Electronic Voting Systems

GUIDE:

PROF. R.K.SHYAMASUNDAR

K. NAGARAJU

M. BHARGAV SRI VENKATESH

Motivation

- There is a need to update voting technologies to improve trust, reliability and convenience.
- For example, the people counting the votes were corrupted and published the wrong number of votes for a party.
- With the correct use of cryptography these issues can be eliminated, which is a great advantage for remote electronic voting systems.
- To improve the current voting systems, we need to study the requirements of voting systems and find ways to fulfil them.

Our work

- Studied Civitas, an electronic voting system and verified its security

- Listed out the various requirements of voting systems.

General voting scenario

- Voters have ids (or public keys) to identify themselves to the system.
- When vote is submitted, voters may have to trust the system, or they may get a receipt for future verification or may receive a proof at the end of election.
- Tabulators count the votes and post the final tally, in some systems they also post proofs to prove their honesty.
- Election officials, supervisor who starts and ends the election, maintains electoral roll and identities of other officials. Registrar who authorises voters.

Civitas – Agents and Phases

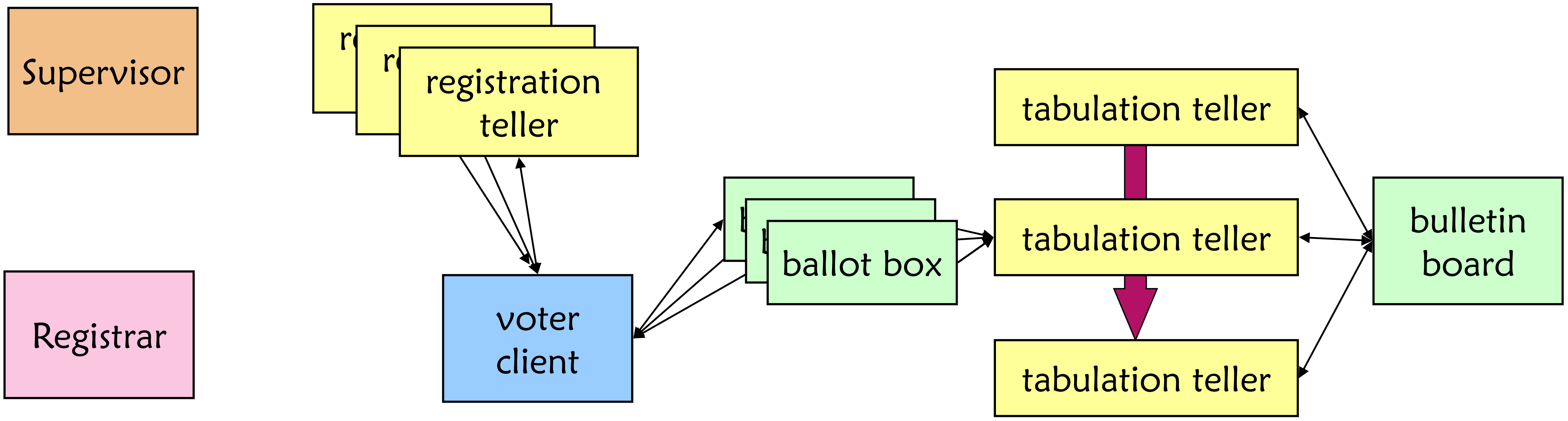
Agents

- Supervisor
- Registrar
- Registration Teller
- Tabulation Teller
- Log service

Phases

- Setup
- Registration
- Voting
- Tabulation
- Verifying an election

Civitas - Architecture



What happens in setup phase?

- **Supervisor** identifies the tellers by posting their individual public keys.
- **Registrar** posts the electoral roll, containing identifiers (names or registration numbers) for all authorized voters, along with the voters' public keys.
- **Tabulation tellers** collectively generate a public key for a distributed encryption scheme and post it on the bulletin board.
- **Registration tellers** generate private credentials, which are used to authenticate votes anonymously and each registration teller stores a share of each private credential. And post the public credentials on the Bulletin Board

Why is setup phase secure?

- Supervisor posts only the public credentials of the parties involved , and the design of the election, which is also a public knowledge.
  - Hence, no breach can happen here.
- The Collective generation of the distributed key generation can be compromised only if all of the tabulation tellers are corrupt.
  - We assume existence of a honest tabulation teller.

El Gamal Key Generation

- Input: El Gamal parameters (p, q, g)
- Output: Public key y, private key x
  - 1. $x \leftarrow \mathbb{Z}_q^*$
  - 2. $y = g^x$ mod p
  - 3. Output (y, x)

El Gamal Encryption

- Input: Public key y, message $m \in \mathbb{Z}_q^*$
- Output: Enc$(m; y)$
  - 1. $r \leftarrow \mathbb{Z}_q^*$
  - 2. Output $(g^r$ mod p, my^r mod p)

El Gamal Decryption

- Input: Private key x, cipher text c = (a; b)
- Output: Dec(c; x)
    - 1. M = b/a^x mod p
    - 2. Output M

Distributed El Gamal Key Generation

- Public input: Parameters (p, q, g)
- Output: Public key Y, public key shares y_i, private key shares x_i
  - 1. S_i: $x_i \leftarrow \mathbb{Z}_q^*$, $y_i = g^{x_i}$ mod p
  - 2. S_i: Publish Commit(y_i) (hash can be used as commitment)
  - 3. S_i: Barrier: wait until all commitments are available
  - 4. S_i: Publish y_i and proof KnowDlog(g, y_i)
  - 5. S_i: Verify all commitments and proofs
  - 6. Y = $\prod_i y_i$ mod p is the distributed public key
  - 7. X = $\sum_i x_i$ mod q is the distributed private key

KnowDlog

- **Principals**: Prover P and Verifier V
- **Public input**: h, v
- **Private input (P)**: x such that $v = h^x$ (mod p)
- 1. P: Compute:
    - z ← $\mathbb{Z}_q^*$
    - a = h^z mod p
    - c = hash(v, a) mod q
    - r = (z + cx) mod q
- 2. P → V : a, c, r
- 3. V : Verify $h^r = av^c$ (mod p).

Distributed El Gamal Decryption

- Public input: Cipher text c = (a, b), public key shares y_i
- Private input (S_i): Private key share x_i
  - 1. S_i: Publish a_i = a^{x_i} mod p and proof EqDlogs(g, a, y_i, a_i)
  - 2. S_i: Verify all proofs
  - 3. A = $\prod_i a_i$ mod p
  - 4. M = b/A mod p
  - 5. Output M.

EqDlogs

- **Public input:** f, h, v, w
- **Private input (P):** x such that $v = f^x$ (mod p) and $w = h^x$ (mod p)
- 1. P: Compute:
  - z ← $\mathbb{Z}_q^*$
  - a = f^z mod p
  - b = h^z mod p
  - c = hash(v, w, a, b) mod q
  - r = (z + cx) mod q
- 2. P → V : a, b, c, r
- 3. V : Verify $f^r = a\, v^c$ (mod p) and $h^r = bw^c$ (mod p).

Why is setup phase secure?

- So, the Collective generation of the distributed key is provably secure which involves zero knowledge proofs discussed above.

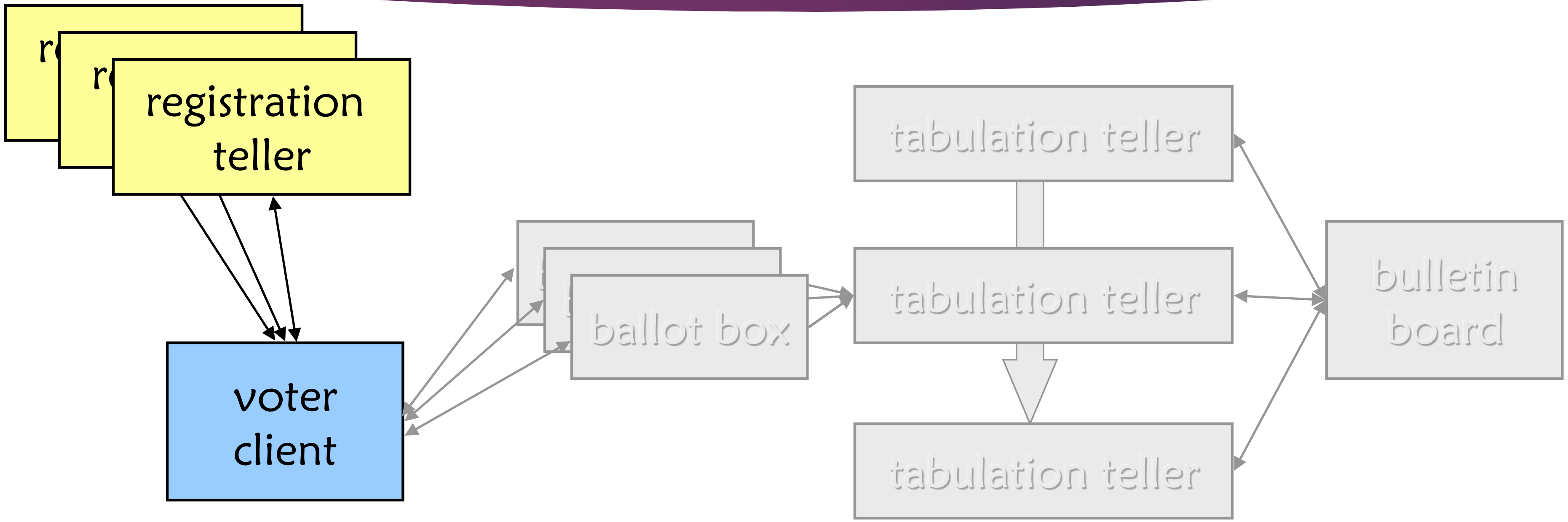
- To decrypt any message which is encrypted using K_{TT}, needs participation of every tabulation teller.

Why is setup phase secure?

- The Registration tellers post the public credential of the voter. For the private credential to get leaked all the registration tellers have to collude.

- We also assumes existence of a honest registration teller. Hence, this phase is secure under our trust assumptions.

Registration



Voter retrieves *credential share* from each registration teller;
combines to form *credential*

What happens in registration phase?

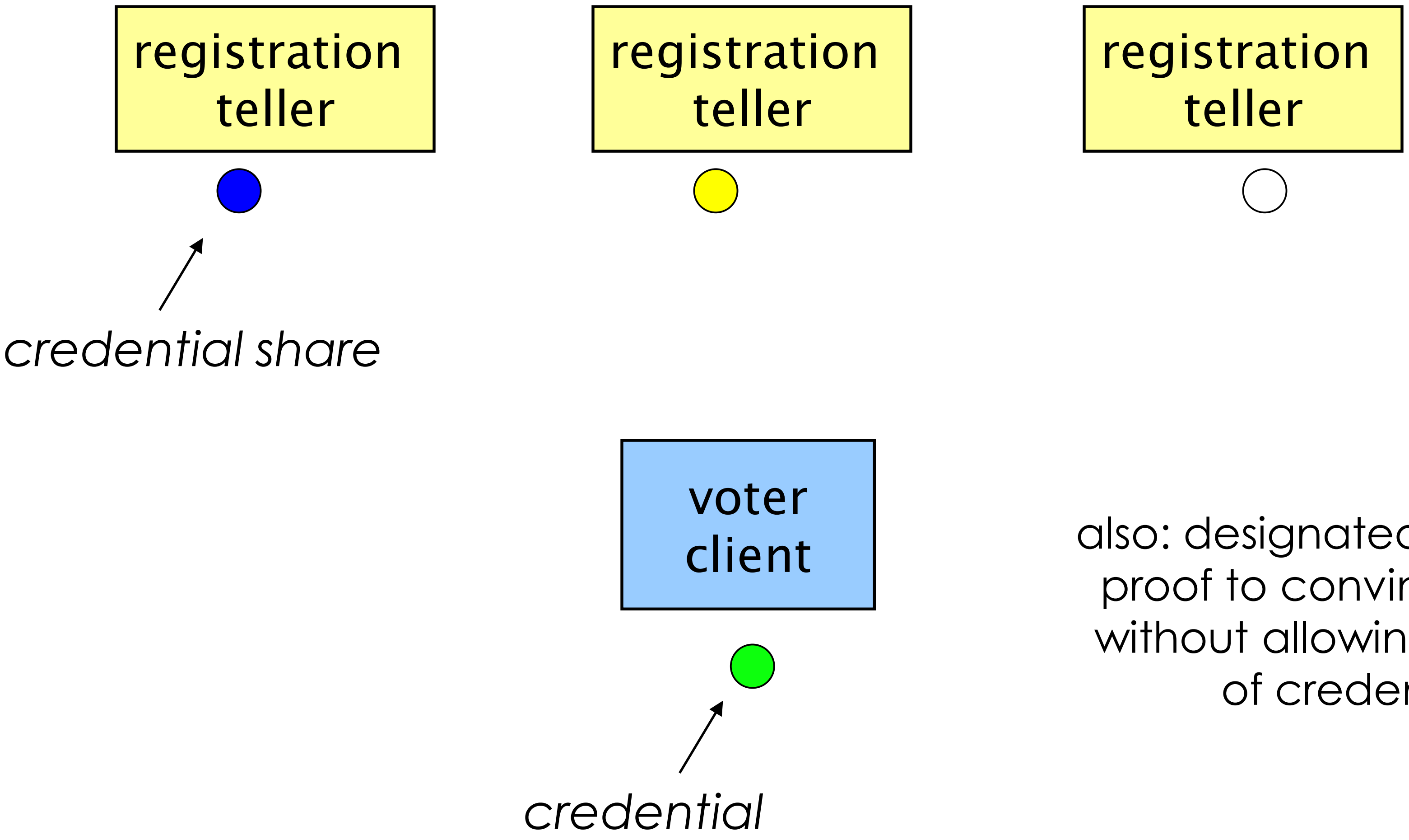
- The voter acquires his part of private credential from each of the registration teller.

- Voter: Authenticates using his registration key, obtains an AES session key using Needham-Schroeder-Lowe protocol. Using this key the Reg.Teller send a message (s,r,S',D).

  - S' = Enc{(s,r), K_{TT}}

  - D is the DVRP proof showing S' is re encryption of S

Registration Protocol



also: designated-verifier ZK proof to convince voter without allowing transfer of credential

Why is registration phase secure?

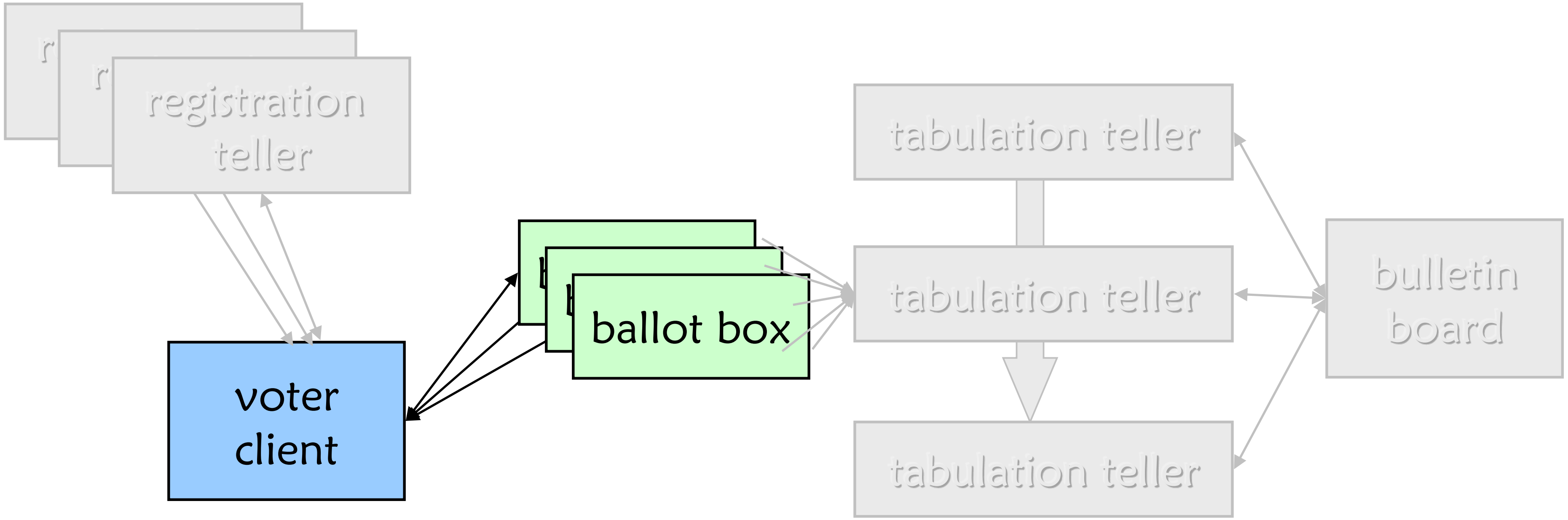
- The above phase is secure under the assumptions that:
  - The Needham-Schroeder-Lowe protocol is secure
  - DVRP is correct.
  - All Reg Tellers are not corrupt

- Under the above assumptions there is no way for any adversary to get to know the private credential of the voter

Drawback

- Procedure mentioned to give fake private credential to an adversary is not efficient.
- The given method assumes that voter knows an honest registration teller and can fake that particular share, thereby giving adversary incorrect private credential.
- But, this assumption of voter knowing which teller is honest, is not practical.

Voting



Voter submits copy of encrypted *choice* and credential
(+ ZK proofs) to each ballot box

What happens in voting phase?

- In the voting phase the voter sends a message to Ballot Box

  < Encr(s, K_{TT}) , Encr(v, K_{TT}), P_w, P_k>

  - s - voter's private credential
  - v - voter's choice for election
  - P_w - Zero knowledge proof to show vote is well formed
  - P_k - Zero knowledge proof to show voter knows s and v simultaneously

Vote Proof

- **Public input:**

Encrypted credential (a_1, b_1)

Encrypted choice (a_2, b_2)

Let E = (g, a_1, b_1, a_2, b_2)

- **Private input (P):**

$\propto_1, \propto_2$ such that $a_i = g^{\propto_i}$ (mod p)

Vote Proof

- P: Compute:
  - $r_1, r_2 \leftarrow Z_q$
  - $c = hash(E, g^{r_1} mod\ p, g^{r_2} mod\ p) \bmod q$
  - $s_1 = (r_1 - c \propto_1) \bmod q$
  - $s_2 = (r_2 - c \propto_2) \bmod q$
- $P \rightarrow V:\ \ c, s_1, s_2$
- $V : Compute\ c' = \mathrm{hash}(\mathrm{E}, g^{s_1} a^{c}{}_1, g^{s_2} a^{c}{}_2)\ mod\ q$
- $V\!: Verify\ c = c'$

Why is voting phase secure?

- Identity of voter cannot be known, as public credential in the vote is a re-encryption of the private credential and its equivalence to the public credential posted on bulletin board can only be revealed, if all the tabulation tellers collude.

- Voter's choice also cannot be revealed unless all the tabulation tellers collude as it was encrypted using distributed elgamal encryption scheme. No one can modify the vote as changing Pk requires knowledge of s and v simultaneously.

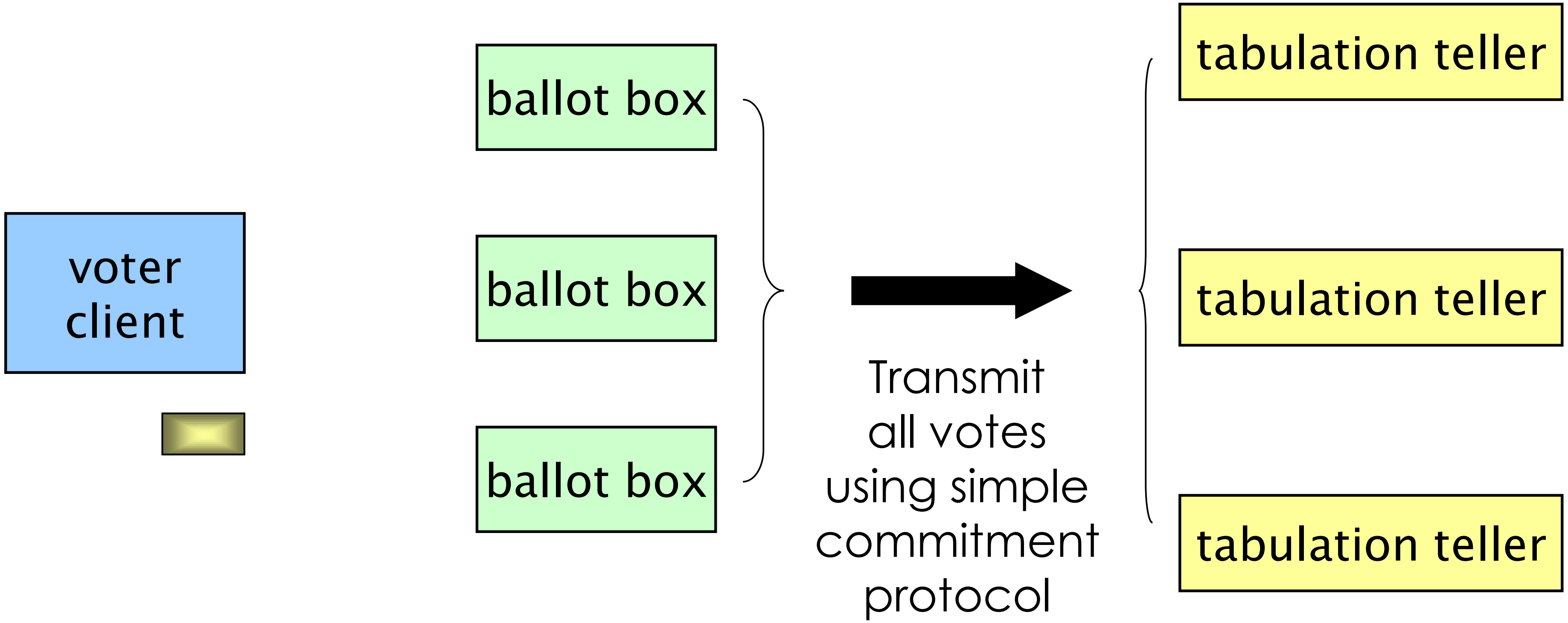
PET (Plaintext Equivalence Test)

- Public input: c_j = Enc(m_j, K_{TT}) = $(a_j,\, b_j)$ for j = 1,2
- Private input (TT_i): Private key share x_i
- Let R = (d; e) = $(a_1/a_2;\, b_1/b_2)$
  - 1. TT_i: $z_i \leftarrow Z_q$; (d_i, e_i) = (d^{z_i}, e^{z_i})
  - 2. TT_i: Publish Commit (d_i, e_i)
  - 3. TT_i: Barrier: wait until all commitments are available
  - 4. TT_i: Publish (d_i, e_i) and proof EqDlogs (d, e, d_i, e_i)

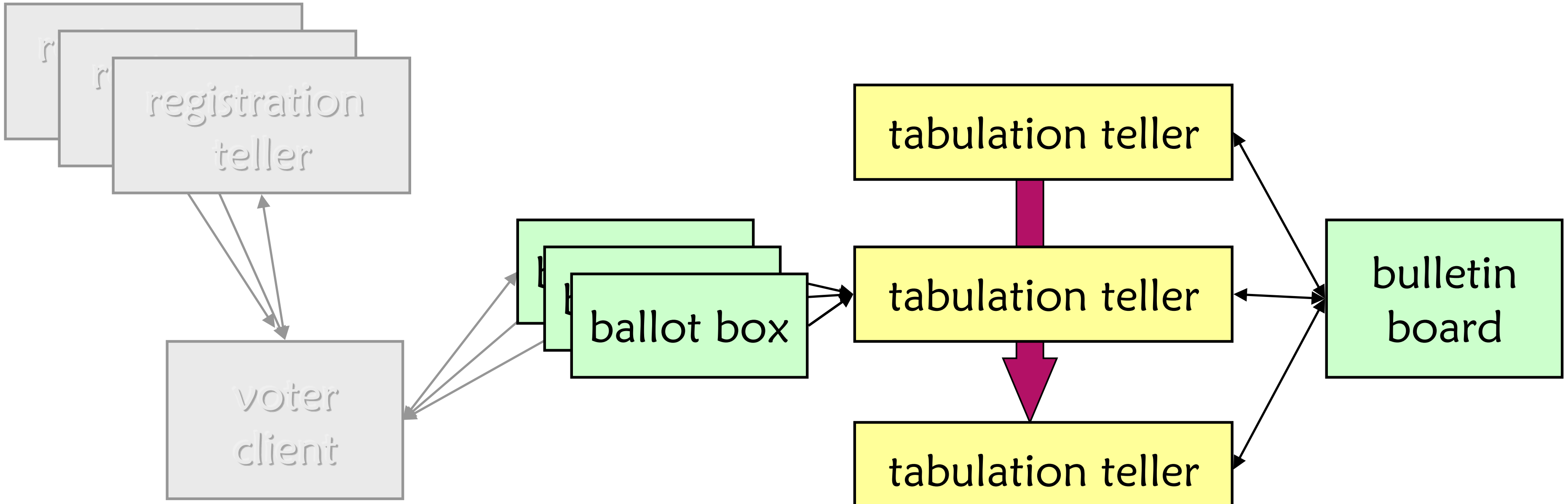
PET (Plaintext Equivalence Test)

- 5. TT_i: Verify all commitments and proofs
- 6. Let c' = $(\prod_i d_i, \prod_i e_i)$
- 7. All TT_i: Compute m' = DistDec(c') using private key shares
- 8. If m' = 1 then output 1 else output 0

Tabulation


voter client
ballot box
ballot box
ballot box
Transmit all votes using simple commitment protocol
tabulation teller
tabulation teller
tabulation teller

Tabulation



Tellers retrieve votes from ballot boxes

What happens in tabulation phase?

All Tabulation Tellers: Proceed sequentially through the following phases. Each phase has a list (e.g., A, B, etc.) as output. In each phase that uses such a list as input, verify that all other tellers are using the same list.

- **Retrieve Votes**

Retrieve all votes from all Ballot Boxes. Verify the commitments. Let the list of votes be A.

This step is secure since ballot boxes are instances of an insert-only log service. So, ballot boxes cannot modify the submitted votes

What happens in tabulation phase?

- **Check Proofs**

Verify all P_w and P_ks in retrieved votes. Eliminate any votes with an invalid proof. Let the resulting list be B. Assuming at least one honest tabulation teller implies that list B is correctly computed.

- **Duplicate Elimination**

Run PET(S_i, S_j) for all i < j, where S_x is the encrypted credential in vote. Eliminate any votes for which the PET returns 1 according to a re-voting policy. (PET- Plaintext Equivalence Test)

Let the remaining votes be C

What happens in tabulation phase?

- **Mix Votes**

 Run MixNet(C) and let the anonymized vote list be D.

- **Mix Credentials**

Retrieve all credentials from Bulletin Board and let this list be E. Run MixNet(E) and let the anonymized credential list be F.

 Here we are assuming security property of mix nets. (Modification of large number of votes without getting caught can only happen with negligible probability)

What happens in tabulation phase?

- **Invalid Elimination**. Run PET(S_i, T_j) where S_i = F[i] and T_j = D[j]. Eliminate any votes (from D) for which the PET returns 0.
  Let the remaining votes be G.

As credentials are passed through mix net, identity of voter cannot be known with PET, since it requires cooperation of all TTs

- **Decrypt**. Run Distributed Decryption on all encrypted choices in G. Output the decryptions as H, the votes to be tallied.

Only choices are decrypted, so identity of voter is not revealed

What happens in tabulation phase?

- **Tally**. Compute tally of H. Verify tally from all other tellers.

As existence of an honest tabulation teller is assumed, tally cannot be incorrect. Hence, tabulation phase is secure.

How is universal verifiability provided?

- Tabulation is made publicly verifiable by requiring each tabulation teller to post proofs that it is honestly following the protocols (in mix nets).
- All tabulation tellers verify these proofs as tabulation proceeds. An honest teller refuses to continue when it discovers an invalid proof.
- Anyone can verify these proofs during and after tabulation, yielding universal verifiability.

How is voter verifiability provided?

- Final list of mixed credentials is decrypted, revealing the identity of all the voters whose vote has been included in the final tally.

- But, corresponding choice is not known, because choices and public credentials are mixed using different mixnets.

Requirements of Voting Schemes

Anonymity and Election Secrecy

- No one can know the choice of a voter in the ballot he submitted.
- More rigorous, No one can know whether a given voter has voted or not

Eligibility

- Only voters who have their identity listed in electoral roll can participate in election
- An attacker could try to vote without being authorized, which obviously should not be allowed.
- An authorized voter could also try to vote multiple times in a way where all votes are counted. This should obviously be recognized and not allowed.

Voter Verifiable

- Each voter can check that their own vote is included in the tally.

- Simple method is to concatenate a random number to your vote, and all the votes are displayed at the end of election.

Universal Verifiability

- The final tally is verifiably correct.

- Anyone can check that all votes cast are counted, that only authorized votes are counted, and that no votes are changed during counting.

Resist coercion

- Voters cannot prove whether or how they voted, even if they can interact with the adversary while voting.

- Simply put, voter should not be able to sell his vote.

Hiding Interim Results

- Partial results should not be released during the voting period.
- This preserves privacy of the voter.

Availability

- A voting system must remain available during the whole election and must serve voters connecting from untrusted clients.

Indian EVM

- The current voting system in India uses Electronic Voting Machines.
- The system uses machines provided by the government to register votes.
- The voters have to go to the polling station to cast the vote.
- At the polling station the identification of the voter is done, and then the voter is allowed to vote on the allotted EVM.

Drawbacks

- The drawbacks of the current system are :
  - Voter has to place trust on the machines and the polling booth in general that his vote is being recorded correctly
  - Voters have no way to verify if their vote has been counted correctly in the final tally

Suggestions

- We would like to suggest the use of cryptographic tools to try to overcome some of the drawbacks.
- Every vote is associated with a random number, which is saved in the EVM as well as given to the voter.
- Now in the final tally the vote as well as the number together are displayed.
- The voter can now easily verify if his vote has been tallied or not. But this is not coercion resistant.

Thank you