# JCJ and CIVITAS

# Coercion Resistance

- A stronger adversary
- Active attacks
  - Vote for a specific candidate
  - Vote randomly
  - Completely abstain from voting
  - Yield private keys – allow simulation
  - Monitor voting systems
- The essential security property for Internet voting
- **Note:** Coercion Resistance ⇒ Receipt freeness

# The JCJ coercion resistance framework

- Intuition:
  - The adversary will not coerce, if they cannot verify that the coercion attempt will succeed
- Techniques
  - Each voter can vote multiple times
  - The voter can generate and register credentials (=random group elements)
  - There is a single valid credential for each voter (=the one registered)
  - During voting the voter may generate indistinguishable credentials through a device or some other manner
  - All the votes accompanied with other credentials are considered fake and should not be counted

A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES, 2005

# JCJ Assumptions

- Each voter has a moment of privacy where they can cast their real ballot
  - May occur before / after the adversarial attack
- The casting phase is anonymous
  - Otherwise, the forced abstention attack would always succeed
- The coercer is uncertain about the *behavior of all the voters*
  - If everyone else votes, then the abstention attack will always succeed
  - If nobody votes for the candidate the coercer demands then the attack will succeed
  - Insertion of dummy votes
- Untappable registration
  - Or the coercer becomes the voter

# JCJ Workflow

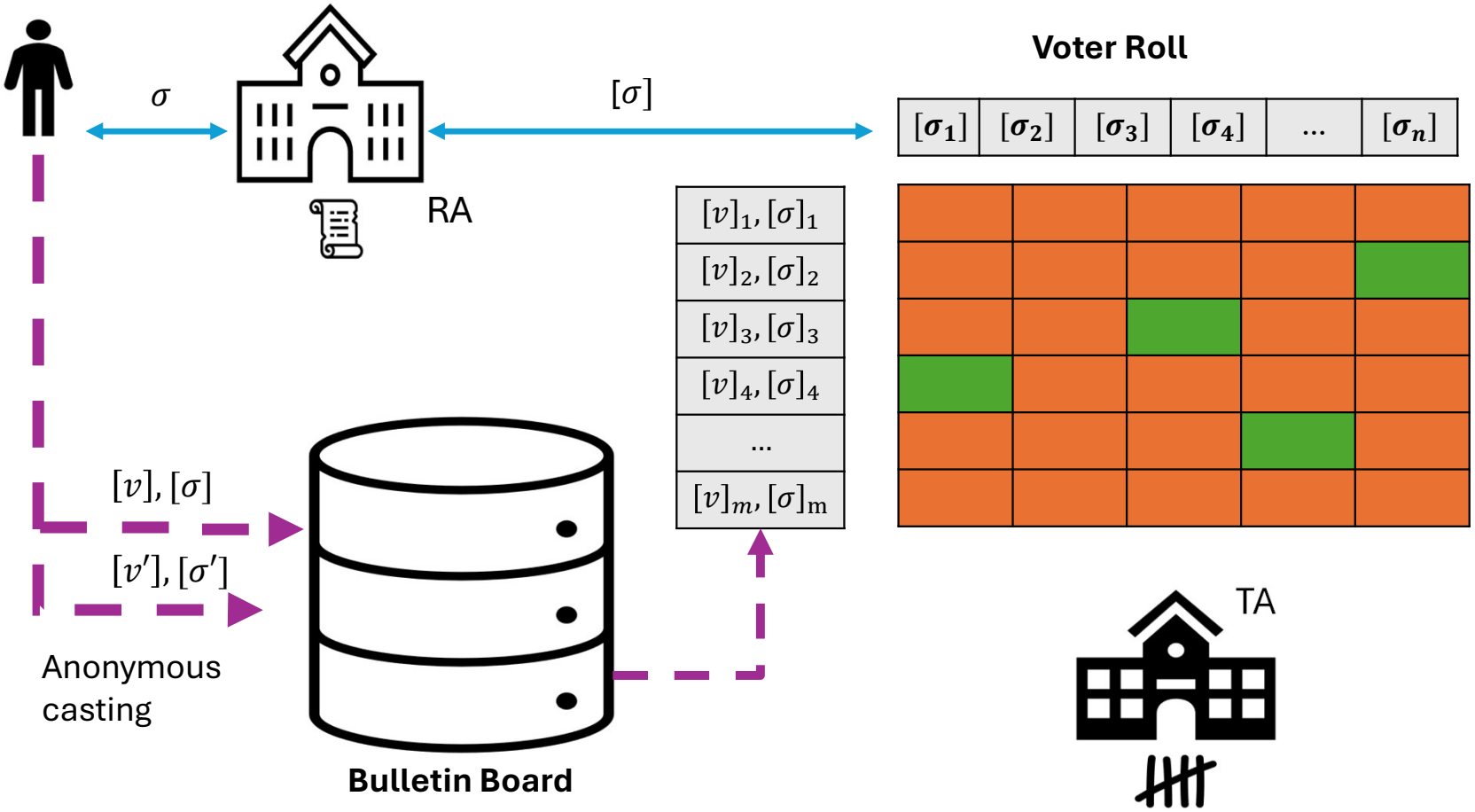
- The voter registers their real credential
  - Untappable registration – occurs once but may be reused
  - The voter may create the credential either alone or together with an authority
- The authorities publishes all real credentials in encrypted form
  - Voter roll
- Coercion Attack
  - The voter generates a fake but indistinguishable credential
  - The voter complies with the commands of the coercer
  - The coercer may monitor the voter afterwards, except during…
- Moment of privacy
  - The voter casts their vote of choice accompanied with their real credential

# JCJ Workflow (2)

- Tallying
  - The BB is anonymized
  - Ballot weeding
  - The authorities disregard **in a verifiable manner**:
    - all duplicate ballots (e.g. by keeping only the last ballot per voter)
    - all ballots with fake credentials
  - How: Blind credential comparisons using PET
    - Between all ballots
    - Between unique ballots and the voter roll
    - Proof of PET serves verifiability

# The scheme

M. R. Clarkson, S. Chong, and A. C. Myers, “Civitas: Toward a Secure Voting System,” in S&P’08. IEEE, 2008.



# JCJ Discussion

- **Quadratic tallying time**
  - $O(m^2)$: To keep one ballot per credential
  - $O(mn)$: To filter out ballots with fake credentials
  - Goal: $O(m + n)$
  - **Solutions:**
    - Blinded hashing
    - Anonymity sets
    - Structured credentials

- **Difficult to use by the voters**
  - Need for hardware tokens that generate fake credentials
  - Solutions:
    - Panic passwords
    - Except for valid, invalid a password can signal coercion
    - Panic password list per voter
    - Moment of privacy:
      - Valid password
    - Coercion:
      - Panic password

# Solution 1: Blinded hashing

- The RA members jointly compute a blinding value $z$
- The encrypted credential $[\sigma]$ is blinded to $[\sigma]^z = [\sigma^z]$
- Instead of using PET the blinded credential is decrypted to obtain $\sigma^z$ (credential fingerprint)
- The pair $(\sigma^z, b)$ is run through a hashtable
- Duplicate detection is $O(1)$
- Vulnerable to the tagging attack!

Warren D. Smith. "New cryptographic voting scheme with best-known theoretical properties". In: *Frontiers in Electronic Elections (FEE 2005)*. 2005.

# Tagging attack during blinded hashing

- The coercer forces the voter to reveal the fake credential
  - The voter complies as this does not happen during the moment of privacy
- Then the coercer posts two ballots one with $[\sigma]$ and one with $[\sigma^2]$
- Two fingerprints will be produced $\sigma^z$ and $\sigma^{2z}$
- The coercer then squares all elements in the BB.
  - If a square matches another element and **both** have been discarded then the adversary learns that $\sigma$ was fake
- **Note:** This attack applies only to fake credentials, not duplicate checks
- So at least: $O(m^2)\ becomes\ O(m)$

# Solution 2: Anonymity sets

- Each ballot contains:
  - The current credential (real or fake)
  - The real credential from the voter roll (rerandomized)
  - Some $\beta - 1$ randomly selected credentials from the voter roll (rerandomized)
- The PET takes places only between the credentials of the ballot
  - And only to detect fake credentials
  - $O(n\beta)$
  - Anonymity set size $\beta$ may be adjusted for performance

J. Clark and U. Hengartner, "Selections: Internet Voting with Over-the Shoulder Coercion-Resistance," in FC'11. Springer, 2011.

# Solution 3: Structured credentials

- The checks for validity and duplicates are embedded in the credential
- For instance:
  - The $RA$ has secret keys $x, y \in \mathbb{Z}_q$ shared with the $TA$
  - A credential is a tuple $(r, A, B = A^y, C = A^{x+rxy})$ where $r \in \mathbb{Z}_q, A \in \mathbb{G}$
  - $r$ should be kept secret by the voter, $A, B, C$ can be public
- During coercion the voter generates a fake $r$
- The voter cannot prove to anyone else that $(r, A, B, C)$ is real or fake.
- Easy to generate new credentials for other elections / revoting
  - If $(r, A, B, C)$ is a credential then $(r, A,^l B,^l C^l)$ is also a credential
- The ballot is $([v], [A], [A^r], [B^r], [C], O^r, \pi)$ where $O \in \mathbb{G}$
  - $O^r$ is used as a tag for duplicate removal
  - Check if $PET([A^r]^y, [B^r]) = 1$
  - Ballot validity if $(C \cdot A^{-x} \cdot (B^r)^{-x})^z = 1$ for some $z \in \mathbb{Z}_q$

Roberto Araújo, Sébastien Foulle, and Jacques Traoré. "A practical and secure coercion resistant scheme for remote elections". In: *Frontiers of Electronic Voting*. 2007

# Formally Defining Security Properties

and proving them for Helios

# Verifiability

- Trust Assumptions
  - The adversary fully controls the relevant system components
    - Universal verifiability:
      - The TA is fully corrupted
    - Eligibility verifiability:
      - The RA is fully corrupted
    - In some models the BB is corrupted, in others it is not
  - The adversary also controls a subset of the voters

# Individual verifiability

- The voters verify that their ballots are correctly included in the tally
  - Recorded as cast!
- A necessary condition:
  - All ballots are unique
- The primary attack:
  - Clash attack: Two voters are tricked to verify the same ballot
  - The $EA$ is free to substitute the second one with its own
- Remark:
  - Paper-based voting does not possess individual verifiability!

# Individual verifiability definition

$\mathbf{Ind - Ver_{VS,A}}$

$pp \leftarrow VS.Setup(1^{\lambda})$
$pk_{EA}, BB \leftarrow VS.SetupElection(pp)$
$(CS, v_0, v_1) \leftarrow A(pp, pk_{EA})$
$b_0 \leftarrow VS.Vote\langle A(\,), V(v_0), pk_{EA}, BB, CS)$
$b_1 \leftarrow VS.Vote\langle A(\,), V(v_1), pk_{EA}, BB, CS)$
$if\ b_0 = b_1\ and\ b_0 \neq \perp\ then$
    $return\ 1$
$return\ 0$

A voting scheme $VS$ satisfies individual verifiability if $\forall\ PPT\ A$:
$\Pr[\mathbf{Ind - Ver_{VS,A}}\ (1^{\lambda}) = 1] \leq negl(\lambda)$

**Notes:**
Voter intent is not taken into account (cast as intended verifiability)
Even if it did, could there be a negligible probability of success?



Steve Kremer, Mark Ryan, and Ben Smyth. "Election Verifiability in Electronic Voting Protocols". In: *ESORICS 2010*

# Clash attacks on Helios

- The use of aliases greatly affects the adversarial capability of mounting clash attacks
- Helios without aliases
  - Clash attacks occur with negligible probability in ElGamal encryption
- Helios with aliases
  - The EA assigns the same alias to 2 voters that have increased probability to vote for the same candidate
  - Points them to the same ballot – Both verify it correctly!
  - Casts a ballot for its preferred candidate in the free slot.
- The voter contributes to the randomness required to encrypt ballots

# Countermeasures for clash attacks

- The BB is always up-to date and updated after each ballot
- Voters observe the BB before casting
- Voters check audited ballots for exact duplicates
- Voters contribute to the encryption randomness
  - real-world by typing a random phrase
- Use external authentication or real-world identities as aliases
  - This might leak abstention

# Universal verifiability

- Every interested party can verify that the tally corresponds to the ballots in the $BB$
- Adversarial goal: Present a tally along with fabricated evidence that passes verification
- Baseline: A function that correctly computes the tally given $only$ the plaintext votes
  - $result(pk_{EA}, BB, CS)[v] = n_v \Leftrightarrow \exists^{n_v} b \in BB : b = Vote(v)$

# Universal Verifiability – A first definition

$\mathbf{Uni-Ver_{VS,A}}$

$pp \leftarrow VS.Setup(1^{\lambda})$
$pk_{EA}, BB \leftarrow VS.SetupElection(pp)$
$(CS, BB', T_A, \pi_{T_A}) \leftarrow A^{VS.Vote}(pp, pk_{EA})$
$T \leftarrow result(\{v_i | b_i \in BB\})$
$if\ T \neq T_A\ and\ VS.Verify(CS, BB', T_A, \pi_{T_A}) = 1\ then$
    $return\ 1$
$return\ 0$

A voting scheme $VS$ satisfies universal verifiability if $\forall\ PPT\ A$:
$\Pr[\mathbf{Uni-Ver_{VS,A}}(1^{\lambda}) = 1] \leq negl(\lambda)$



Steve Kremer, Mark Ryan, and Ben Smyth. "Election Verifiability in Electronic Voting Protocols". In: *ESORICS 2010*

# Additional considerations

- Is the BB passive (simply stores ballots)?
- Does the BB contain public voter identities?
  - If yes, who maintains them? (e.g. a registration authority)
  - If not, is the BB passive?
  - If not, is the BB honest?
    - Does it add / remove / alter ballots?
- Do all voters indeed verify their ballots?
  - In real-life elections this is not the case
  - **Too strong definition**

**Helios does not specify these!!!**



Cortier, V., Galindo, D., Glondu, S., Izabachène, M. (2014). Election Verifiability for Helios under Weaker Trust Assumptions. ESORICS 2014

# Election verifiability – Intuition

- A voting scheme is verifiable if the result corresponds to the votes of:
  - All honest voters that have verified their ballot
  - All the valid votes of the corrupt voters (**at most** – no ballot stuffing)
  - Some of the honest voters that have not checked their ballots

# Election verifiability –adversarial objective

- Cause a tally to be accepted if:
  - Ballot stuffing occurs
    - $\#corrupted\_votes > \#corrupted\_voters$
  - Verification was partially bypassed
    - Some of the votes of the honest voters **that verified their ballots** are **not** included in the tally
  - Some of the votes of the honest voters **that did not check** are **not** included in the tally

# Election verifiability – A finer grained approach

- Include two new authorities to model their impact on the voting system
  - Registration authority
    - Create and distribute voter credentials (maybe offline)
    - Vote algorithm should include them
  - Bulletin' Board authority
    - Authenticate the voters and maintain the $BB$
    - Not passive: Add or remove ballots
- **Weak** election verifiability
  - Both the BB and the RA are honest
- **Strong** election verifiability
  - The BB and the RA are **not both** corrupt.

# Election verifiability – Formal definition

- Helper oracles

$\boldsymbol{ORegister(i)}$

$(sk_i, pk_i) \leftarrow VS.Register(RA(sk_{RA}), V_i())$
$V_{EL} \Leftarrow (i, pk_i)$

$\boldsymbol{OVote(i, v_i)}$ – **Honest votes**

$if\ i \in V_{EL}\ \ AND\ i \notin V_{CR}\ then$
$\quad b_i \leftarrow VS.Vote(i, pk_i, sk_i, v_i)$
$\quad V_H \Leftarrow V_H \cup \{i, v_i, b_i\}$

$\boldsymbol{OCorrupt(i)}$

$if\ V_i \in V_{EL}\ then$
$\quad V_{CR} \Leftarrow (i, sk_i, pk_i)$

$\boldsymbol{OCast(i, b)}$ – **Honest BB**

$BB \Leftarrow (i, b)$

# Weak verifiability

$\mathbf{Weak - Ver_{VS,A}}$

$pp \leftarrow VS.Setup(1^{\lambda})$
$pk_{EA}, sk_{EA}, BB \leftarrow VS.SetupElection(pp)$
$(CS, T_A, \pi_{T_A}) \leftarrow A^{ORegister, OCorrupt, OVote, OCast}(pp, pk_{EA})$
$if\ T_A = \perp\ \mathbf{or}\ VS.Verify(CS, BB, T_A, \pi_{T_A}) = 0\ then\ return\ 0$
$if\ \ \exists \{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}\ where\ 0 \leq n_{V_{CR}} \leq |V_{CR}|$
$\quad \mathbf{and}\ T_A = result\left(\{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}\right) \oplus result\left(\{v_i \mid i \in V_H\}_{i=1}^{n_{V_H}}\right)$
$\quad return\ 0$
$return\ 1$

A voting scheme $VS$ satisfies weak verifiability if $\forall\ PPT\ A$:
$\Pr[\mathbf{Weak - Ver_{VS,A}}(1^{\lambda}) = 1] \leq negl(\lambda)$

The adversary:
- controls the TA
- controls some voters $V_{CR}$
- Cannot add or alter ballots (BB is updated only through $OCast$)

The adversary wins if:
- Its result verifies
- Its result cannot be 'explained' by the result of **all** the honest and **some** of the corrupt voters

# Strong verifiability with malicious BB

$\mathbf{Weak - Ver_{VS,A}}$

$pp \leftarrow VS.Setup(1^{\lambda})$

$pk_{EA}, sk_{EA}, BB \leftarrow VS.SetupElection(pp)$

$(\boldsymbol{CS, BB, T_A, \pi_{T_A}}) \leftarrow \boldsymbol{A^{ORegister, OCorrupt, OVote}}\ (\boldsymbol{pp, pk_{EA}})$

$if\ T_A = \perp\ \boldsymbol{or}\ VS.Verify(CS, BB, T_A, \pi_{T_A}) = 0\ then$

$\quad return\ 0$

$V_{CH} \leftarrow \{(i_{CH}, v_{CH}, b_{CH})\}_{i=1}^{n_{CH}}$ ***//voters who performed verification***

$if\ \exists \{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}\ where\ 0 \leq n_{V_{CR}} \leq |V_{CR}|$

$\quad \boldsymbol{and}\ T_A = result\left(\{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}\right) \oplus result\left(\{v_i \mid i \in V_H - V_{CH}\}_{i=1}^{n_{V_H} - n_{CH}}\right) \oplus result\left(\{v_i \mid i \in V_{CH}\}_{i=1}^{n_{CH}}\right)$

$\quad return\ 0$

$return\ 1$

A voting scheme $VS$ satisfies weak verifiability if $\forall\ PPT\ A$:

$\Pr[\mathbf{Strong - Ver_{VS,A}}\ (1^{\lambda}) = 1] \leq negl(\lambda)$

The adversary controls:

- The TA
- Some voters $V_{CR}$
- The BB: can add / alter / remove ballots

The adversary wins if:

- Its result verifies
- Its result cannot be 'explained' by the votes of all the honest who checked, some of the honest voters who did not check (no need to delete them all) and at most all the corrupt voters (some ballots were added)

# Strong verifiability with malicious RA

$\mathbf{Weak - Ver_{VS,A}}$

$pp \leftarrow VS.Setup(1^{\lambda})$

$pk_{EA}, sk_{EA}, BB \leftarrow VS.SetupElection(pp)$

$(\boldsymbol{CS, T_A, \pi_{T_A}}) \leftarrow \boldsymbol{A^{OCorrupt,OVote,OCast}}\ (\boldsymbol{pp, pk_{EA}})$

$if\ T_A = \perp\ \boldsymbol{or}\ VS.Verify(CS, BB, T_A, \pi_{T_A}) = 0\ then$

$\quad return\ 0$

$V_{CH} \leftarrow \{(i_{CH}, v_{CH}, b_{CH})\}_{i=1}^{n_{CH}}$ ***//voters who performed verification***

$if\ \exists \{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}\ where\ 0 \leq n_{V_{CR}} \leq |V_{CR}|$

$\quad \boldsymbol{and}\ T_A = result\left(\{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}\right) \oplus result\left(\{v_i \mid i \in V_H - V_{CH}\}_{i=1}^{n_{V_H} - n_{CH}}\right) \oplus result\left(\{v_i \mid i \in V_{CH}\}_{i=1}^{n_{CH}}\right)$

$\quad return\ 0$

$return\ 1$

A voting scheme $VS$ satisfies weak verifiability if $\forall\ PPT\ A$:

$\Pr[\mathbf{Strong - Ver_{VS,A}}\ (1^{\lambda}) = 1] \leq negl(\lambda)$

The adversary controls:

- The TA
- Some voters $V_{CR}$
- Cannot add / alter / remove ballots directly
- Only through credentials issued by the RA (false users, malicious credentials)

The adversary wins if:

- Its result verifies
- Its result cannot be 'explained' by the result of the honest who checked the honest voters who did not check and some of the corrupt voters

# Proving election verifiability for Helios

- Helios is weakly verifiable
- Strong verifiability does not apply
  - No RA / BB authority in formal specifications
- Belenios (Helios-C) is strongly verifiable

- Helper Notions
  - **Correctness**
    - Honestly generated ballots are always accepted
    - The tally output matches the result output
    - All verifications pass
  - **Partial Counting / Tallying**
    - The result function does not change if it is applied first on some subset of voters and the partial results are then joined
  - **Tally uniqueness**
    - A tally that passes verification is unique
  - **Accuracy**
    - Any ballot that passes verification will be counted (even if it is generated by the adversary)

# Helper Notions

- **Correctness**
  - $$Pr\left[\begin{array}{c}(T,\pi) \leftarrow VS.Tally(\{b_1,\dots,b_n\});\\ b_i = VS.Vote(i,v_i,sk_i), v_i \in CS \; and \; VS.IsValid(b_i) = 1 \; \forall i\\ and \; VS.Verify(T,\pi,\{b_1,\dots,b_n\}) = 1\\ and \; T = result(\{v_1,\dots v_n\})\end{array}\right] = 1$$
- **Partial Counting**
  - $result(V) = result(V_1) \boxplus result(V_2)$ where $V = V_1 \cup V_2$
- **Partial Tallying**
  - $tally(\mathrm{BB}) = tally(BB_1) \oplus tally(BB_2)$
    where $BB = BB_1 \cup BB_2$ and $BB_1 \cap BB_2 = \emptyset$

# Helper Notions

- **Tally Uniqueness**

$$Pr\left[\begin{matrix}(BB, T_1, \pi_1, T_2, \pi_2) \leftarrow A(pp); \\ T_1 \neq T_2 \; and \\ Verify(BB, T_1, \pi_1) = 1 \; and \\ Verify(BB, T_2, \pi_2) = 1\end{matrix}\right] \leq negl(\lambda)$$

- **Accuracy**
  - Every ballot (even if it dishonest) that passes verification corresponds to an admissible vote
    - If $IsValid(b) = 1$ and $Verify(\{b\}, T_b, \pi_b) = 1$ then $v_b \in CS$ and $T_b = result(v_b)$
  - An honestly generated proof passes the verification test (even for adversarially generated BBs)
    - $Verify\big(BB, Tally(BB, sk_{EA})\big) = 1, \forall BB$

# Sufficient Conditions For Weak Verifiability

- **Correctness**
- **Tally Uniqueness**
- **Partial Tallying**
- **Accuracy**

suffice for **weak verifiability**

- $BB, T, \pi_T$ is the output of $A$ in $\mathbf{Weak - Ver_{VS,A}}$
- $Verify(BB, T, \pi_T) = 1$ and $T \neq \perp$
- Since the BB is honest:
  - $\forall b \; in \; BB: IsValid(b) = 1$ (well formed)
  - **No ballot has been altered or deleted**
- We split $BB$ in 2 parts
  - One contains the honest votes (outputs of $OVote$)
  - One contains the votes of corrupt voters
  - $BB = BB_H \cup BB_{CR}$ and $BB_H \cap BB_{CR} = \emptyset$

# Sufficient Conditions For Weak Verifiability

- **We tally $BB_H$:**
- $T_H, \pi_{T_H} = Tally(BB_H, sk)$
- From correctness we get that:
  - $T_H = result(\{v_i\}_{i=1}^{n_{V_H}})$

- **We tally $BB_{CR}$:**
- $T_{CR}, \pi_{T_{CR}} = Tally(BB_{CR}, sk)$
- This tally is unique
- $|BB_{CR}| \leq |V_{CR}|$ (honest BB)
- From accuracy $Verify\left(BB_{CR}, T_{CR}, \pi_{T_{CR}}\right) = 1$ and
- $T_C = result(\{v_i\}_{i=1}^{n_{V_{CR}}})$ by second condition of accuracy

From partial tallying $Tally(\text{BB}) = Tally(BB_1) \oplus Tally(BB_2)$

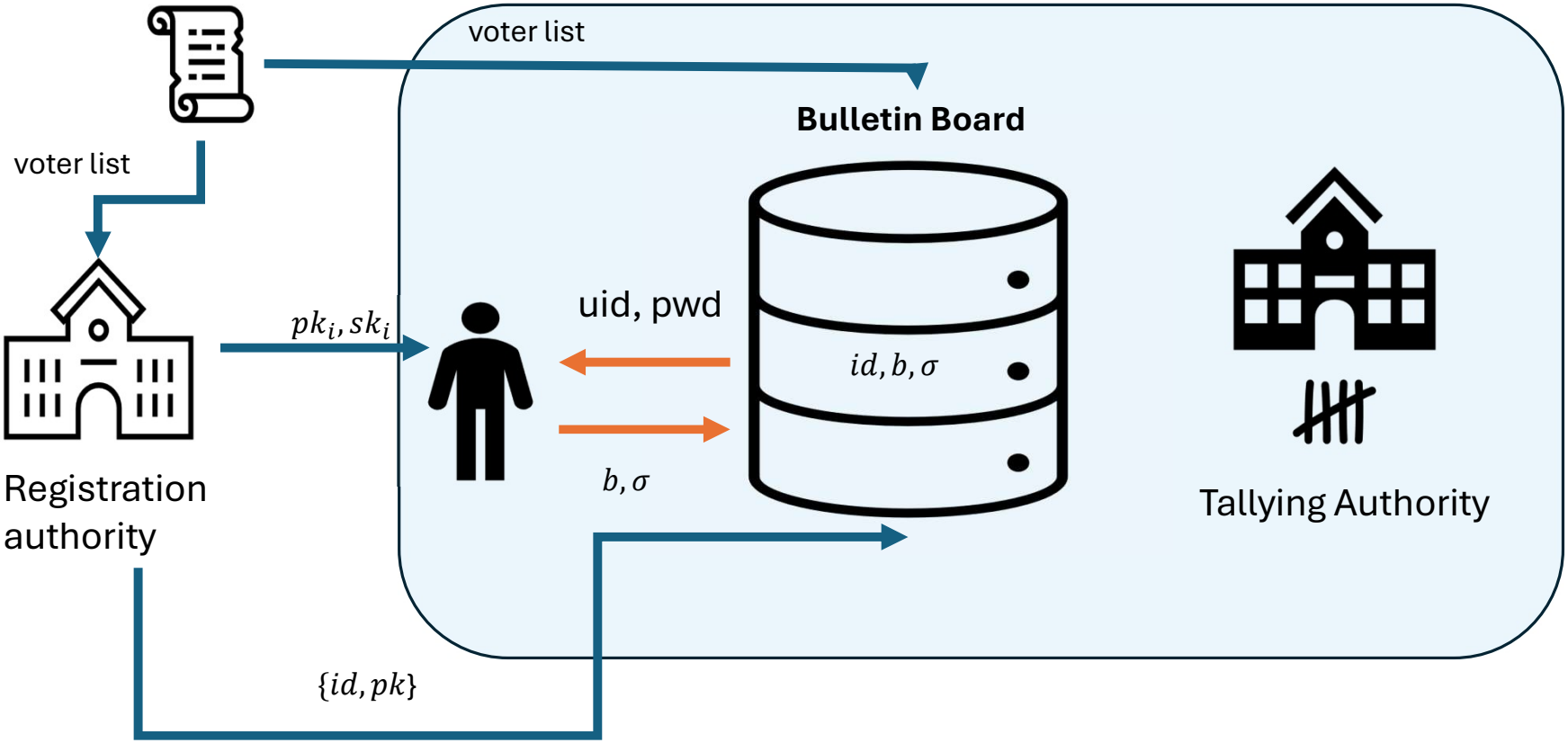
# Weak verifiability for Helios

- Correctness
  - Correctness of ElGamal encryption
  - Completeness of $DLPRV, EQPRV, DJPRV$
- Tally uniqueness
  - Special soundness of $DLPRV, EQPRV, DJPRV$
- Accuracy
  - $IsValid(\cdot) = 1, Verify(\cdot) = 1$ then $v \in CS$ with negligible probability because of the negligible soundness error of $DJPRV$

# A generic construction for strong verifiability

- Given a $VS$ with weak verifiability we can construct a $VS^{\sigma}$ with strong verifiability as follows:
  - Add a EUF-CMA secure signature scheme
  - Add a registration authority that hands credentials (secret keys) to the voters
  - Each voter **signs the ballot** using the secret key
  - The RA announces the list of public credential for eligible voters
    - Unordered, disassociated with voter identities
  - Voter login to the BB
  - The validation by the BB includes
    - Signature verification using a public key obtained from the public list
    - Maintenance of the mapping between votes and id's (keep one vote per voter)

# A generic construction for strong verifiability



# Sufficient Conditions for Strong Verifiability

*A voting system with weak verifiability combined with an existentially unforgeable signature scheme provides strong universal verifiability*

**Case 1: Corrupted RA**

- Since the RA is corrupted the adversary has all the credentials.
- However, the authenticated channel between A and the honest BB forbids him from ballot stuffing

(same intuition as why Helios provides weak verifiability with honest BB)

# Case 2: Corrupted BB

Every adversary $A^{\sigma}$ against $VS^{\sigma}$ **is as powerful as** an adversary $A$ against a weakly verifiabile $VS$, unless he can break EUF-CMA.

Facts (from strong verifiability definition):

- $T \neq \perp$ and $Verify^{\sigma}(BB^{\sigma}, T, \pi) = 1$
- $BB^{\sigma}$ is well-formed, since it passes $Verify^{\sigma}$.
- This means that all ballots in $BB^{\sigma}$ are valid. Thus:
- For every honest vote that has a corresponding ballot $b_H = (pk_u, a_H, \sigma)$ in $BB_{\sigma}$ there exists has also a ballot $a_H$ in BB which is valid (from weak verifiability). Thus, it is counted.

# Case 2: Corrupted BB

Every vote $vt \in V_H \setminus V_{CH}$ that has a corresponding ballot in $BB^{\sigma}$ corresponds to an honest vote (output of Vote)

If not: since it is placed in $BB^{\sigma}$ it must have a valid signature.

Since $\sigma$ does not come from Vote then it must have been forged (contradiction).

Conclusion: Every $vt \in V_H \setminus V_{CH}$ comes from Vote.

- $nCorr \leq |VCorr|$

If not:

There are two (at least 2) ballots in $BB^{\sigma}$ with the same credential. But $BB^{\sigma}$ is well-formed.

Or: $A^{\sigma}$ added a valid ballot without calling Corrupt (without knowing $sk_i$).

This contradicts unforgeability again

# Ballot Privacy

- Goal: Nobody can learn the vote cast by a voter
- Recall:
  - Ballot privacy is not absolute: The result leaks information
    - In a unanimous vote there is no privacy
    - In an all-but-one result, the person who voted differently knows how everyone else voted
    - The result yields a probability of a particular vote
      - Important in small populations
- The adversary should not learn anything else!

# Threat model for privacy

- The adversary may corrupt some voters
- The system (EA, RA, TA) is honest!
  - Not sure if this acceptable for voters
- Intuition:
  - Indistinguishability games for cryptographic secrecy
  - Instead of distinguishing between message encryptions the adversary tries to distinguish between voting scenarios
  - Considering:
    - Verifiability compliance
    - Different voting rules

# Ballot independence

- A voter may not repost a related version of a ballot that is already in the BB
  - E.g. take advantage of the malleability of the underlying cryptosystem and post a different version of a ballot found in the BB
- Lack of ballot independence can break ballot privacy
  - Assume an election with 3 voters $V_1, V_2, V_3$
  - The adversary replays a ballot (e.g. of $V_1$)
  - The candidate that receives more than two votes is the one preferred by $V_1$
- Large scale implementation
  - Replay ballots in place of absentee voters
  - Observe the shift of the distribution of votes from previous elections
  - May not reveal individual votes but leaks some information

# Countermeasures

- Ballot weeding
  - Filter out ballots with exact duplicate parts
- Strong Fiat-Shamir Heuristic to counter encryption malleability
  - Recall: Enc + PoK provides non malleability
- Add voter id as an extra input to the hash function for non-interactive proofs

# The BPRIV Framework

- The adversary tries to distinguish between two worlds by having access to their respective $BB$s
  - The real $BB$ ($BB_0$):
    - Contains honest and adversarial ballots
  - The fake $BB$ ($BB_1$):
    - When the adversary has access to the fake BB, the tally is computed from the real BB and the proofs of correctness are simulated
    - Otherwise, it would be trivial for the adversary to win the game because they control the selections of the honest voters

David Bernhard, Véronique Cortier, David Galindo, Olivier Pereira, and Bogdan Warinschi, Sok: A comprehensive analysis of game-based ballot privacy definitions, 2015 IEEE Symposium on Security and Privacy, 2015, pp. 499–516.

# The BPRIV Framework

The adversary has access to the following oracles:

- $Board(b)$:
  - Retrieve the public contents of $BB_b$ (Publish Operation)
- $Vote(i, v_0, v_1)$:
  - Select two votes and post the corresponding ballots to $BB_0, BB_1$ respectively
  - Both ballots are honestly created

$\boldsymbol{OBoard(b)}$

$return\ Publish(BB_b)$

$\boldsymbol{OVote(i, v_0, v_1)}$

$\mathrm{b}_0 = Vote(i, v_0)$
$\mathrm{b}_1 = Vote(i, v_1)$
$if\ Valid(\mathrm{b}_0, BB_0)\ and\ Valid(\mathrm{b}_1, BB_1)\ then$
$\quad BB_0 \Leftarrow \mathrm{b}_0\ ; BB_1 \Leftarrow \mathrm{b}_1$
*Else*
$\quad return\ \perp$

# The BPRIV Framework

The adversary has access to the following oracles:

- $Cast(\mathrm{i}, \mathrm{b})$:
  - Cast the same ballot to both $BBs$
  - Represents adversarial casting (e.g. ballot replication)
- $Tally(b)$:
  - Obtain the result of $BB_0$.
  - Yield real or simulated proofs
- The adversary can call the oracles Board, Vote, Cast at will
- The adversary can call Tally only once
- Finally the adversary must guess which $BB$ they viewed

$\boldsymbol{OCast(i, \mathrm{b})}$

$if\ Valid(\mathrm{b}\ , BB_0)\ and\ Valid(\mathrm{b}\ , BB_1)\ then$
$\quad BB_0 \Leftarrow \mathrm{b}\ ; BB_1 \Leftarrow \mathrm{b}$
*Else*
$\quad return\ \perp$

$\boldsymbol{OTally(b)}$

if $b = 0$ then
$\quad (T, \pi) \leftarrow Tally(sk_{TA}, CS, BB_0)$
else
$\quad (T, \pi) \leftarrow Tally(sk_{TA}, CS, BB_0)$
$\quad \pi \leftarrow Sim(sk_{TA}, CS, BB_0, BB_1, T)$
Return $(T, \pi)$

# The BPRIV definition

$\boldsymbol{BPRIV^{b}_{VS,A}}$

$(pk_{TA}, sk_{TA}) \leftarrow VS.Setup(1^{\lambda})$
$CS \leftarrow A(\mathrm{pk_{TA}})$
$b' \leftarrow A^{OBoard,OVote,OCast,OTally}(pk_{TA})$
return $b = b'$

A voting system $VS$ supports ballot privacy if there exists a simulator $S$ such that ∀ PPT adversaries

$$\left|\Pr\left[\boldsymbol{BPRIV^{0}_{VS,A}}(1^{\lambda}) = 1\right] - \Pr\left[\boldsymbol{BPRIV^{1}_{VS,A}}(1^{\lambda}) = 1\right]\right| \leq negl(\lambda)$$

# Helios is BPRIV

- Observations
  - The visible BB consists of ballots (i.e tuples $(i, b)$ cast either through Vote or through Cast
  - For tuples originating from the Vote oracle the challenger maintains a table consisting of the respective inputs, outputs
    - $\{(i, v_0, b_0, v_1, b_1)\}$
  - For tuples originating from the Cast oracle the relevant entry of the table is
    - $\{(i, \_, b, \_, b)\}$
- Proof strategy
  - A sequence of games beginning from $BPRIV^0$ and ending to $BPRIV^1$ where the ballots are swapped without the adversary noticing

# Helios is BPRIV

- $Game_0$: $BPRIV^0$ where the adversary has access to $BB_0$ and the tally $T$ is computed from $BB_0$ ($OTally()$ with $b = 0$)
- $Game_1$: The proof of the tally is always simulated for the particular $T$ (with the help of the random oracle). The adversary cannot distinguish it except with negligible probability
- $Game_{2,i}, i \in [n]$:
  - For all ballots **cast through the Vote oracle** the challenger retrieves the relevant internal entry and swaps $b_{i0}$ with $b_{i1}$
  - **The adversary cannot distinguish any change because of the NM-CPA property of the ENC+PoK Scheme**
  - For all ballots cast through the Cast oracle nothing happens
- $Game_{2,n}$:
  - It is actually $BPRIV^1$

# Helios is BPRIV (NM-CPA ⇒ $Game_{2,i-1} \approx Game_{2,i}$)