

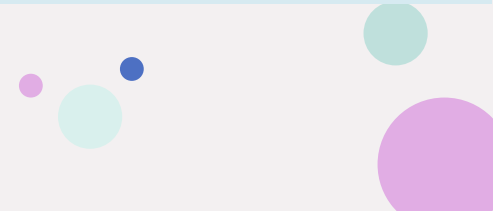


Lecture 21

E-Voting: Application of PKE, Sigma Protocols

CS409: Introduction to Cryptography

Sruthi Sekar



Physical Elections

Setting

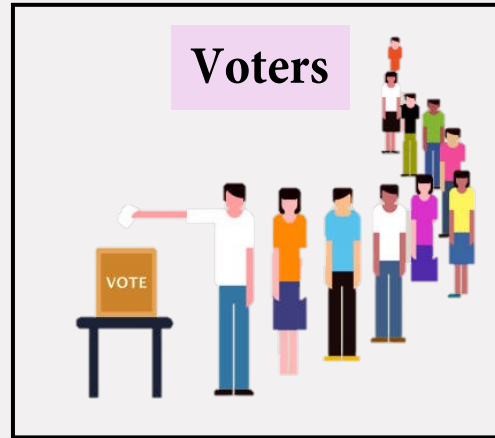
- **Registration:** Voter ID
- **Tallying** by EC

Assumption: Trust EC

Requirements

- **Ballot Privacy:** Ballot should not reveal details about the voter! (**voting machines ensure it**)
- **Verifiability:**
 1. Voter should be able to verify that his vote is counted.
 2. Verify that tallying is correct!

} **Relies on the assumption of trusting EC**
- **Integrity/Unique Vote:** Multiple-voting should not be possible! (**voting machines ensure it**)
- **Accessibility:** Anyone can vote (**Voting booths everywhere!**)



E-Voting

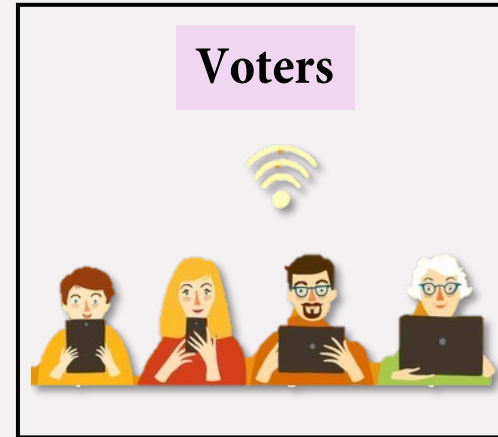
Setting

****How to authenticate?**

- **Registration:** managed by administrator (invitation by email)
- **Tallying by some trustees:** all of them need to be corrupt to cheat!

Requirements

- **Ballot Privacy:** Ballot should not reveal details about the voter! (use PKEs!)
- **Verifiability:**
 1. **Open auditability:** Voter should be able to verify (privately) that his vote is counted. (ballots posted on bulletin board)
 2. **End-to-end Verifiability:** Verify (publicly) that tallying is correct! (use ZKPs!)
- **Integrity/Unique Vote:** Multiple-voting should not be possible! (use ZKPs!)
- **Accessibility:** Anyone can vote (just need a device with internet connection!)



helios Voting: Registration

Trust the vote.

1. The administrator create a private election by creating specific voter list
(E.g. could be all IACR members, or all IITB members LDAP id)
2. Generate public parameters to enable encrypted voting (will discuss more shortly)!
3. Email the voting invite to each voter in the list with a **voter's username and password**, a URL to voting booth and an SHA – 1 hash of election parameters! (or ask voter to authenticate using LDAP id)

Create a New Election	
Use voter aliases:	☐ If selected, voter identities will be replaced with aliases, e.g. "V12", in the ballot tracking center
Randomize answer order:	☐ enable this if you want the answers to questions to appear in random order for each voter
Private?	☐ A private election is only visible to registered voters.



(show example demo!)

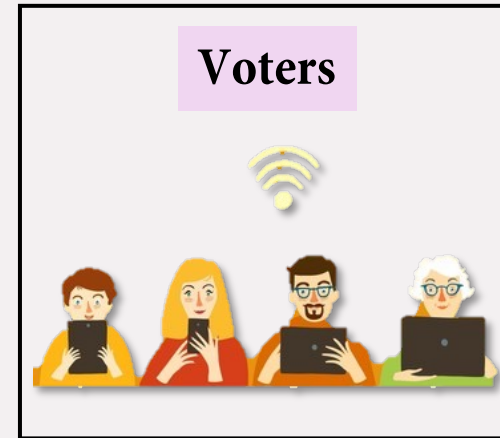
E-Voting

Setting

- **Registration:** managed by administrator (invitation by email)
- **Tallying by some trustees:** all of them need to be corrupt to cheat!

Requirements

- **Ballot Privacy:** Ballot should not reveal details about the voter! (use PKEs!)
- **Verifiability:**
 1. **Open auditability:** Voter should be able to verify (privately) that his vote is counted.
(ballots posted on bulletin board)
 2. **End-to-end Verifiability:** Verify (publicly) that tallying is correct! (use ZKPs!)
- **Integrity/Unique Vote:** Multiple-voting should not be possible! (use ZKPs!)
- **Accessibility:** Anyone can vote (just need a device with internet connection!)



****What PKE?**



helios Voting: Ballot Privacy

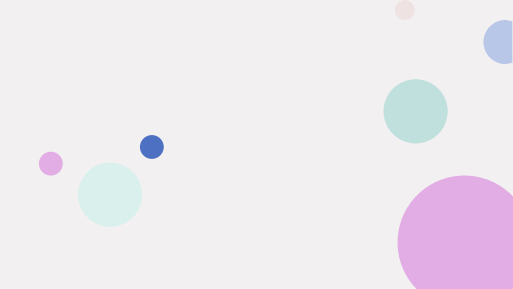
Trust the vote.

- I:** The trustee(s) post a public key that all voters can use to encrypt their votes. To ensure privacy of votes, the tallying should be done on encrypted votes before decrypting to the final outcome!

Homomorphic Public Key Encryption

- II:** Having just a single trustee who holds the decryption key breaks privacy! The trustee can just decrypt individual votes! Must have multiple trustees, at least one of whom is trusted!

Public Key Encryption allowing distributed decryption



helios Voting: Ballot Privacy

Trust the vote.

$$\text{Gen}(1^n): \mathcal{G}(1^n) \rightarrow (\mathbb{G}, q, g), x \in_R \mathbb{Z}_q$$
$$\text{pk} = (\mathbb{G}, g, X = g^x), \text{ sk} = (\mathbb{G}, g, x)$$

$$\text{Enc}(\text{pk}, m): y \leftarrow \mathbb{Z}_q$$
$$(Y = g^y, C = mX^y)$$

$$\text{Dec}(\text{sk}, (Y, C)): CY^{-x}$$

RECALL

El Gamal Encryption

I: Will this allow us to homomorphically tally?
(i.e., aggregate encrypted votes into an encryption of the outcome, which can later be decrypted?)

(Vote will be (yes)1/0(no))

How to correctly encode the votes?

helios Voting: Ballot Privacy

Trust the vote.

$\text{Gen}(1^n): \mathcal{G}(1^n) \rightarrow (\mathbb{G}, q, g), x \in_R \mathbb{Z}_q$
 $\text{pk} = (\mathbb{G}, g, X = g^x), \text{sk} = (\mathbb{G}, g, x)$

$\text{Enc}(\text{pk}, g^b): y \leftarrow \mathbb{Z}_q; b \in \{0,1\}$
 $(Y = g^y, C = g^b X^y)$

No-vote encoded as: g^0
Yes-vote encoded as: g^1

$\text{Dec}(\text{sk}, (Y, C)): CY^{-x}$

Exponential El Gamal Encryption

Homomorphism (Tallying)

Given $\text{CT}_{b_1} = (Y_1, C_1 = g^{b_1} X^{y_1})$ and $\text{CT}_{b_2} = (Y_2, C_2 = g^{b_2} X^{y_2})$ under same pk:
 $\text{CT}_{b_1} \cdot \text{CT}_{b_2} = (g^{y_1+y_2}, g^{b_1+b_2} \cdot X^{y_1+y_2}) = \text{CT}_{b_1+b_2}$

Decrypting Tallied Votes

$\text{Dec}(\text{sk}, \text{CT}_{\sum_i b_i}) := g^{\sum_i b_i}$. Extract $\sum_i b_i$ using **baby-step giant-step**!**

****Shanks' baby-step giant-step algorithm can extract 40-bit discrete log in seconds on a standard laptop!**

helios Voting: Ballot Privacy

Trust the vote.

Key Generation: $\mathcal{G}(1^n) \rightarrow (\mathbb{G}, q, g)$ sent to all T_i . Each trustee T_i picks $x_i \in_R \mathbb{Z}_q$ and sets $pk_i = (\mathbb{G}, g, X_i = g^{x_i})$, $sk_i = (\mathbb{G}, g, x_i)$. Publish pk_i 's.
Election public key (sent to voters): $epk = (\mathbb{G}, g, X = \prod_i X_i = g^{\sum_i x_i})$

Enc(pk, g^b): $y \leftarrow \mathbb{Z}_q; b \in \{0,1\}$
($Y = g^y, C = g^b X^y$)

No-vote encoded as: g^0
Yes-vote encoded as: g^1

Decryption: Given (Y, C) each trustee T_i computes and publishes $D_i = Y^{x_i}$
Plaintext recovered as: $C / \prod_i D_i$

Threshold Exponential El Gamal Encryption

II: How to distribute the decryption by distributing sk amongst the trustees $T_1, T_2, \dots, T_n$?
(Atleast on trustee T_i must be trusted to be honest!)

III: Does this guarantee ballot privacy?
(pk_n, sk_n) not well-formed! $\leftarrow$

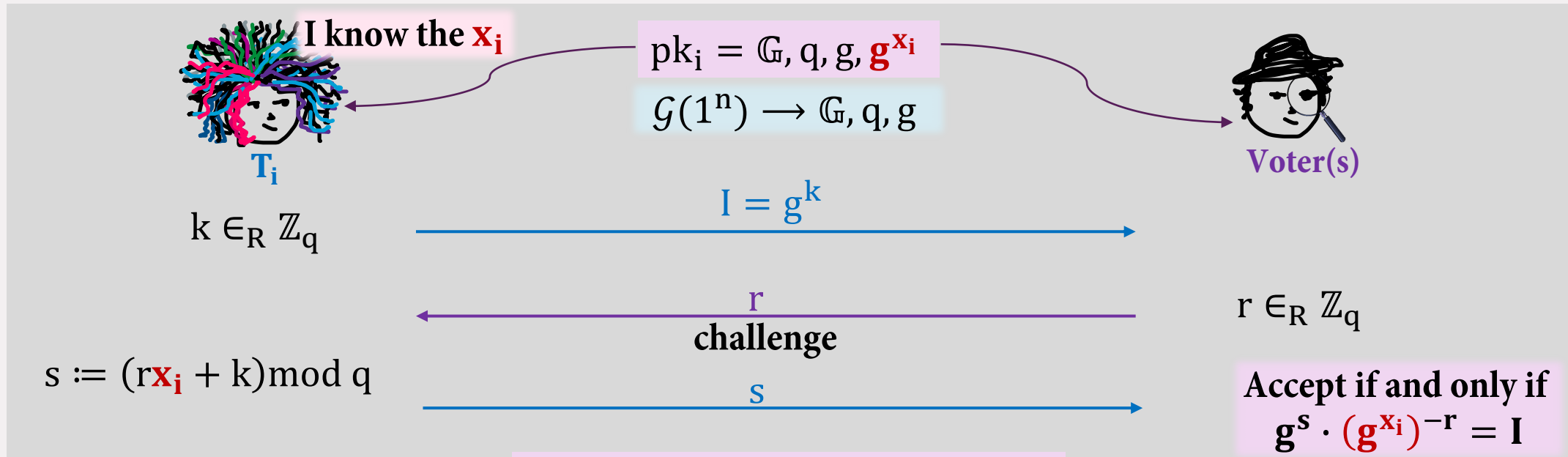
If a single rogue T_n sees $(n-1)$ pk_i 's, generates $x_n \in_R \mathbb{Z}_q$ and sets $X_n = g^{x_n} / (\prod_{i=1}^{n-1} X_i)$ which leads to $epk = (\mathbb{G}, g, g^{x_n})$
($\therefore T_n$ can decrypt each vote!)

helios Voting: Ballot Privacy

Trust the vote.

Well formedness of (pk_i, sk_i) : Each T_i proves that it knows the $sk_i (x_i)$ corresponding to their public key (g^{x_i}) !

We already saw how to do this! Schnorr ZKPoK+Fiat-Shamir (to make non-interactive)



RECALL

Schnorr's ZKPoK for L_{Dlog} : Π_{Dlog}

$$L_{Dlog} = \{h \in \mathbb{G} : \exists x \in \mathbb{Z}_q \text{ s.t. } g^x = h\}$$

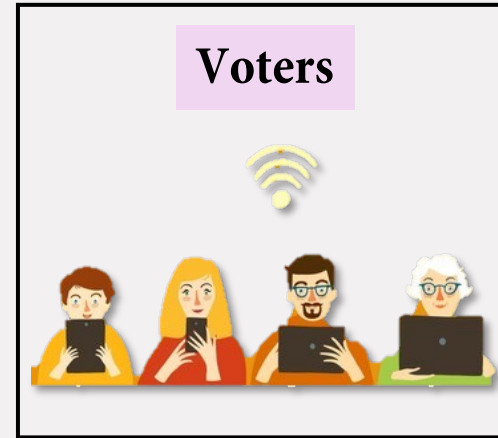
E-Voting

Setting

- ✓ **Registration:** managed by administrator (invitation by email)
- ✓ **Tallying by trustees:** all of them need to be corrupt to cheat!

Requirements

- ✓ **Ballot Privacy:** Ballot should not reveal details about the voter! (use PKEs!)
- **Verifiability:**
 - ✓ 1. **Open auditability:** Voter should be able to verify (privately) that his vote is counted.
(ballots posted on bulletin board)
 - 2. **End-to-end Verifiability:** Verify (publicly) that tallying is correct! (use ZKPs!)
- **Integrity/Unique Vote:** Multiple-voting should not be possible! (use ZKPs!)
- ✓ **Accessibility:** Anyone can vote (just need a device with internet connection!)



helios Voting: Verifiability

Trust the vote.

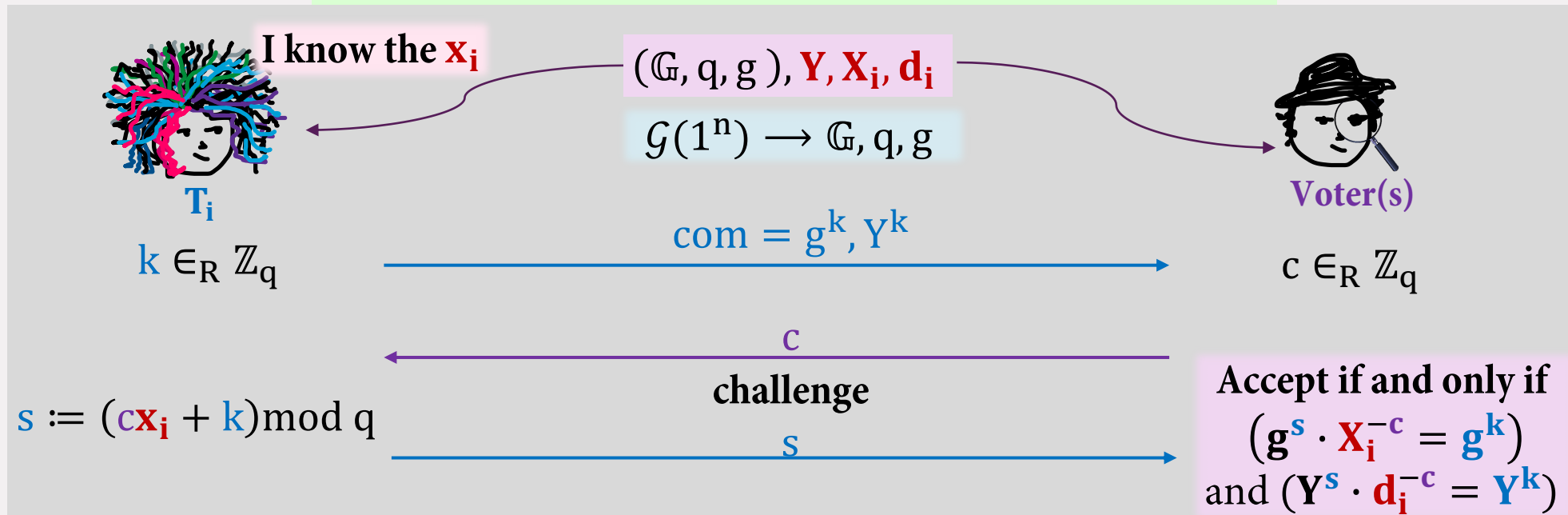
What if the trustees do not tally the votes correctly?

While decrypting, given (Y, C) , if a **rogue** T_n computes and publishes $d_i = Y^{x_i}/g^v$ for some v , then the plaintext decrypted will be $(C/(\prod_i Y^{x_i})) \cdot g^v$, adding v votes to the count!

It can use this attack to add and subtract votes and keep the total vote count consistent!

$\therefore$ make each T_i prove that the discrete log of X_i in base g is same as the discrete log of d_i in base Y

Chaum-Pedersen ZkPoK for $L_{DH}(\Pi_{DH})$ + Fiat Shamir



Sigma Protocol for DH Tuple: Chaum-Pedersen

Theorem 1: Chaum-Pedersen Σ protocol

Π_{DH} is a Σ -protocol for $\mathcal{R}_{\text{DH}}$.

Proof Sketch

Completeness: H.W.

Honest Verifier Zero-Knowledge:



$\text{Sim}(Y, X_i, d_i)$: (out-of-order) $c, s \in_R \mathbb{Z}_q$
output: $((g^s \cdot X_i^{-c}), Y^s \cdot d_i^{-c}), c, s$

$\text{View}_V(\langle P, V \rangle(Y, X_i, d_i)) := ((g^k, Y^k), c, s = cx_i + k \bmod q)$

Special Soundness: (exactly like Schnorr)



Ext

$\text{Ext}((Y, X_i, d_i), (\text{com} = (g^k, h^k), c, s), (\text{com}, c', s'))$:

- We know $c \neq c'$ and both are accepting transcripts.
 $\because g^s X_i^{-c} = g^k = g^{s'} X_i^{-c'} \quad \& \quad Y^s d_i^{-c} = Y^k = Y^{s'} d_i^{-c'}$
- $\Rightarrow \log_g X_i = \log_Y d_i = (s - s')(c - c')^{-1} \bmod q$

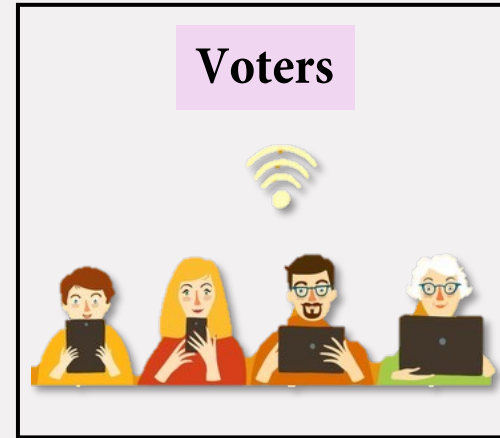
E-Voting

Setting

- ✓ **Registration:** managed by administrator (invitation by email)
- ✓ **Tallying by trustees:** all of them need to be corrupt to cheat!

Requirements

- ✓ **Ballot Privacy:** Ballot should not reveal details about the voter! (use PKEs!)
- **Verifiability:**
 - ✓ 1. **Open auditability:** Voter should be able to verify (privately) that his vote is counted. (ballots posted on bulletin board)
 - ✓ 2. **End-to-end Verifiability:** Verify (publicly) that tallying is correct! (use ZKPs!)
- **Integrity/Unique Vote:** Multiple-voting should not be possible! (use ZKPs!)
- ✓ **Accessibility:** Anyone can vote (just need a device with internet connection!)



helios Voting: Voter Integrity

Trust the vote.

What if a malicious voter encrypts a value different from 0/1?

A **malicious voter** can encrypt g^{1000} for **one candidate** (multiple votes by same voter!)
and g^{-999} for **another candidate**, thus not impacting total vote count!

∴ Each voter proves that each ciphertext sent is either an encryption of 0 or of 1 (valid vote per candidate)
and the product of all these ciphertexts is an encryption of an integer lying in the prescribed range
(valid number of answers have been selected, if multiple answers allowed in voting!)

Prove:

$CT_b = (Y = g^y, C_b = (X^y \cdot g^b))$ is either an encryption of 0 or an encryption of 1



Prove:

$((X, Y, C_0), y) \in R_{DH}$

or

$((X, Y, C_1/g), y) \in R_{DH}$

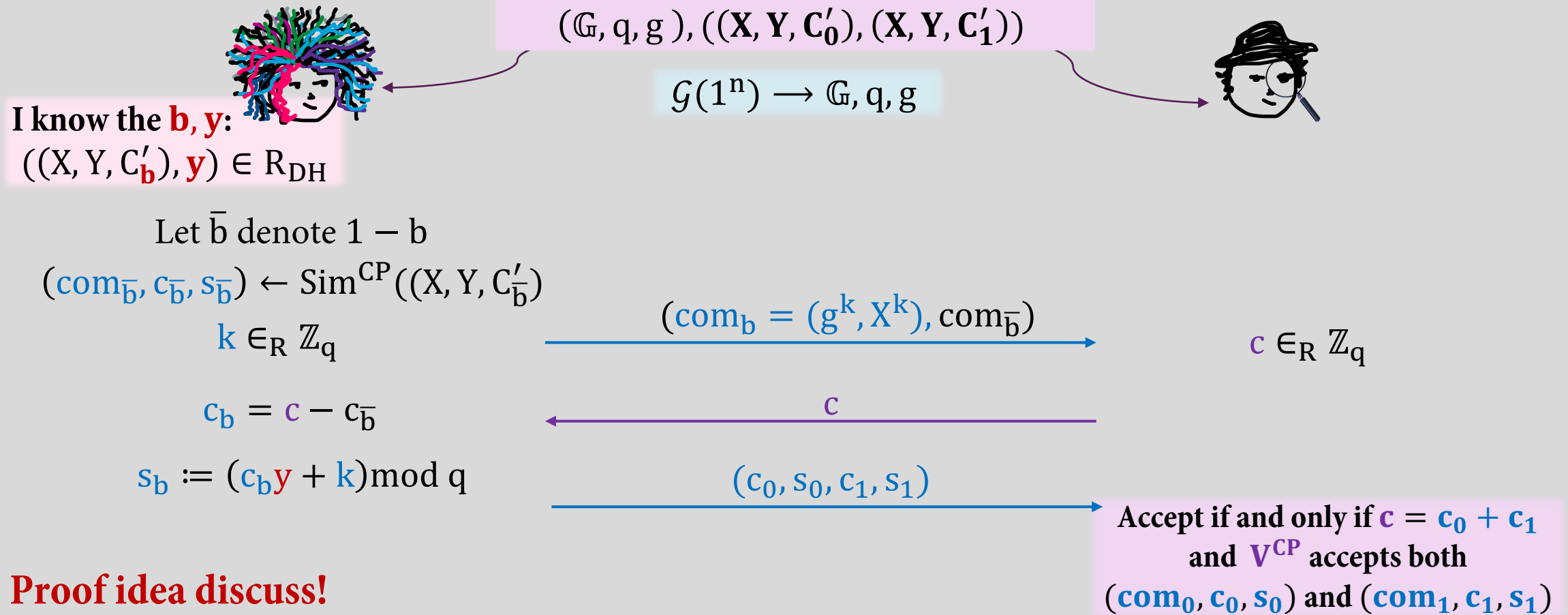
$$Y = g^y, C_0 = X^y$$

$$Y = g^y, C_1/g = X^y$$

Disjunction of Chaum-Pedersen [Cramer, Damgård, Schoenmakers, 94]

helios Voting: Voter Integrity

Trust the vote.



Proof idea discuss!

Disjunction of Chaum-Pedersen [Cramer, Damgård, Schoenmakers, 94]

$$((X, Y, C'_0), \mathbf{y}) \in R_{DH} \quad \text{or} \quad ((X, Y, C'_1), \mathbf{y}) \in R_{DH}$$

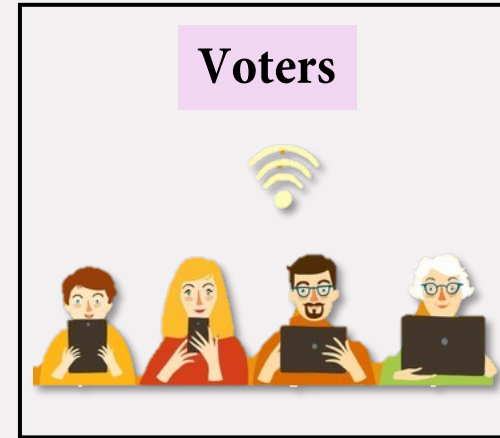
E-Voting

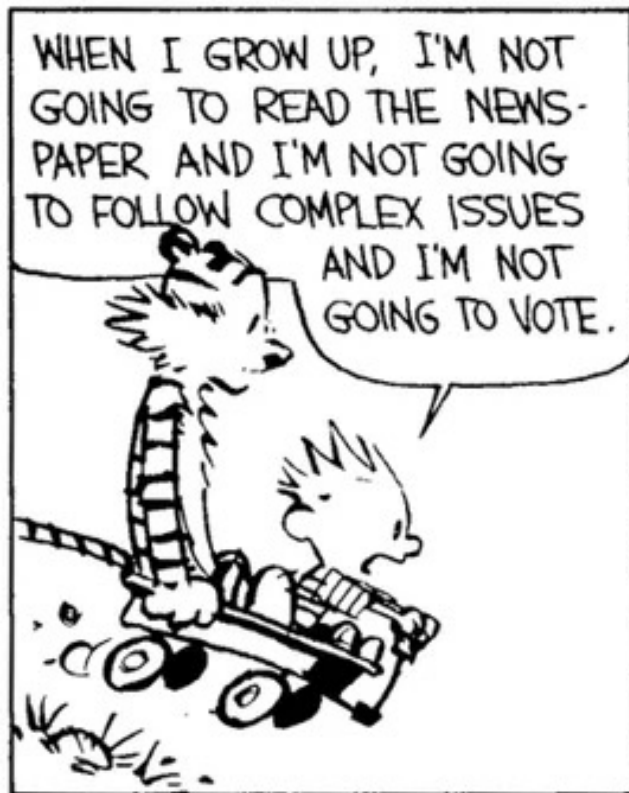
Setting

- ✓ **Registration:** managed by administrator (invitation by email)
- ✓ **Tallying by trustees:** all of them need to be corrupt to cheat!

Requirements

- ✓ **Ballot Privacy:** Ballot should not reveal details about the voter! (**threshold homomorphic PKEs + Schnorr**)
- **Verifiability:**
 - ✓ 1. **Open auditability:** Voter should be able to verify (privately) that his vote is counted. (**ballots posted on bulletin board**)
 - ✓ 2. **End-to-end Verifiability:** Verify (publicly) that tallying is correct! (**Chaum-Pedersen**)
- ✓ **Integrity/Unique Vote:** Multiple-voting should not be possible! (**Disjunction of Chaum-Pedersen**)
- ✓ **Accessibility:** Anyone can vote (**just need a device with internet connection!**)





THANK YOU