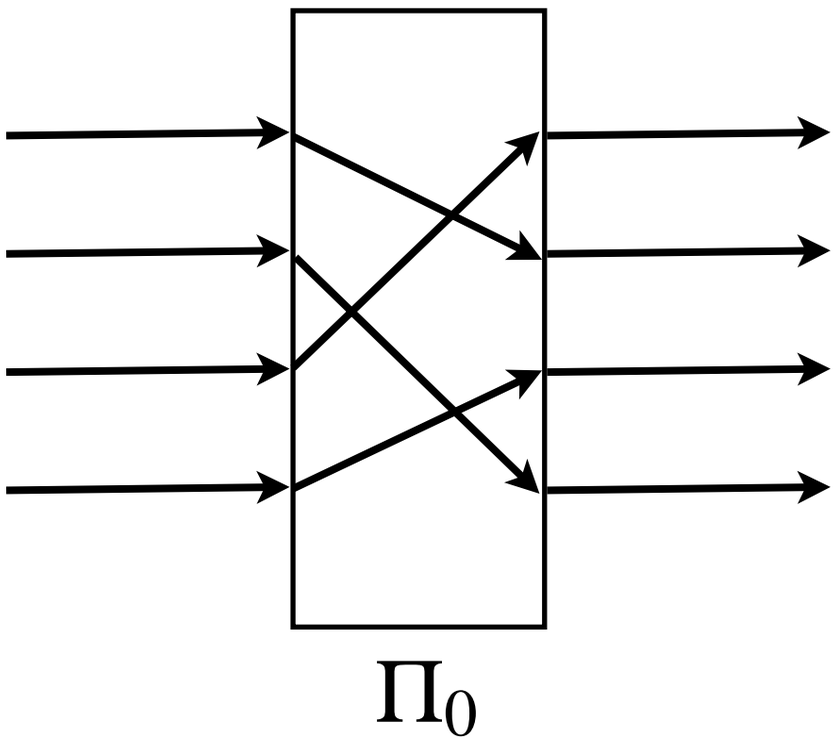
Permutations Π_i, where output is reencrypted.



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

Non-solution: reveal the mix

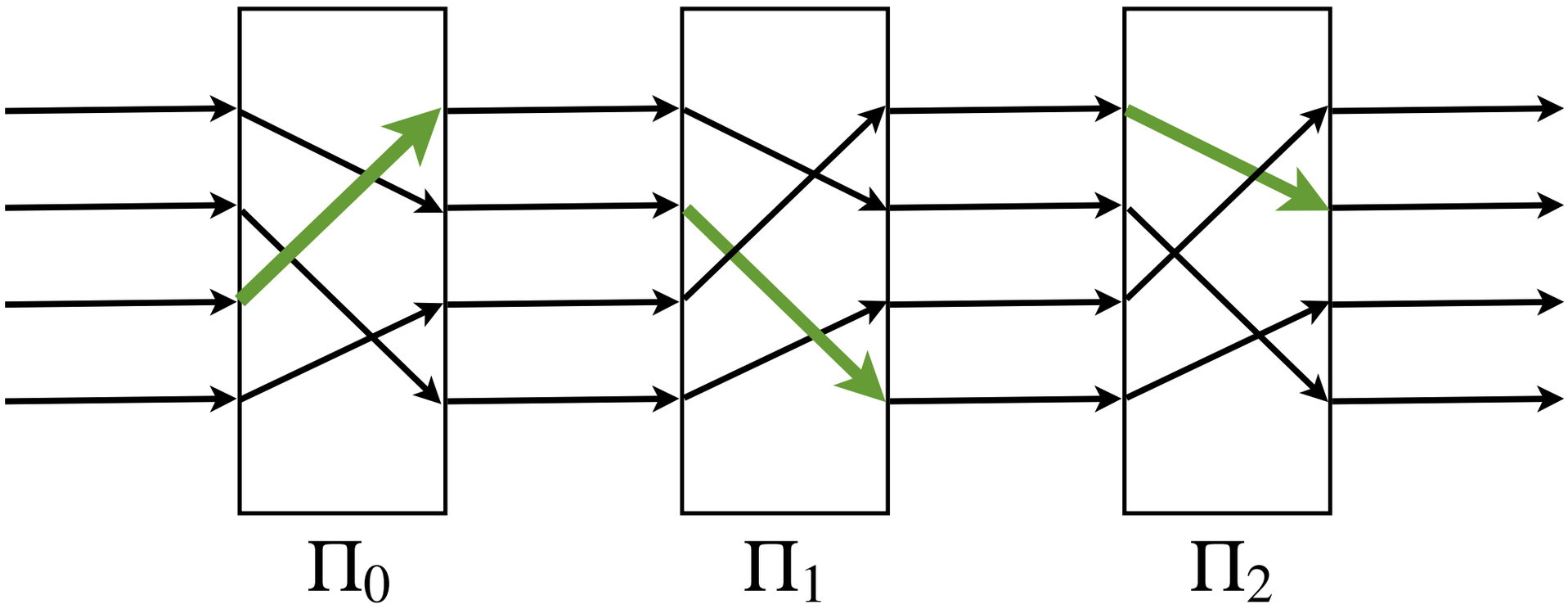
Publish the random numbers and the permutation.



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.

Hash functions

Widely used throughout cryptography.
Arbitrary string in.
Fixed-length string (e.g., 256 bits) out.

$$h_a = H(a)$$
$$h_b = H(b)$$

Usefulness: if $h_a \neq h_b$ then $a \neq b$.

Pre-image resistance: given h_a, it's infeasible to derive a.

Second pre-image resistance: given a and h_a, it's infeasible to derive $a' \neq a$ such that $H(a') = H(a)$

Simple use of hash functions

Tamper resistant storage: we'll store a and give you h_a

Hash chains: if new records include the hash of older records, then keep the latest somewhere safe and verify *all* older records.

Commitments: in a game like rock-paper-scissors, both parties first send the hash of their move, then reveal the moves later. (But be sure to include a random number!)

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

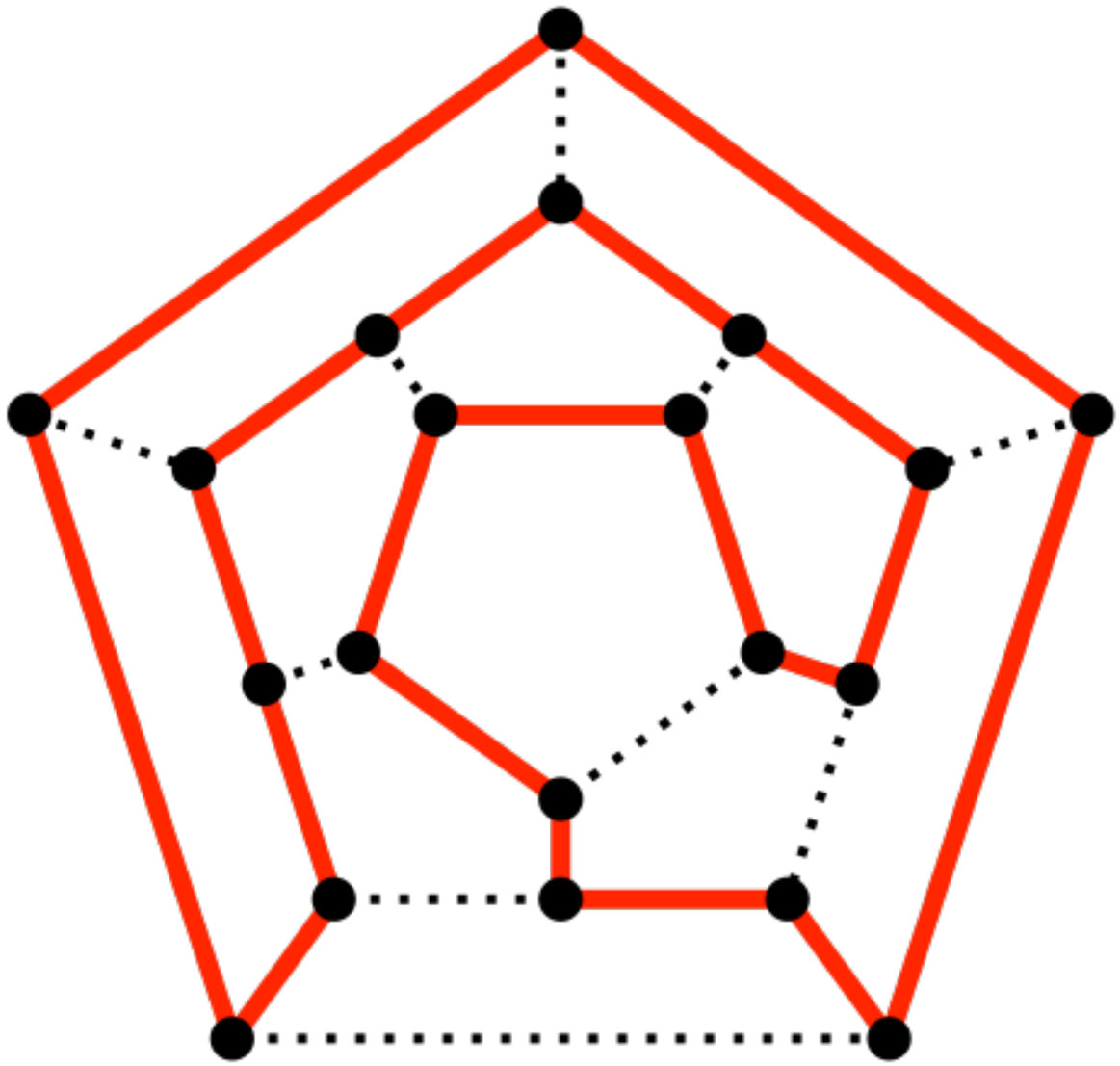
Example: Hamiltonian paths

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.

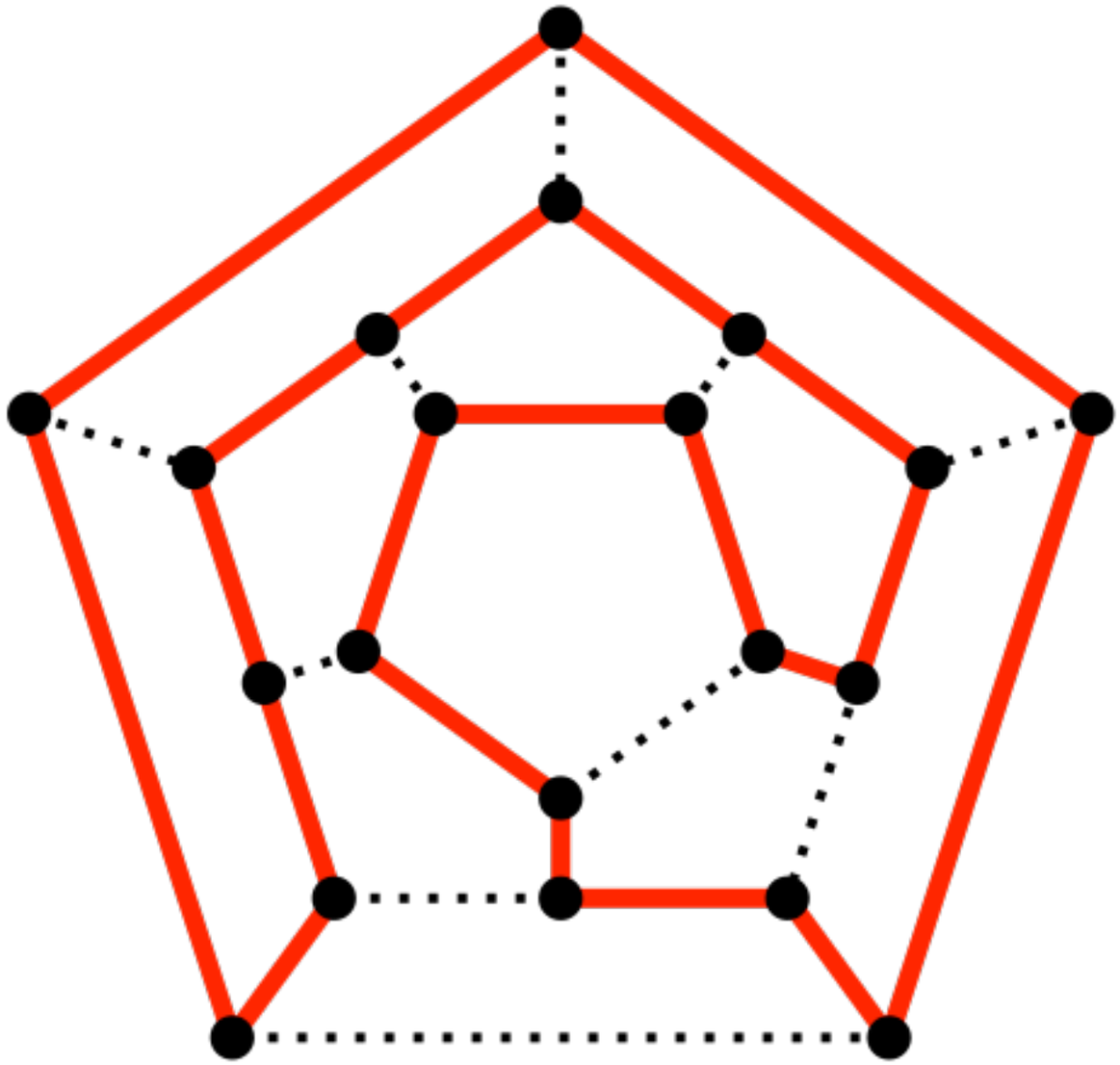


Non-interactive ZK proofs

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

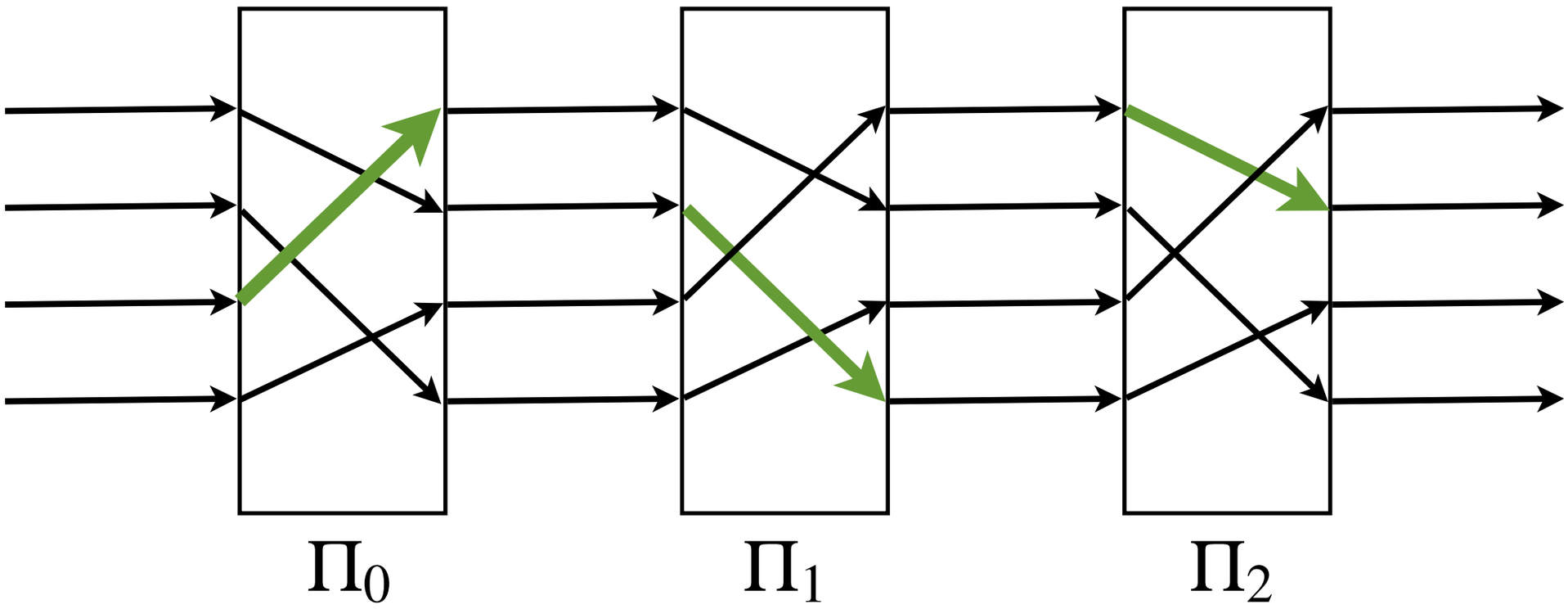
(Assumes good hash functions.)

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 $(A = g^w, B = h^w)$

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

Send (A,B) to ***V***

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

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

P: compute ~~$R = w + xc$~~ R

send R to V

V: Compute $A(g^x)^c$

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

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

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 s.t. $A = g^R (g^{xc})^{-1}$.

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

$$A(g^x)^c = g^w g^{xc} = g^{w+xc} = g^R$$

$$B(h^x)^c = h^w h^{xc} = h^{w+xc} = h^R$$

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 s.t. $A = g^R (g^{xc})^{-1}$.

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).

One newer, useful trick:

ballot challenge

ballot challenge

ballot challenge

a technique due to [Benaloh 2007]

ballot challenge

a technique due to [Benaloh 2007]

at the end, instead of casting your ballot:

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

ballot challenge

a technique due to [Benaloh 2007]

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”)

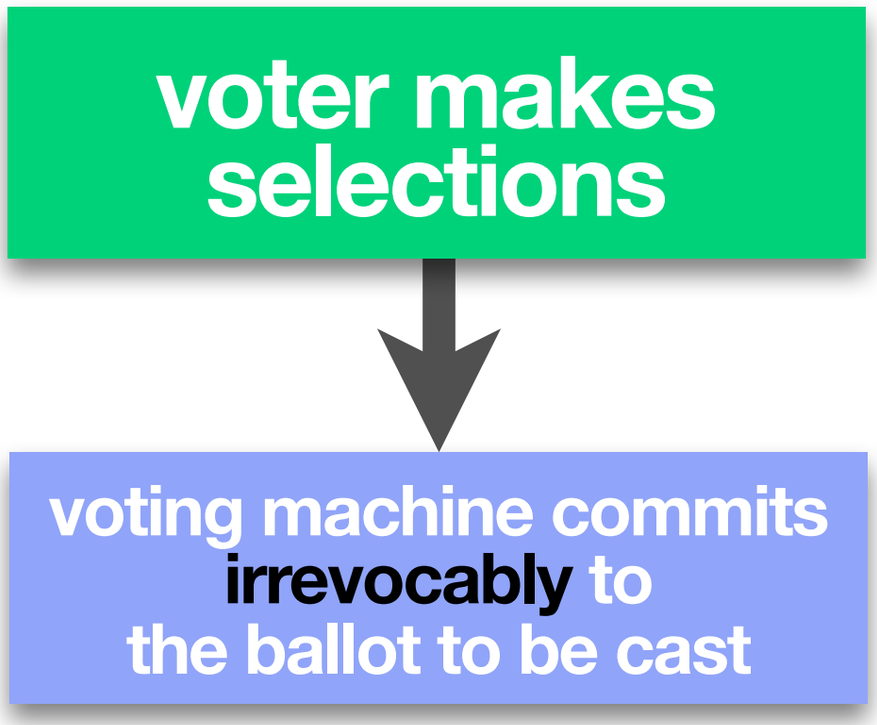
the voting machine cannot distinguish this from a real vote until the challenge

ballot challenge

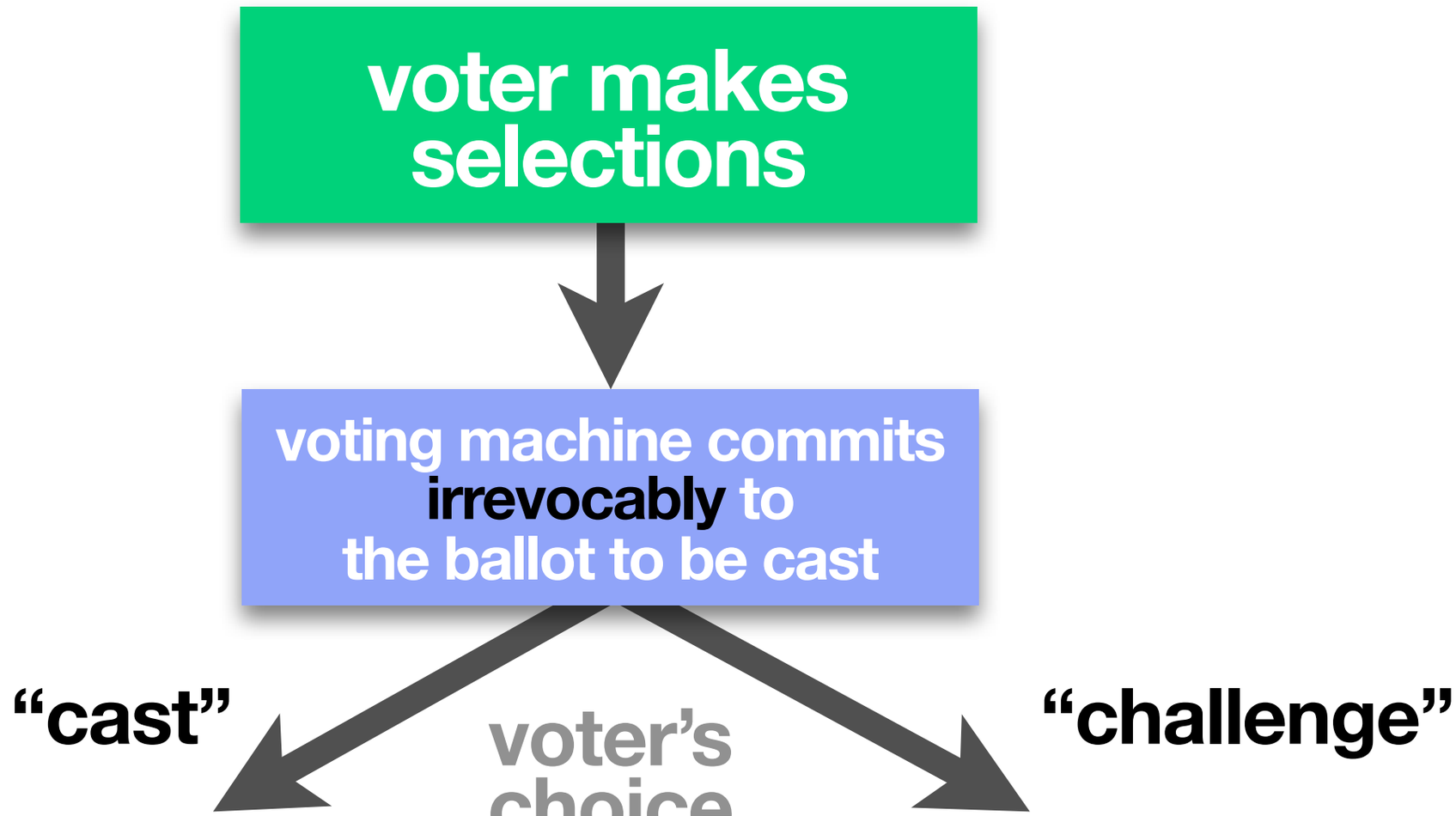
ballot challenge

voter makes selections

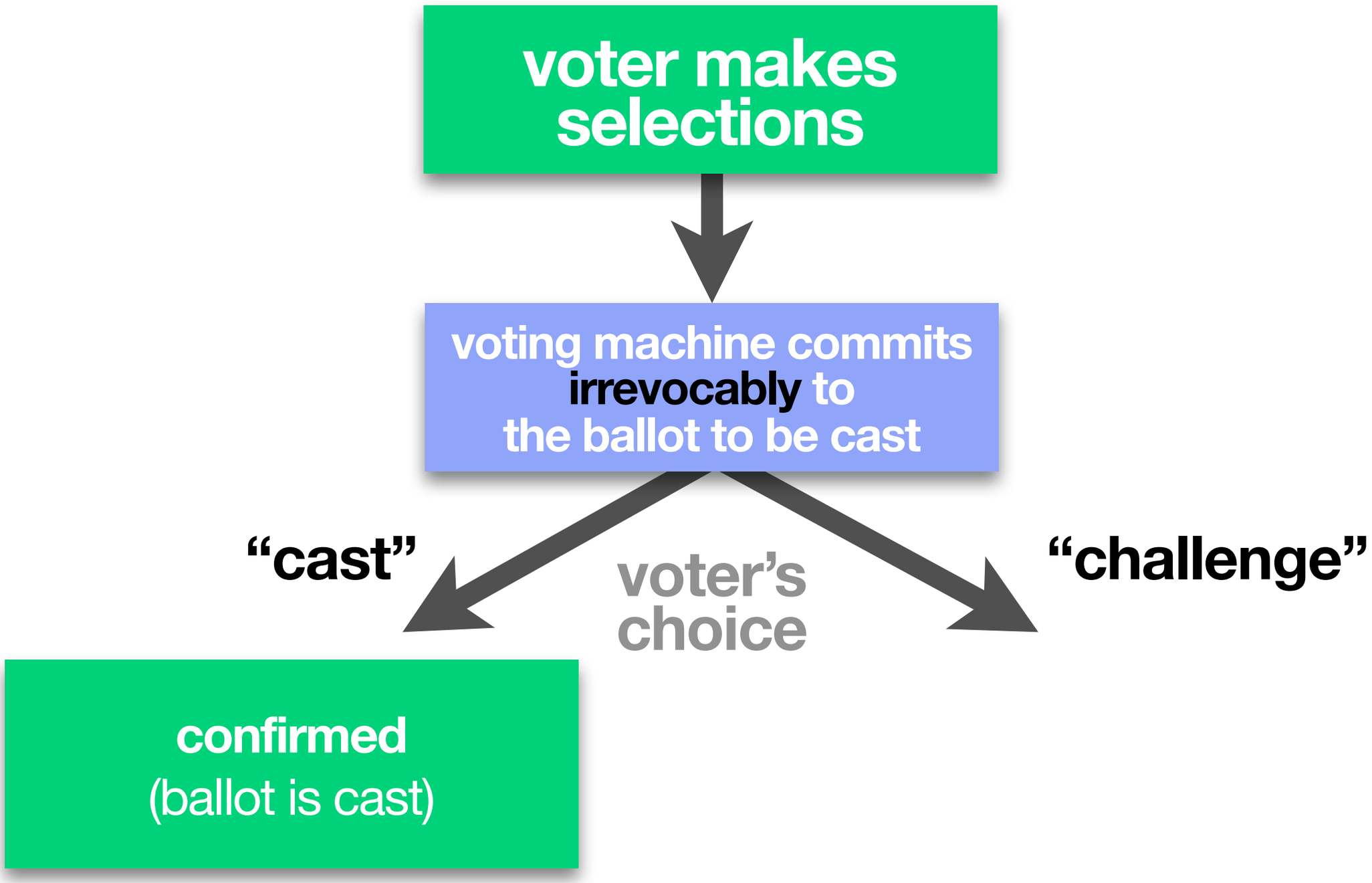
ballot challenge



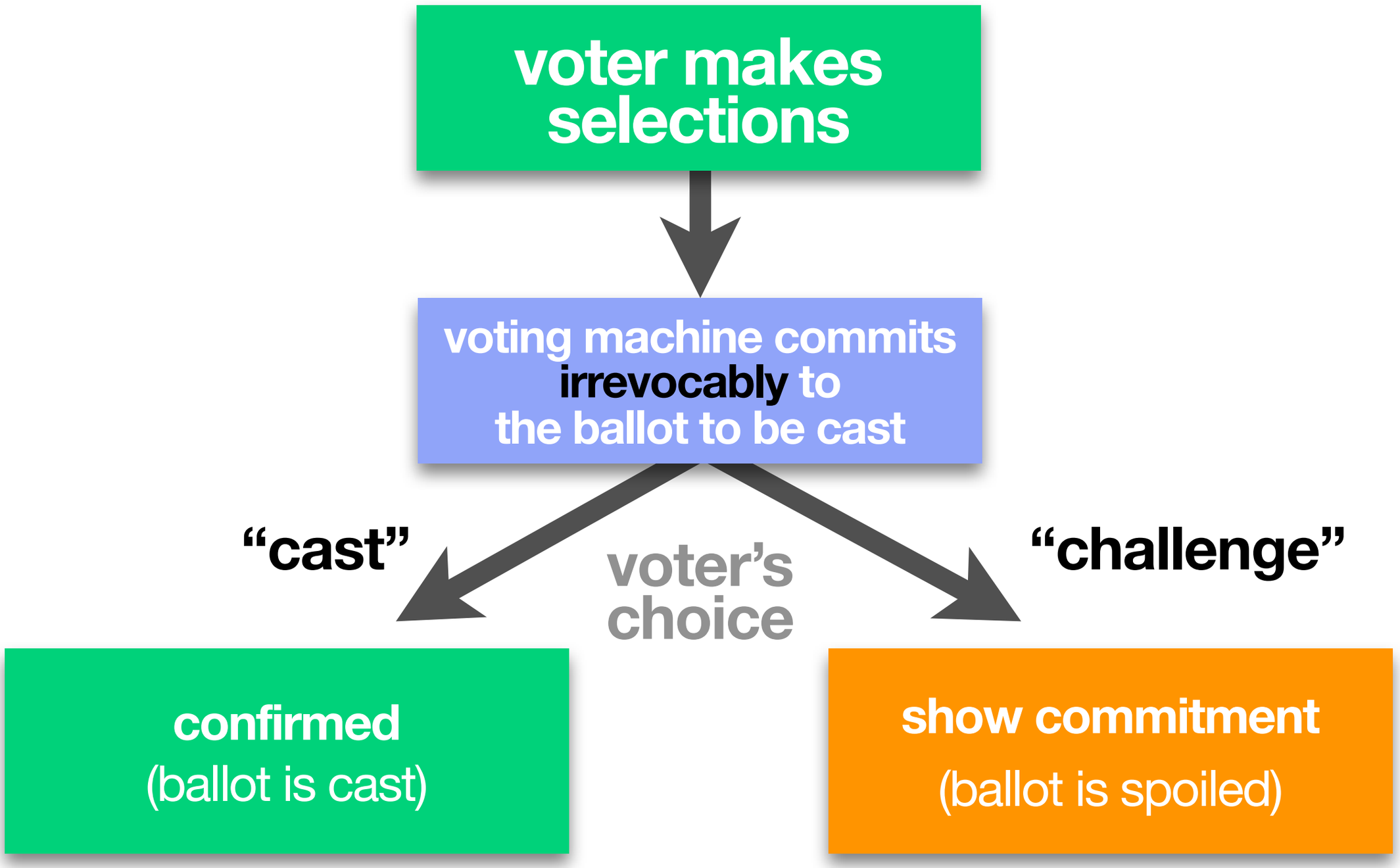
ballot challenge



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:

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

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

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

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

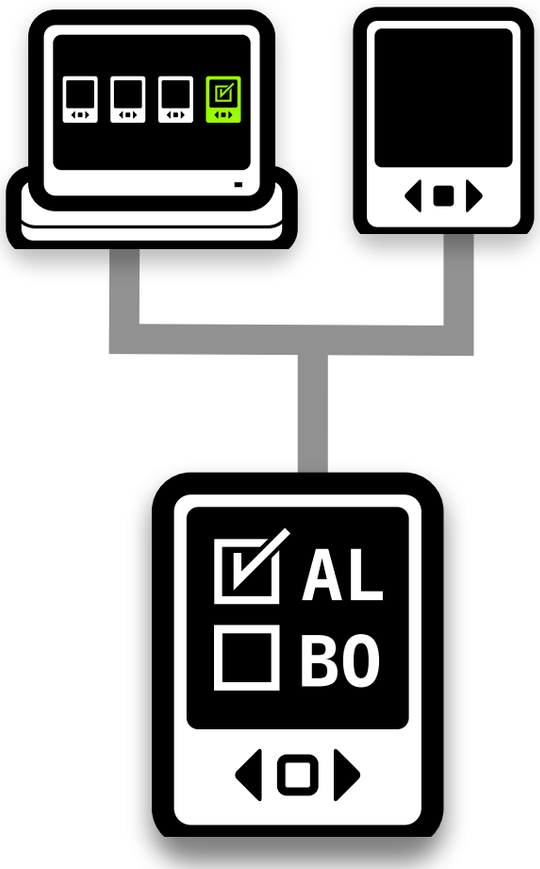
An **irrevocable** output medium

decrypting requires additional equipment

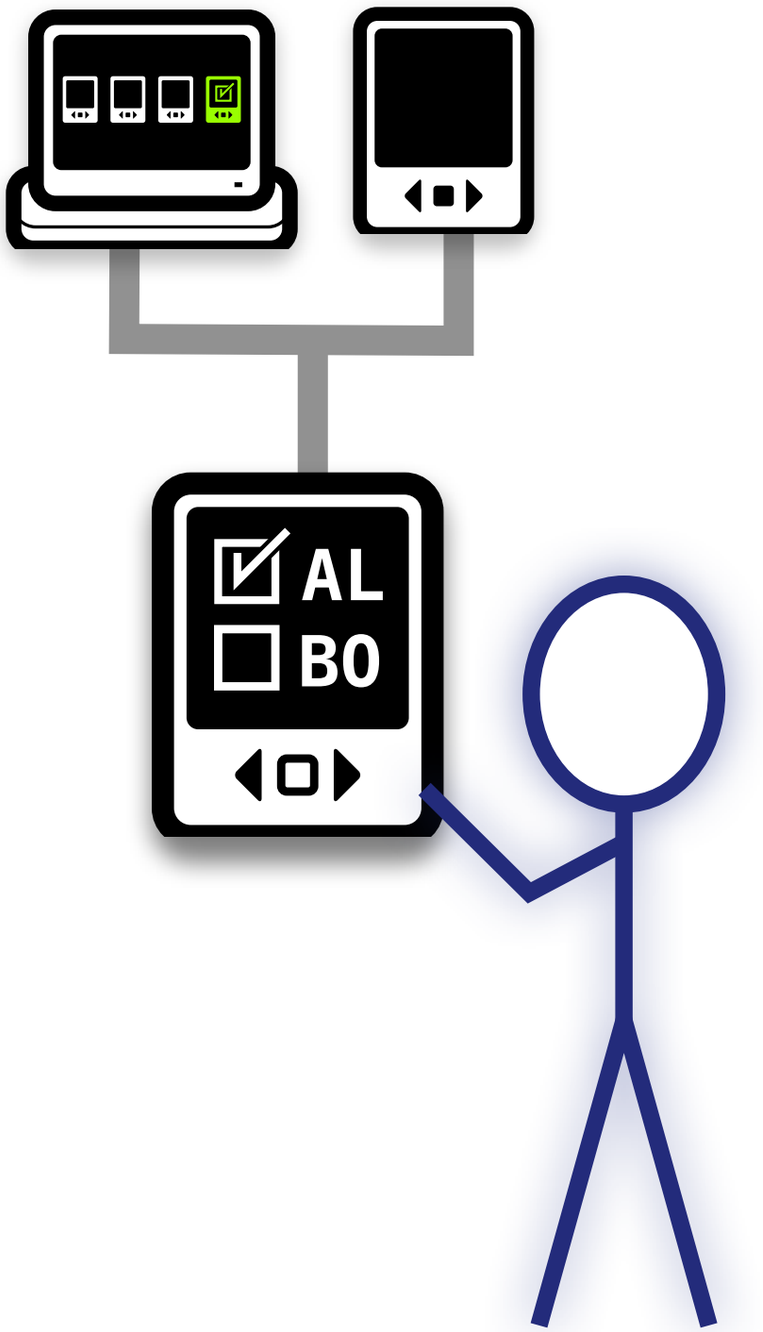
VoteBox happens to have its own irrevocable publishing system

(Its in-precinct LAN, where all machines replicate everywhere.)

polling place

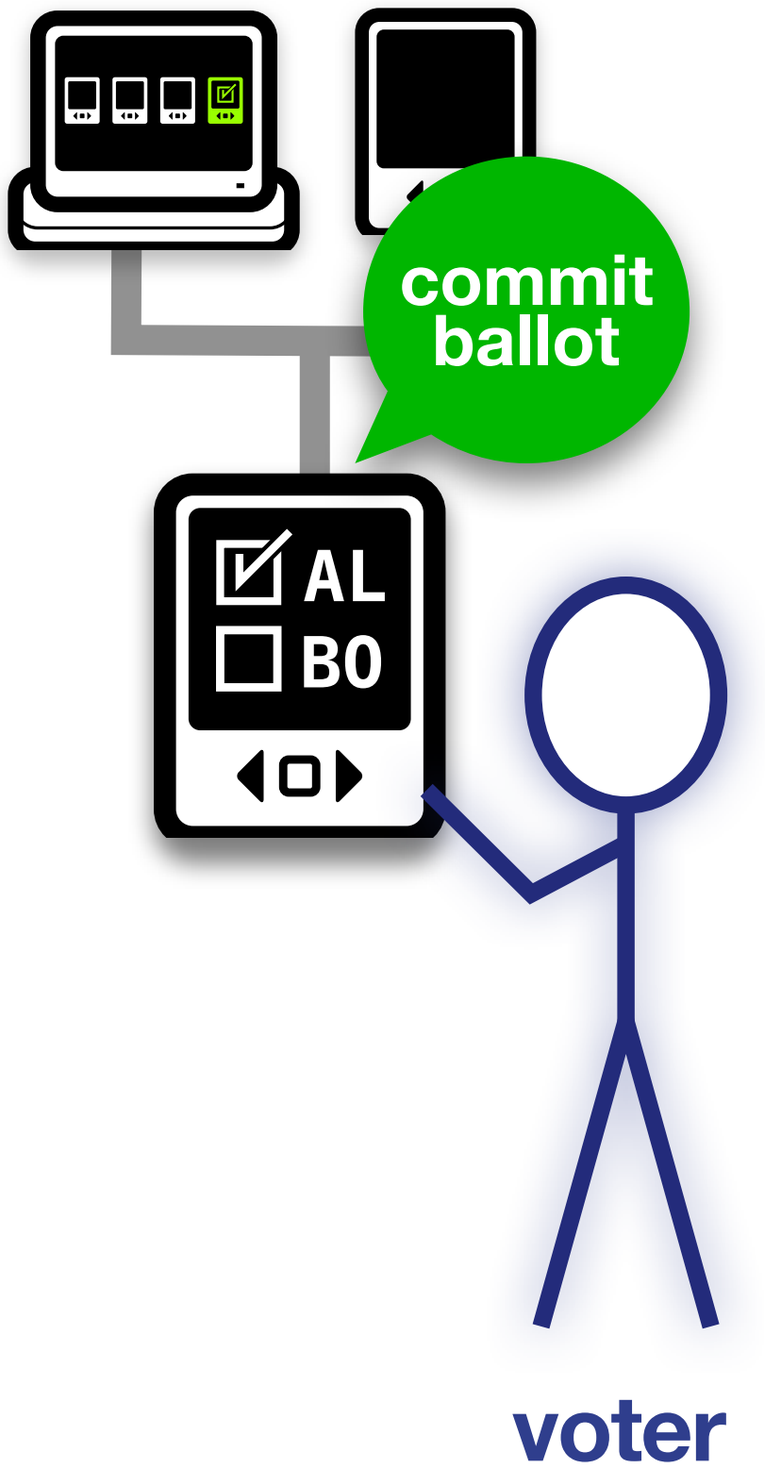


polling place



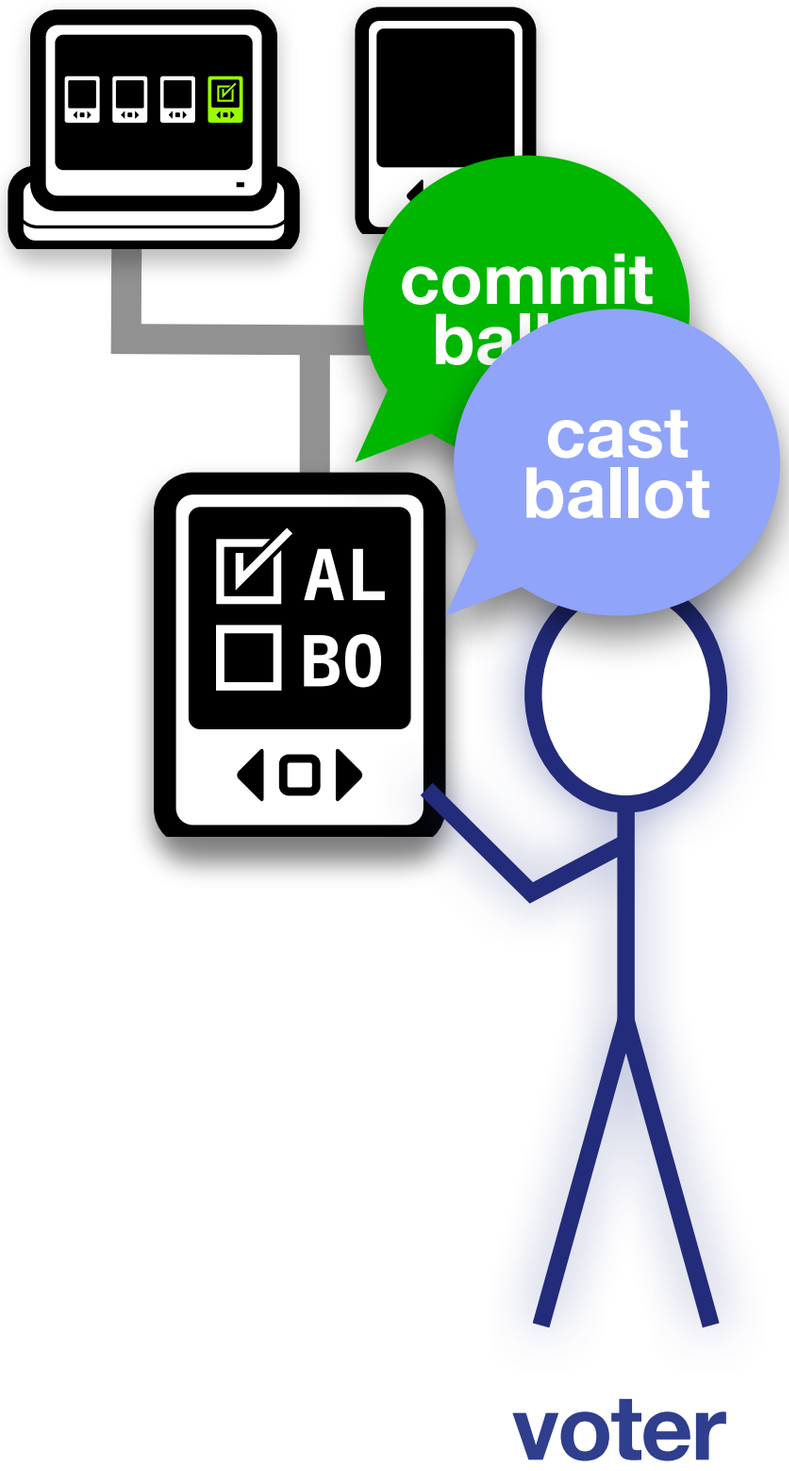
voter

polling place

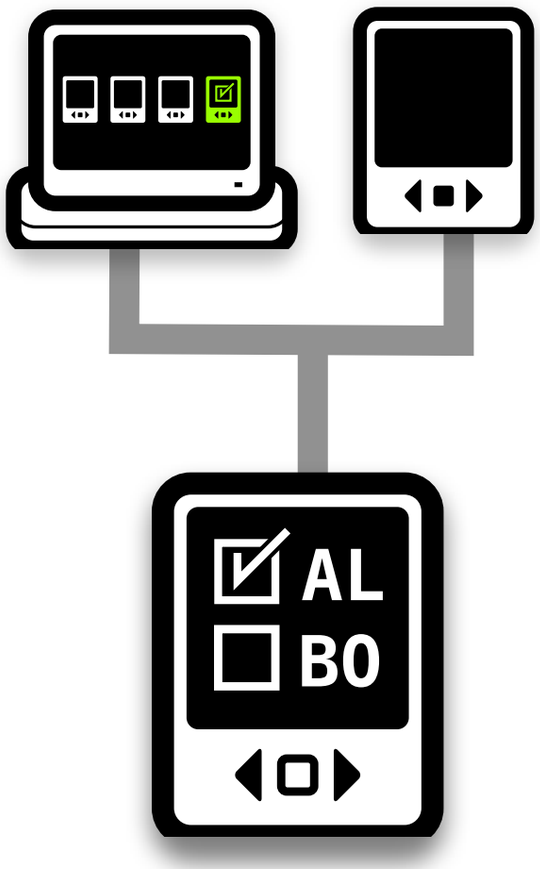


voter

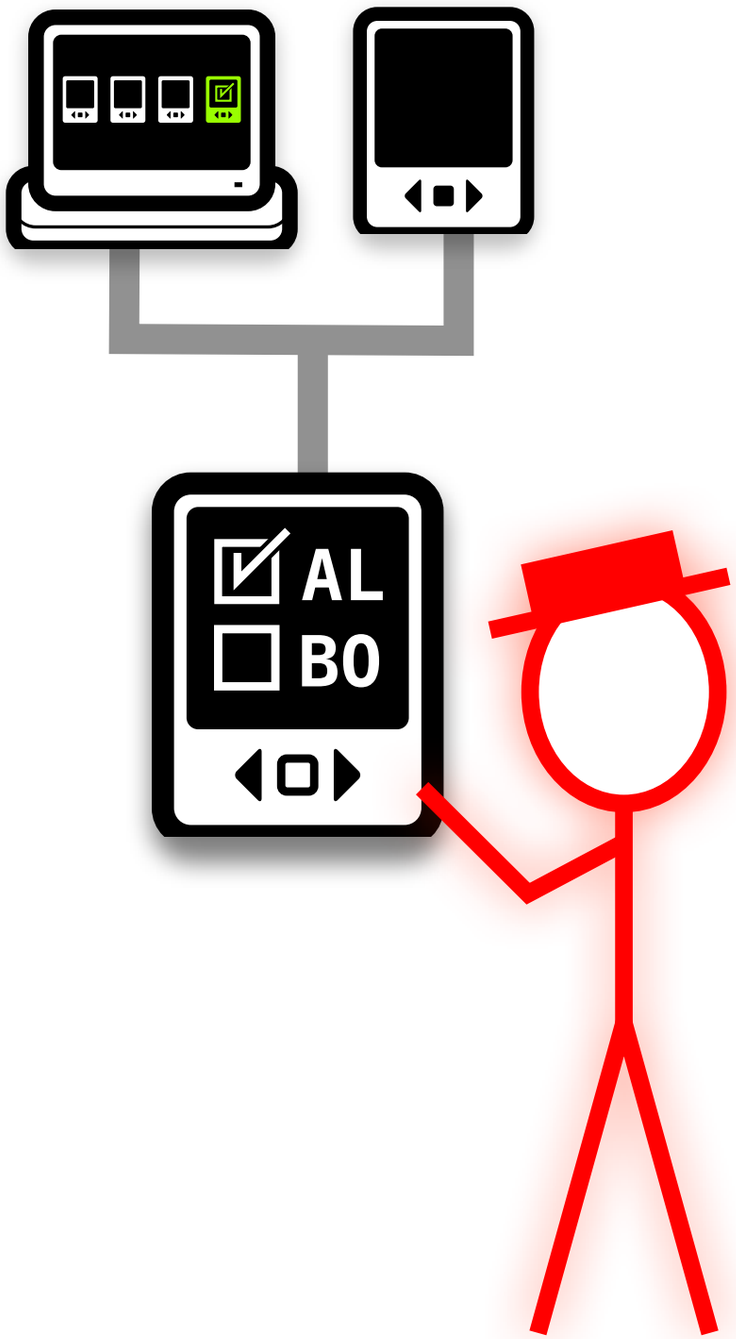
polling place



polling place

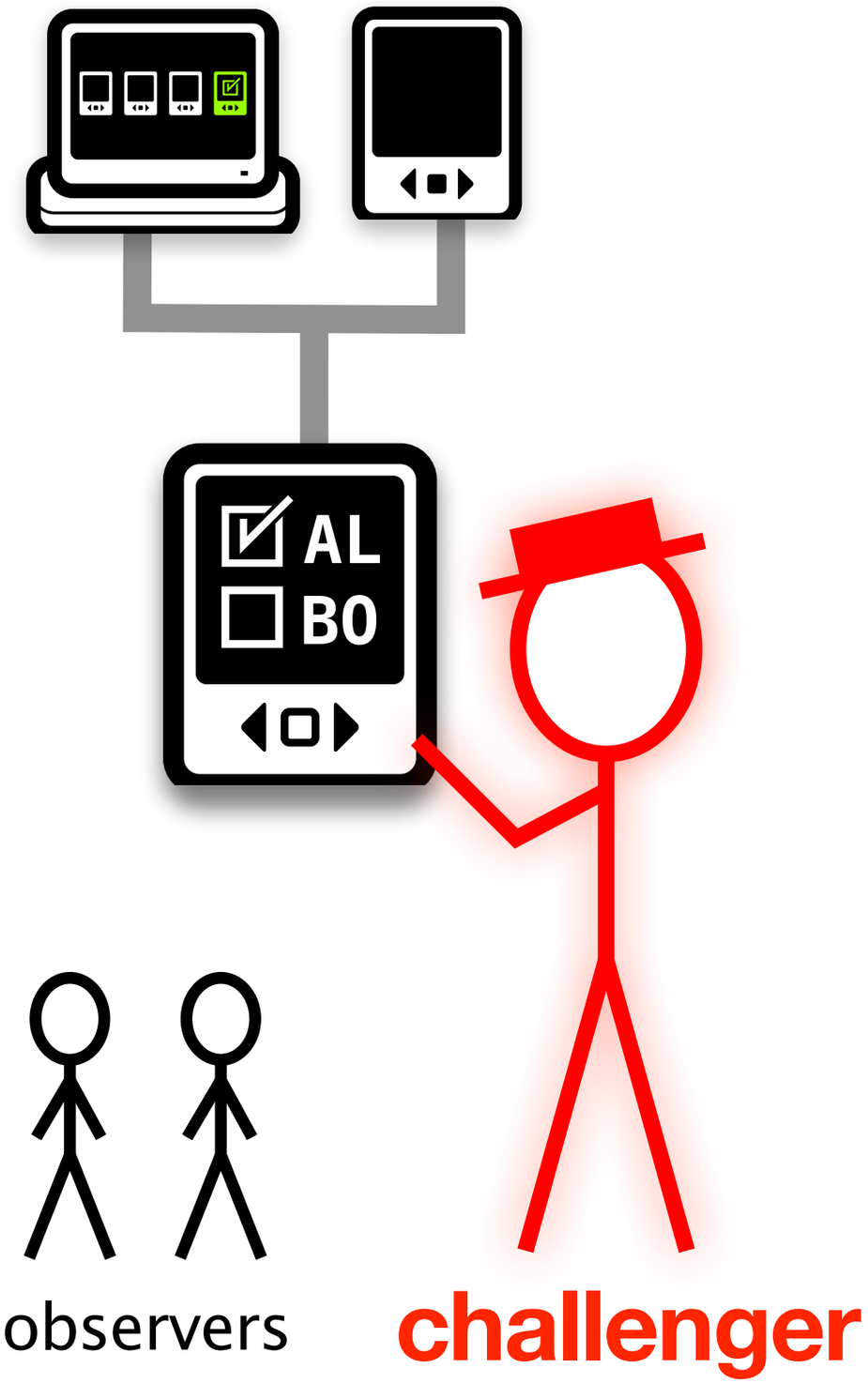


polling place



challenger

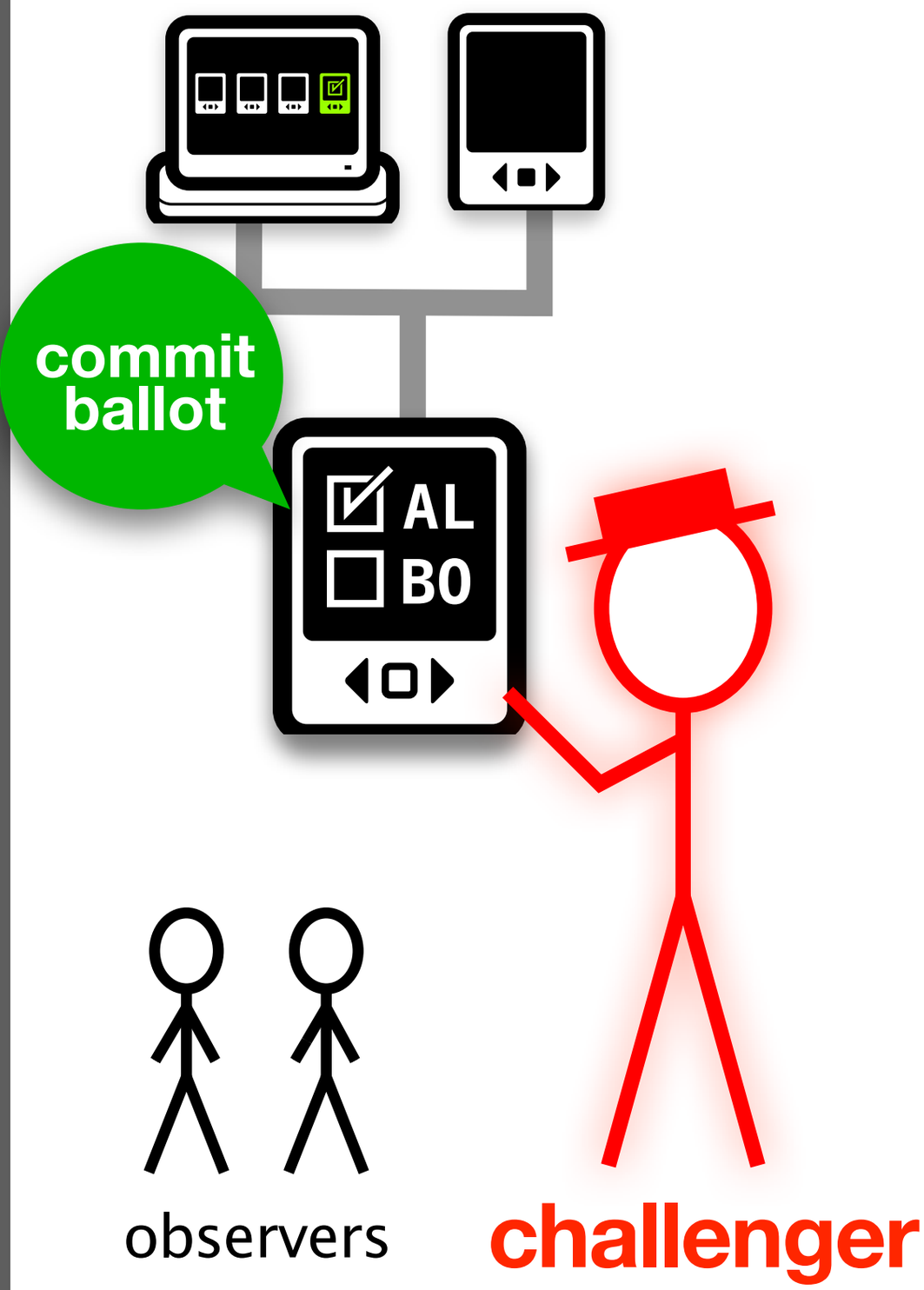
polling place



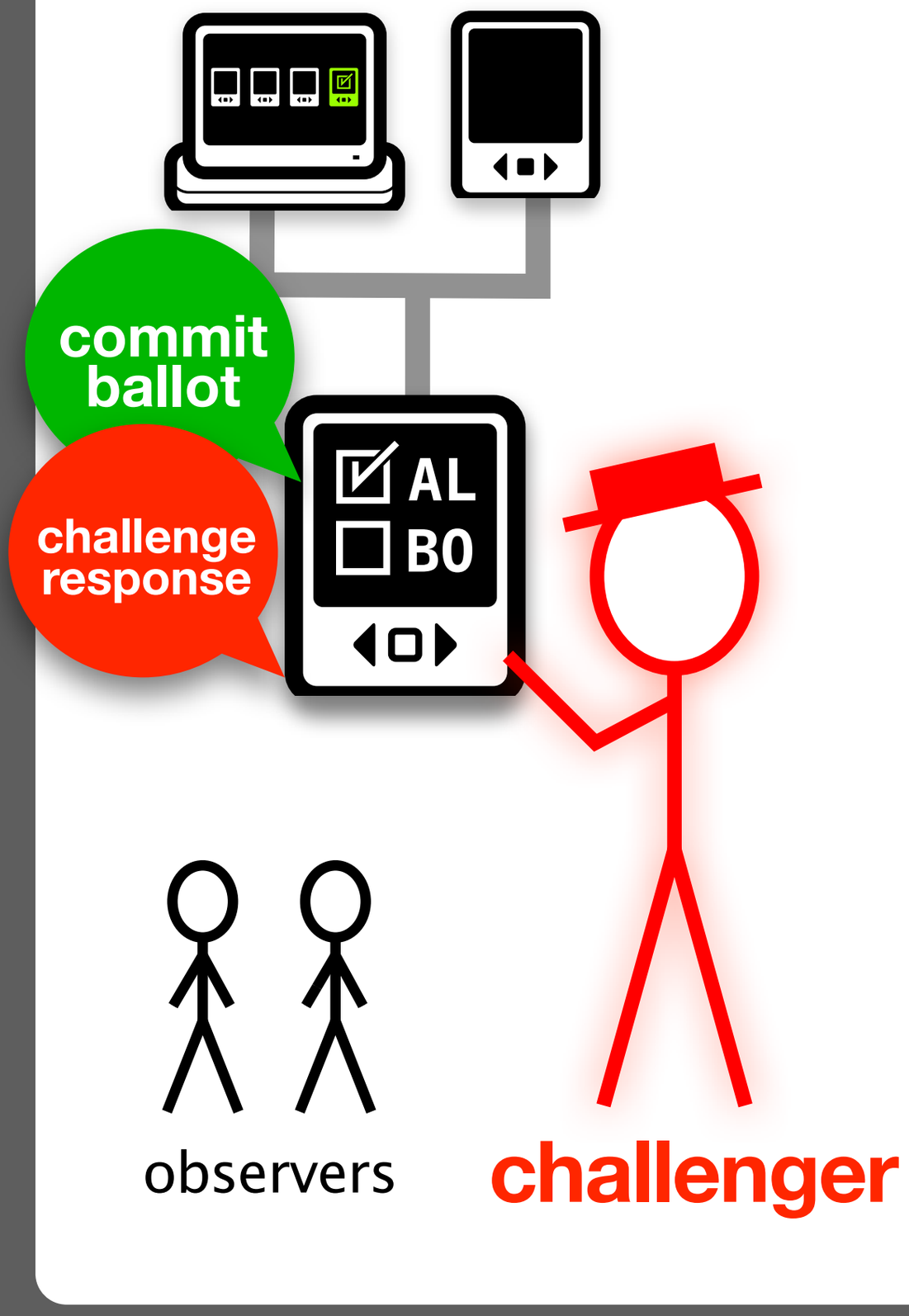
observers

challenger

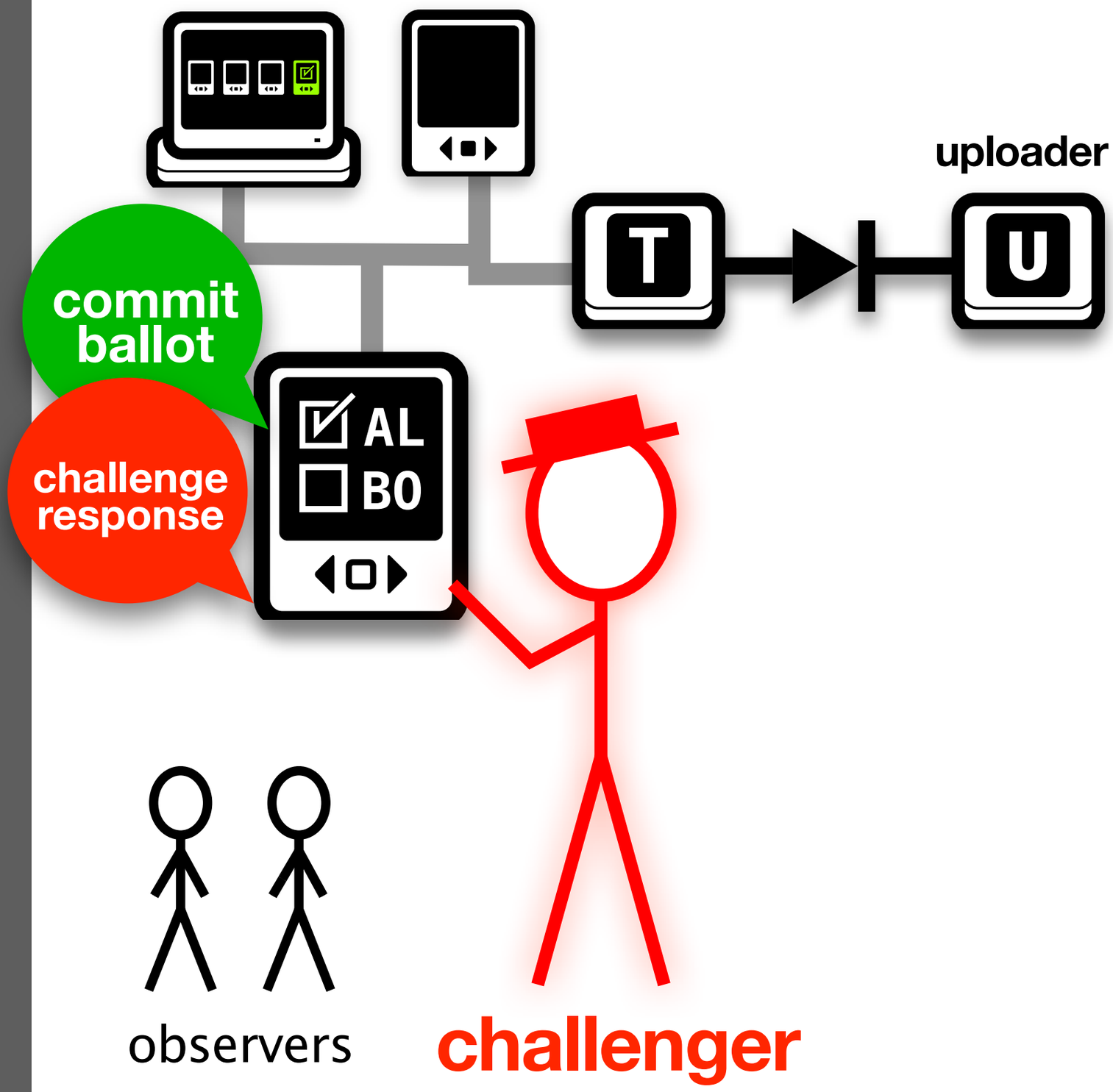
polling place



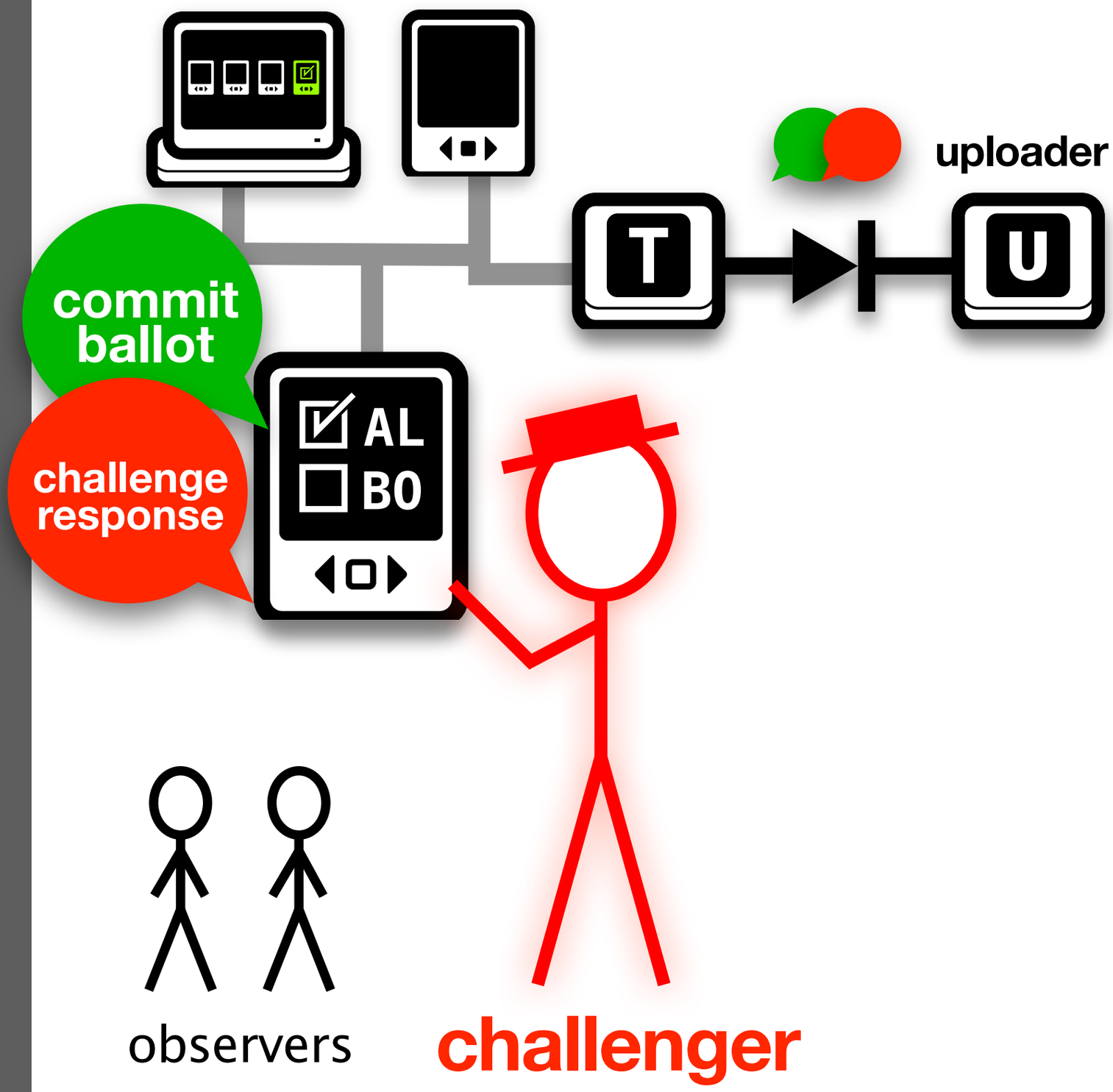
polling place



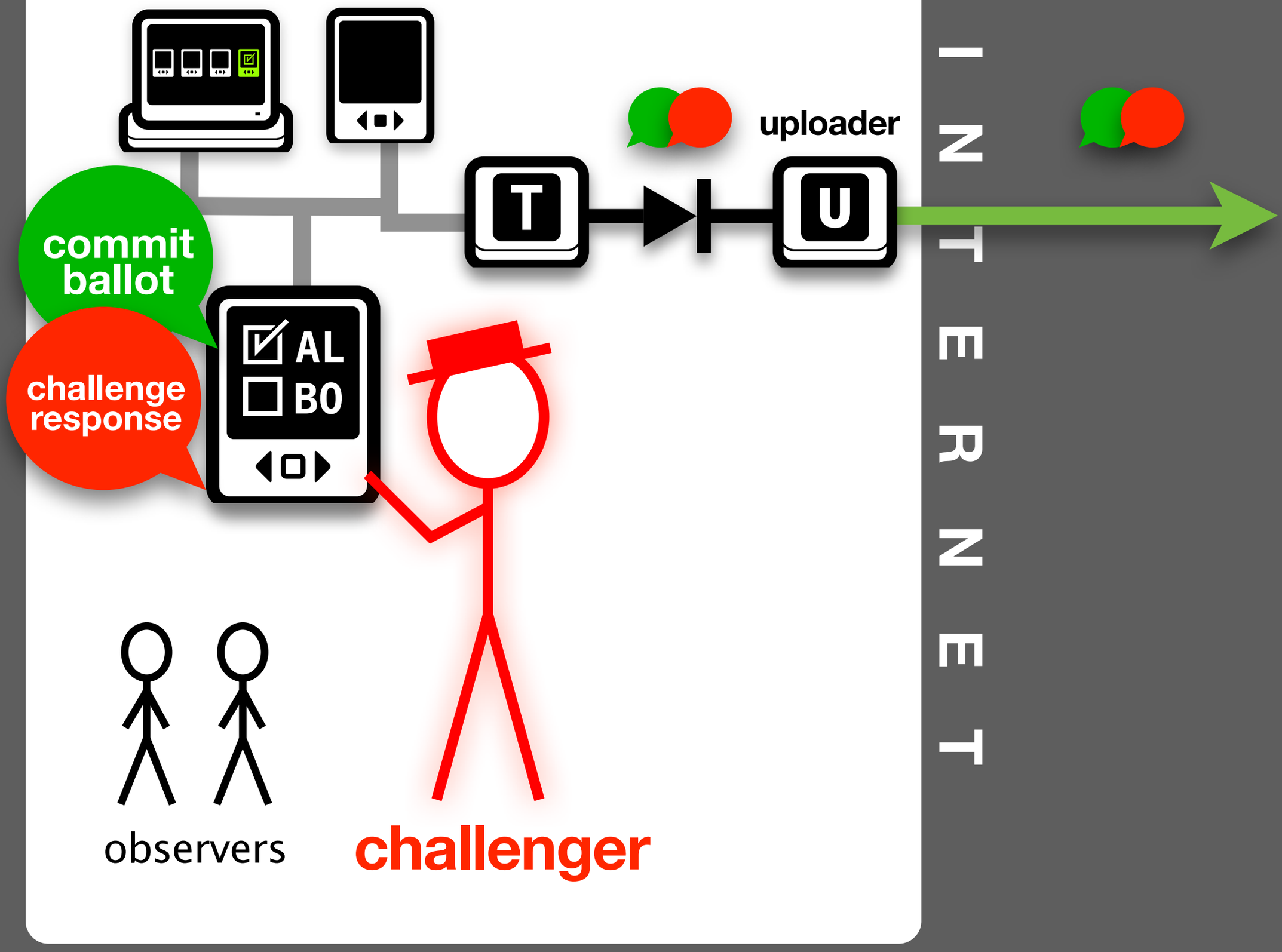
polling place



polling place



polling place



polling place

challenge center

INTERNET

uploader

T

U

commit ballot

challenge response

AL

BO

observers

challenger

polling place

challenge center

INTERNET

uploader

T

U

commit ballot

challenge response

AL

BO

challenge verification results

observers

challenger

polling place

challenge center

INTERNET

commit ballot

challenge response

T

uploader

U

AL

BO

ALICE

challenge verification results

observers

challenger