Electronic Voting with Cryptography

Panagiotis Grontas

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

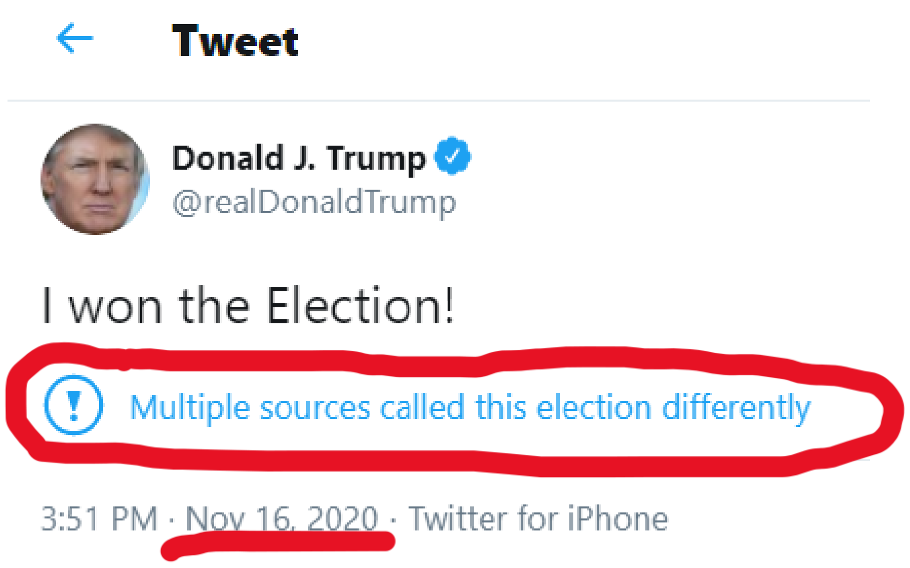
Advanced Topics In Cryptography
Προχωρημένα Θέματα Κρυπτογραφίας
Μάρτιος 2025

Contents

Introduction

Famous words…

It is enough that the people know there was an election. The people who cast the votes decide nothing. The people who count the votes decide everything.



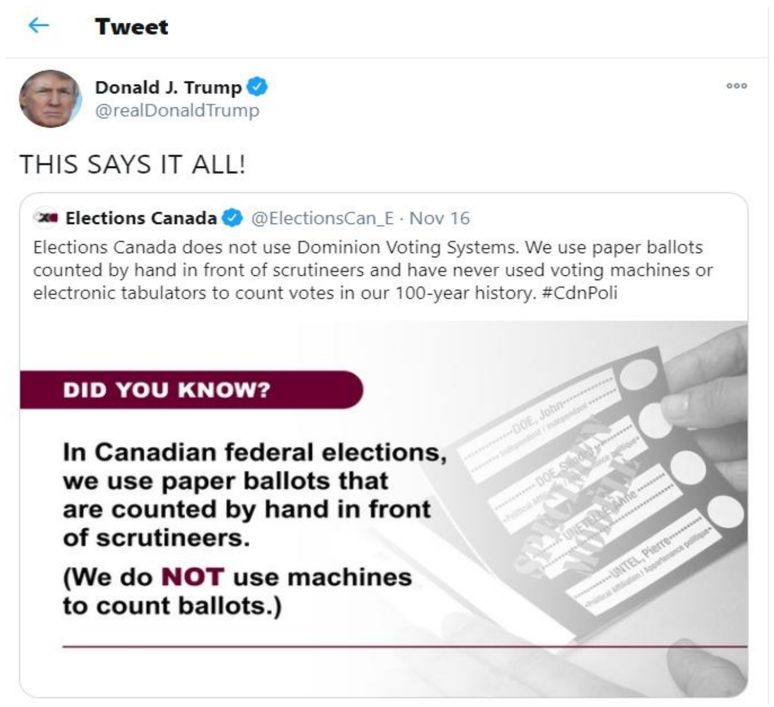
Tweet

Donald J. Trump
@realDonaldTrump

I won the Election!

Multiple sources called this election differently

3:51 PM · Nov 16, 2020 · Twitter for iPhone



Tweet

Donald J. Trump
@realDonaldTrump

THIS SAYS IT ALL!

Elections Canada @ElectionsCan_E · Nov 16
Elections Canada does not use Dominion Voting Systems. We use paper ballots counted by hand in front of scrutineers and have never used voting machines or electronic tabulators to count votes in our 100-year history. #CdnPoli

DID YOU KNOW?

In Canadian federal elections, we use paper ballots that are counted by hand in front of scrutineers.

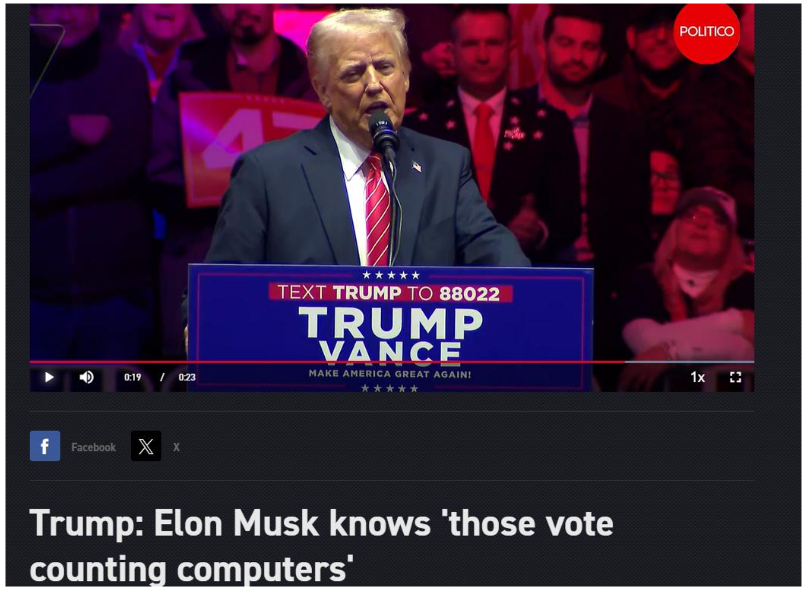
(We do NOT use machines to count ballots.)

The People have spoken…. the bastards!

Famous words…

The voting booth is separated by a curtain and there is a guy behind the curtain that would write down your vote. You dictate the vote and once you ‘re done you leave, without being able to look at the ballot. Most people in their right mind, would not trust this process. The guy behind the curtain could be incompetent, hear the votes wrong and register it incorrectly or it could be that he did not like your political affiliation and prefer your vote would go to another party

Internet voting is like drunk driving…



Trump: Elon Musk knows 'those vote counting computers'

The voting problem

Really, isn't it all about counting? What is difficult about that?

- Elections

A distributed procedure to reach a common decision

… as old as societies

… streamlined with each era's technology

… with conflicting security requirements

… where every participant is an adversary

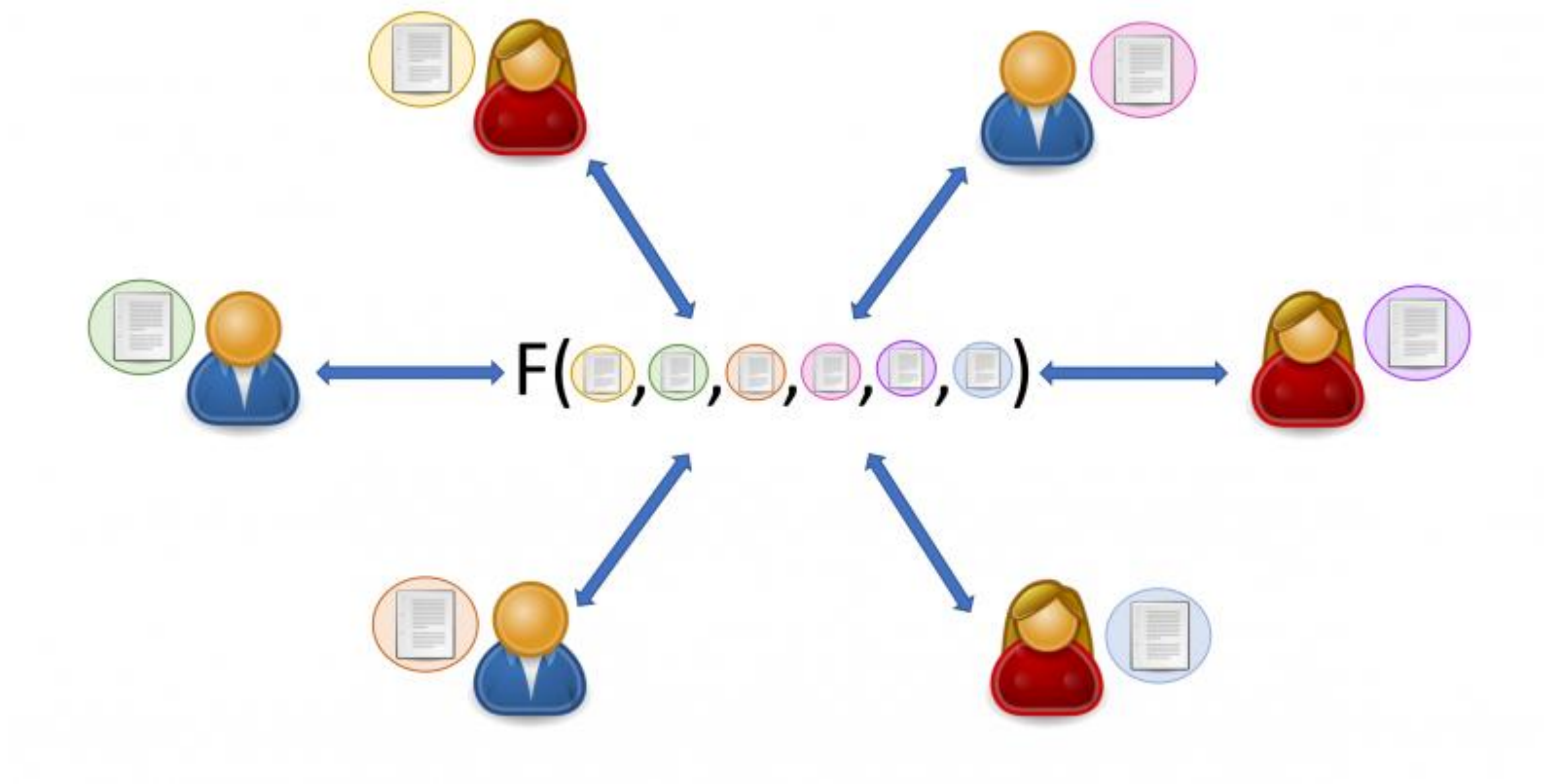
- Electronic Elections

… are already happening

  - Voter registration
  - Partial result communication and combination
  - Winner announcements

- Election only with computers
  - Inherent problems are made worse

Electronic Voting



Secure Multiparty Computation
with **stronger** security and **usability** requirements

Security Requirements - Correctness

- Integrity
  - The result corresponds to the ballots cast
  - Not enough…
- Verifiability
  - The voter (esp. one supporting the losing side) should be convinced about integrity
  - By checking election data
  - Enables voters to regain the trust endangered by the volatile nature of computer systems and the motives of voting authorities (systemic errors or malice)

Adversary: The voting system itself

Verifiability

- Types of verifiability
  - Cast as intended
  - Recorded as cast
  - Tallied As Recorded
  - E2E Verifiability
  - Eligibility Verifiability
    - Avoid ballot stuffing

- Ways to verify
  - Individual
    - Cast as intended / Recorded As Cast
  - Universal
    - Any interested party
  - Administrative (TTP)
    - Real world elections

Verifiability ≠ Verification

Security Requirements - Privacy

- Privacy
  - The voter must express their true will
  - Secrecy
    - The vote is tied to the voter
    - The contents of the vote are never revealed
  - Anonymity
    - The vote is disassociated from the voter identity
    - Its contents can be revealed

- Adversary
  - The voting system
    - Ballot privacy
  - Voters themselves
    - Vote selling
    - Receipt Freeness
  - Other voters
    - Passive
    - Active - Coercers
    - Coercion Resistance

Privacy

- Secrecy in voting differs from secrecy in other applications (e.g. in secure messaging)
  - Ballot privacy is not absolute
  - The result leaks information
    - In a unanimous vote, everyone knows how everyone voted
    - In an all-but-one vote, the one that differs knows how everyone else voted
  - The result also yields a probability of a particular vote
    - Important in small voting populations

The primary incompatibility

- Privacy without verifiability
  - Useless
  - We don’t know if our vote will be considered
  - Leads to abstention

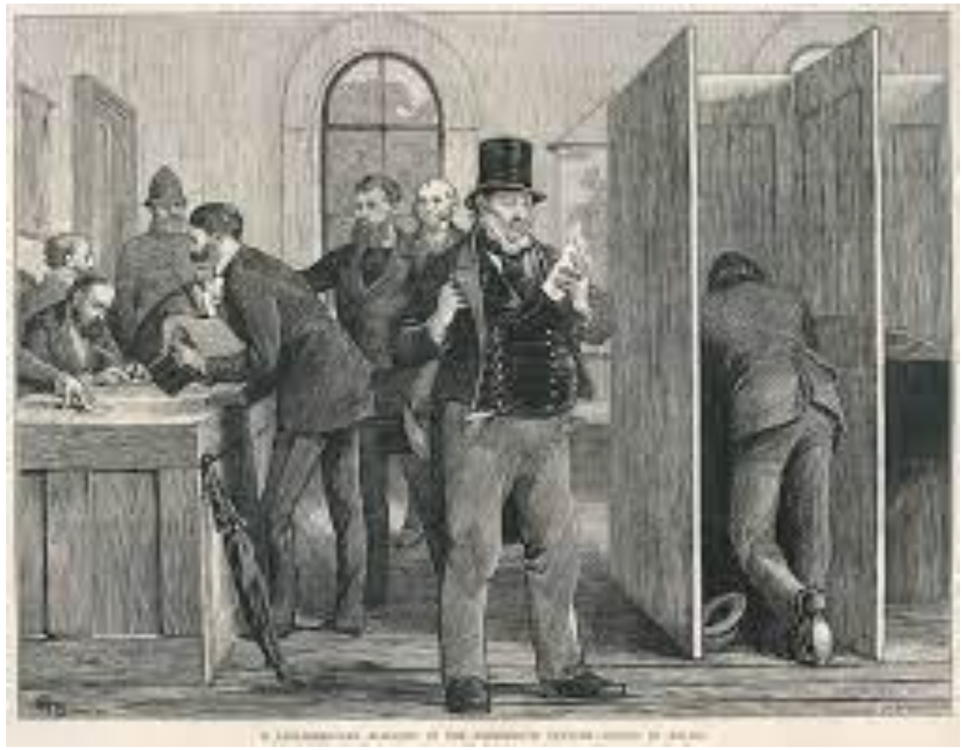
- Verifiability without privacy
  - Raise of hands
  - The lack of privacy forces the voters to self – censor
    - i.e., the vote loses the integrity property before it leaves the voter

Other requirements

- Fairness
  - No intermediate results are made public
- Enfranchisement
  - The process is open to all
  - And understood by all
- Availability
- Efficiency
  - Time
  - Money

Traditional Elections: Australian Ballot

- Privacy
  - Primitive countermeasures
    - Voting in a specialized booth
    - Envelope
    - Ballot box
    - Ballot Shuffling
    - Trust in the Electoral Committee
- Verifiability
  - Only administrative!
- Integrity
  - Trust in the Electoral Committee
  - Conflicting interests
  - Trusted Third Parties



Problems in traditional elections

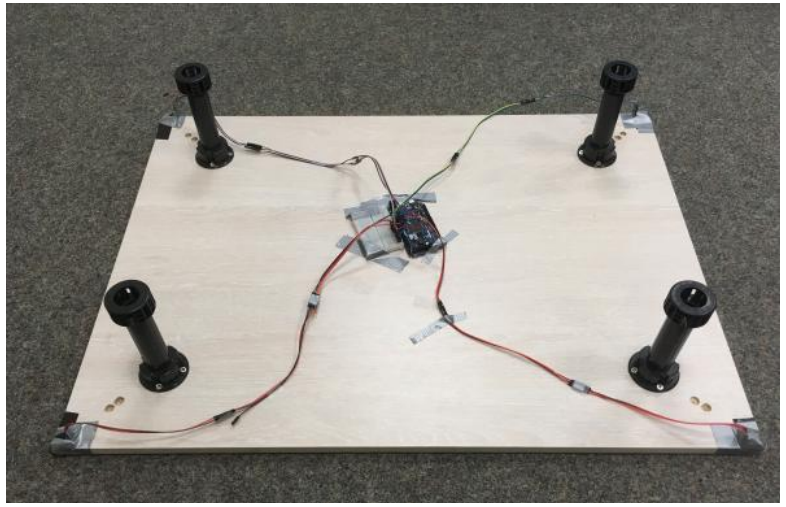
- The counting process is time-consuming.
- There are significant infrastructure expenses.
    - Ballots
    - Voting locations
    - Payment for trusted third parties
- Implementing intricate counting functions is challenging.
    - Solutions tend to raise costs
    - Elections that involve multiple rounds
- Considerations for voters with special needs.

Attacks against traditional elections: Integrity

- Before voting
  - Changing of the voter rolls
- During voting
  - Invalid ballots
  - Ballot stuffing
- During counting
  - Omit ballots
  - Cancel ballots
  - Changing Ballots
- During result announcement
  - Different result
- Countermeasures
  - Conflicting interests
  - Trusted third parties

Attacks against traditional elections: Privacy

- Incorrect ballot shuffling
  - Correlate with voting order
- Target a voter and mark their ballot
  - Different color
  - Different type of paper
- Fingerprints?
- Side channels



Countermeasures

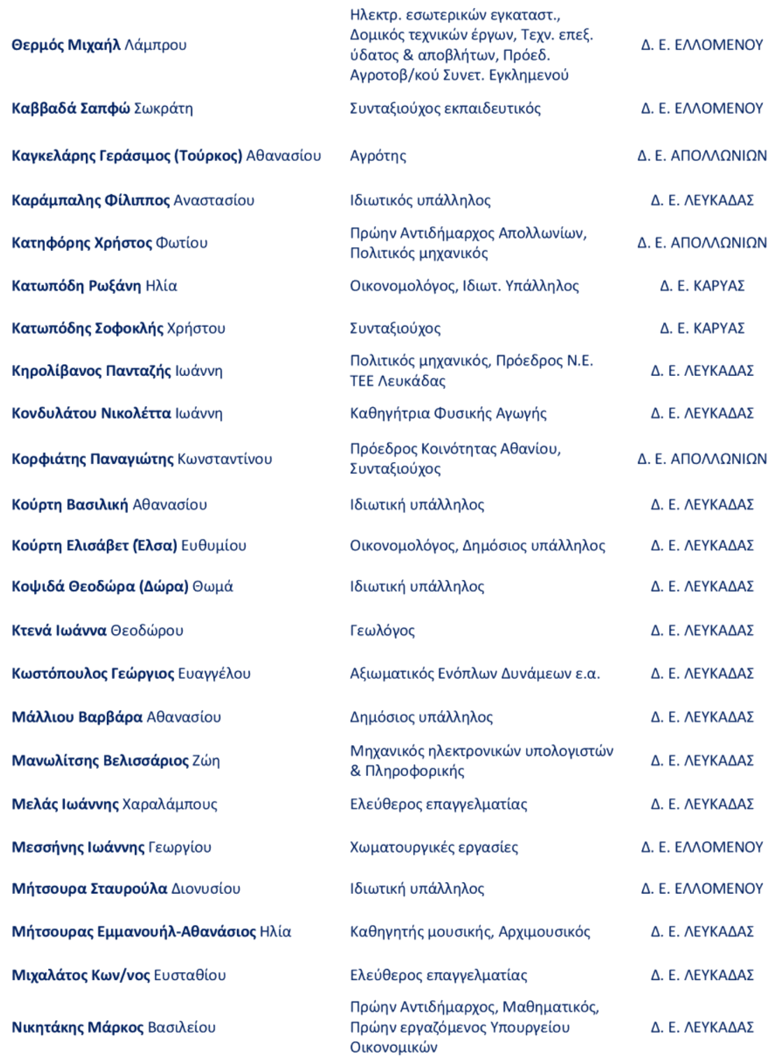
Conflicting interests

Trusted third parties

K. Krips, J. Willemson and S. Värv, "Is Your Vote Overheard? A New Scalable Side-Channel Attack Against Paper Voting," *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, Stockholm, Sweden, 2019, pp. 621-634, doi: 10.1109/EuroSP.2019.00051.

Attacks against traditional elections: Vote selling – coercion resistance

- Photo of ballot
- Video of voting
  - Google glass
- Ballot switching
  - Coercer:
    - Prepare a ballot with a particular vote
  - Voter:
    - Return ballots for every candidate
- Italian (Large ballot) attack
  - Coercer:
    - You will vote for x and a particular (rare) permutation of candidates
  - Coercer:
    - Check the results for the rare permutation



Θερμός Μιχαήλ Λάμπρου	Ηλεκτρ. εσωτερικών εγκαταστ., Δομικός τεχνικών έργων, Τεχν. επεξ. ύδατος & αποβλήτων, Πρόεδ. Αγροτοβ/κού Συνετ. Εγκλημενού	Δ. Ε. ΕΛΛΟΜΕΝΟΥ
Καββαδά Σαπφώ Σωκράτη	Συνταξιούχος εκπαιδευτικός	Δ. Ε. ΕΛΛΟΜΕΝΟΥ
Καγκελάρης Γεράσιμος (Τούρκος) Αθανασίου	Αγρότης	Δ. Ε. ΑΠΟΛΛΩΝΙΩΝ
Καράμπαλης Φίλιππος Αναστασίου	Ιδιωτικός υπάλληλος	Δ. Ε. ΛΕΥΚΑΔΑΣ
Κατηφόρης Χρήστος Φωτίου	Πρώην Αντιδήμαρχος Απολλωνίων, Πολιτικός μηχανικός	Δ. Ε. ΑΠΟΛΛΩΝΙΩΝ
Κατωπόδη Ρωξάνη Ηλία	Οικονομολόγος, Ιδιωτ. Υπάλληλος	Δ. Ε. ΚΑΡΥΑΣ
Κατωπόδης Σοφοκλής Χρήστου	Συνταξιούχος	Δ. Ε. ΚΑΡΥΑΣ
Κηρολίβανος Πανταζής Ιωάννη	Πολιτικός μηχανικός, Πρόεδρος Ν.Ε. ΤΕΕ Λευκάδας	Δ. Ε. ΛΕΥΚΑΔΑΣ
Κονδυλάτου Νικολέττα Ιωάννη	Καθηγήτρια Φυσικής Αγωγής	Δ. Ε. ΛΕΥΚΑΔΑΣ
Κορφιάτης Παναγιώτης Κωνσταντίνου	Πρόεδρος Κοινότητας Αθανίου, Συνταξιούχος	Δ. Ε. ΑΠΟΛΛΩΝΙΩΝ
Κούρτη Βασιλική Αθανασίου	Ιδιωτική υπάλληλος	Δ. Ε. ΛΕΥΚΑΔΑΣ
Κούρτη Ελισάβετ (Έλσα) Ευθυμίου	Οικονομολόγος, Δημόσιος υπάλληλος	Δ. Ε. ΛΕΥΚΑΔΑΣ
Κοψιδά Θεοδώρα (Δώρα) Θωμά	Ιδιωτική υπάλληλος	Δ. Ε. ΛΕΥΚΑΔΑΣ
Κτενά Ιωάννα Θεοδώρου	Γεωλόγος	Δ. Ε. ΛΕΥΚΑΔΑΣ
Κωστόπουλος Γεώργιος Ευαγγέλου	Αξιωματικός Ενόπλων Δυνάμεων ε.α.	Δ. Ε. ΛΕΥΚΑΔΑΣ
Μάλλιου Βαρβάρα Αθανασίου	Δημόσιος υπάλληλος	Δ. Ε. ΛΕΥΚΑΔΑΣ
Μανωλίτσης Βελισσάριος Ζώη	Μηχανικός ηλεκτρονικών υπολογιστών & Πληροφορικής	Δ. Ε. ΛΕΥΚΑΔΑΣ
Μελάς Ιωάννης Χαραλάμπους	Ελεύθερος επαγγελματίας	Δ. Ε. ΛΕΥΚΑΔΑΣ
Μεσσήνης Ιωάννης Γεωργίου	Χωματουργικές εργασίες	Δ. Ε. ΕΛΛΟΜΕΝΟΥ
Μήτσουρα Σταυρούλα Διονυσίου	Ιδιωτική υπάλληλος	Δ. Ε. ΕΛΛΟΜΕΝΟΥ
Μήτσουρας Εμμανουήλ-Αθανάσιος Ηλία	Καθηγητής μουσικής, Αρχιμουσικός	Δ. Ε. ΛΕΥΚΑΔΑΣ
Μιχαλάτος Κων/νος Ευσταθίου	Ελεύθερος επαγγελματίας	Δ. Ε. ΛΕΥΚΑΔΑΣ
Νικητάκης Μάρκος Βασιλείου	Πρώην Αντιδήμαρχος, Μαθηματικός, Πρώην εργαζόμενος Υπουργείου Οικονομικών	Δ. Ε. ΛΕΥΚΑΔΑΣ

First generation electronic voting

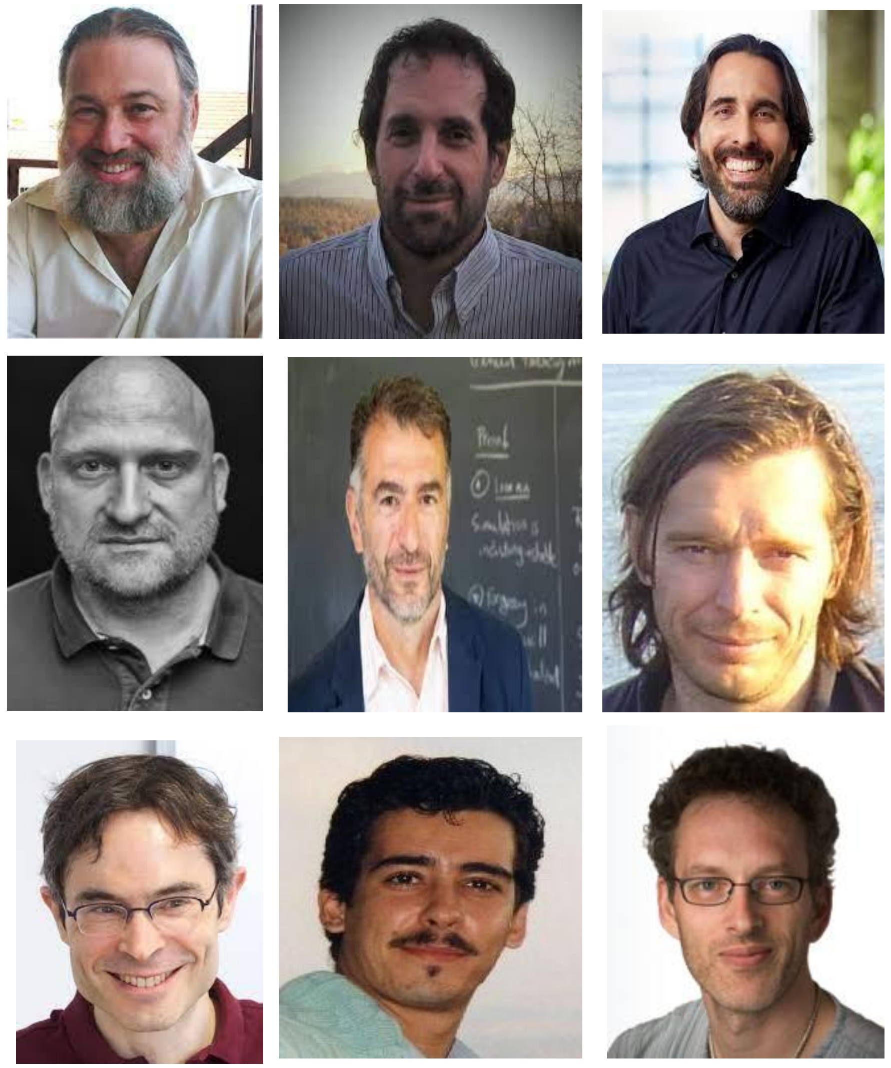
- In reality
  - Replace the ballot box with a computer
  - Input the voter choice
  - Electronic counting
  - No secrecy whatsoever!
- Voting with an untrusted intermediary
  - Malicious software
  - Programming errors
  - Targeted attacks
  - Interface problems
- No verifiability
- Open source: Necessary but not sufficient

First generation electronic voting

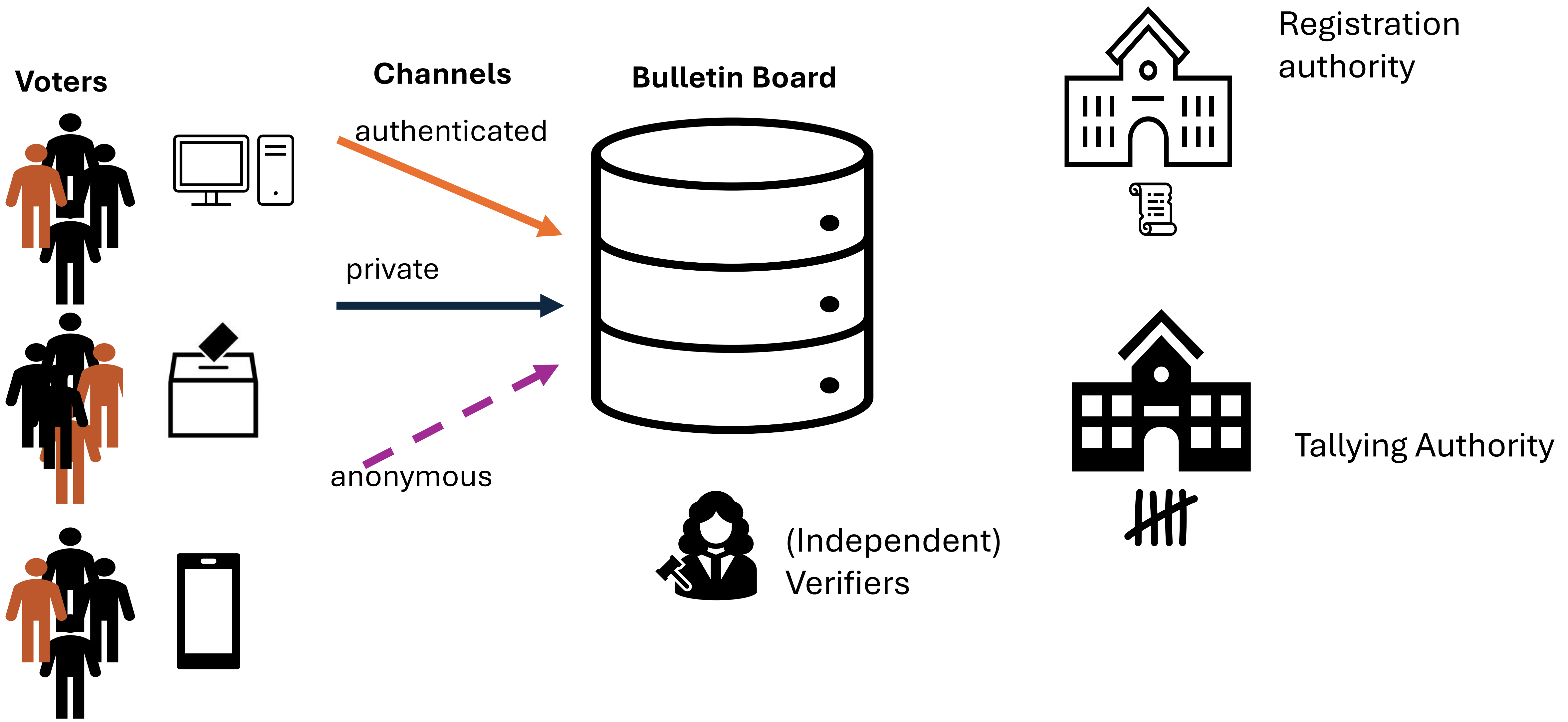
- Software independence (Rivest)
  - *A voting system is software-independent if an undetected change or error in its software cannot cause an undetectable change or error in an election outcome.*
- **Solution 1^n:**
  - Voter Verifiable Paper Trail (VVPAT) + Risk Limiting Audits (RLA)
- **Solution 2^n:**
  - Cryptography!

Second generation electronic voting

- Elections without TTPs
- Cryptography
  - Secrecy
  - Integrity
  - Verifiability
- Basic Ideas
  - David Chaum (1981)
  - Josh Benaloh (1987)
  - Ben Adida (2008)
  - Cramer, Gennaro, Schoenmakers (1997)
  - Juels, Catalano, Jakobsson (2005)



General Architecture



Bernhard, M. *et al.* (2017). Public Evidence from Secret Ballots. In: Krimmer, R., Volkamer, M., Braun Binder, N., Kersting, N., Pereira, O., Schürmann, C. (eds) Electronic Voting. E-Vote-ID 2017. Lecture Notes in Computer Science(), vol 10615. Springer, Cham. https://doi.org/10.1007/978-3-319-68687-5_6

Cryptographic Voting Schemes

Architecture and Primitives

Public Key Cryptosystems

- ElGamal Encryption
  - $\mathbb{G}$ is a cyclic group of prime order q generated by g
  - $sk \overset{\$}{\leftarrow} \mathbb{Z}_q, pk = g^{sk}$
  - pk belongs to the tallying authority
  - $Enc_{pk}(m) = (g^r, m \cdot pk^r), r \overset{\$}{\leftarrow} \mathbb{Z}_q, m \in \mathbb{G}$
  - $Dec_{sk}(c) = c_2 \cdot c_1^{-\mathrm{sk}} = \mathrm{m}$
- Exponential ElGamal
  - $Enc_{pk}(m) = (g^r, g^m \cdot pk^r), r \overset{\$}{\leftarrow} \mathbb{Z}_q, m \in \mathbb{Z}_q$
  - $Dec_{sk}(c) = c_2 \cdot c_1^{-\mathrm{sk}} = \mathrm{g}^{\mathrm{m}}$
  - Solve ‘small’ DLOG
- Homomorphic properties

$$Enc_{pk}(v_1) \otimes Enc_{pk}(v_2) =$$

$$(g^{r_1}, g^{v_1} \cdot pk^{r_1}) \otimes (g^{r_2}, g^{v_2} \cdot pk^{r_2}) =$$

$$(g^{r_1+r_2}, g^{v_1+v_2} \cdot pk^{r_1+r_2}) = Enc_{pk}(v_1 + v_2)$$

- Reencryption

$$ReEnc_{pk}(c) = c \otimes Enc_{pk}(1) =$$

$$(g^r, m \cdot pk^r) \otimes (g^{r_1}, pk^{r_1}) = (g^{r \ \ +r_1}, m \cdot pk^{r \ \ +r_2})$$

Alternatives: Paillier Cryptosystem, DJ Cryptosystem

- Paillier, Pascal (1999). "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes" (PDF). EUROCRYPT ’99.
- Ivan Damgård, Mads Jurik: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. Public Key Cryptography 2001: 119-136

Benaloh Challenge

A *cut & choose* technique to encrypt a vote from an untrusted device

- The voter enters the choice to the device
- The device creates the ciphertext
- The voters selects Audit or Cast
- On Audit
  - The device releases the randomness used to encrypt the choice
  - The voter can recreate the encryption on their own
  - The encrypted vote is not admissible
  - Repeat
- On Cast
  - The ballot is sent to the **BB**

Basic Idea:

- The device does not know in advance if the voter will audit or cast
- If it changes the voter input it might be caught
- Game theoretic argument

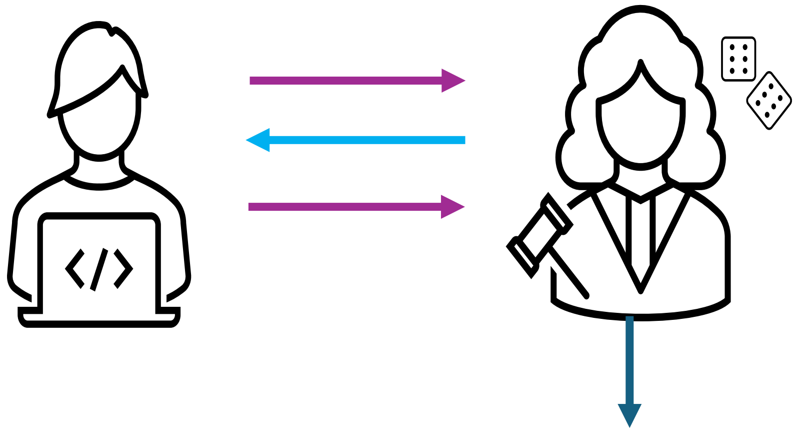
Wojciech Jamroga:
Pretty Good Strategies for Benaloh Challenge. E-Vote-ID 2023: 106-122

Commitment schemes

- **Pedersen Commitments**
- $\mathbb{G}$ is a cyclic group of prime order q generated by g, h
  - $Commit(m) = g^m h^r$
  - $Open(c, m, r) = (g^m h^r =_? c\ )$
- Perfectly hiding
- Binding if DLOG is hard
- If $DLOG_g(h) = x$ is known:
  - $m,\ m + x(r - r')\ mod\ q$ have the same commitments under r, r'
- Trusted setup!

- **Generalized Form**
- Commitment to a vector $\mathbf{m} = (m_1, \dots, m_n)$
- $\mathbb{G}$ is a cyclic group of prime order q generated by $g_1, \dots, g_n, h$
  - $Commit(\mathbf{m}) = h^r \prod_i g_i^{m_i}$

Schnorr’s Protocol

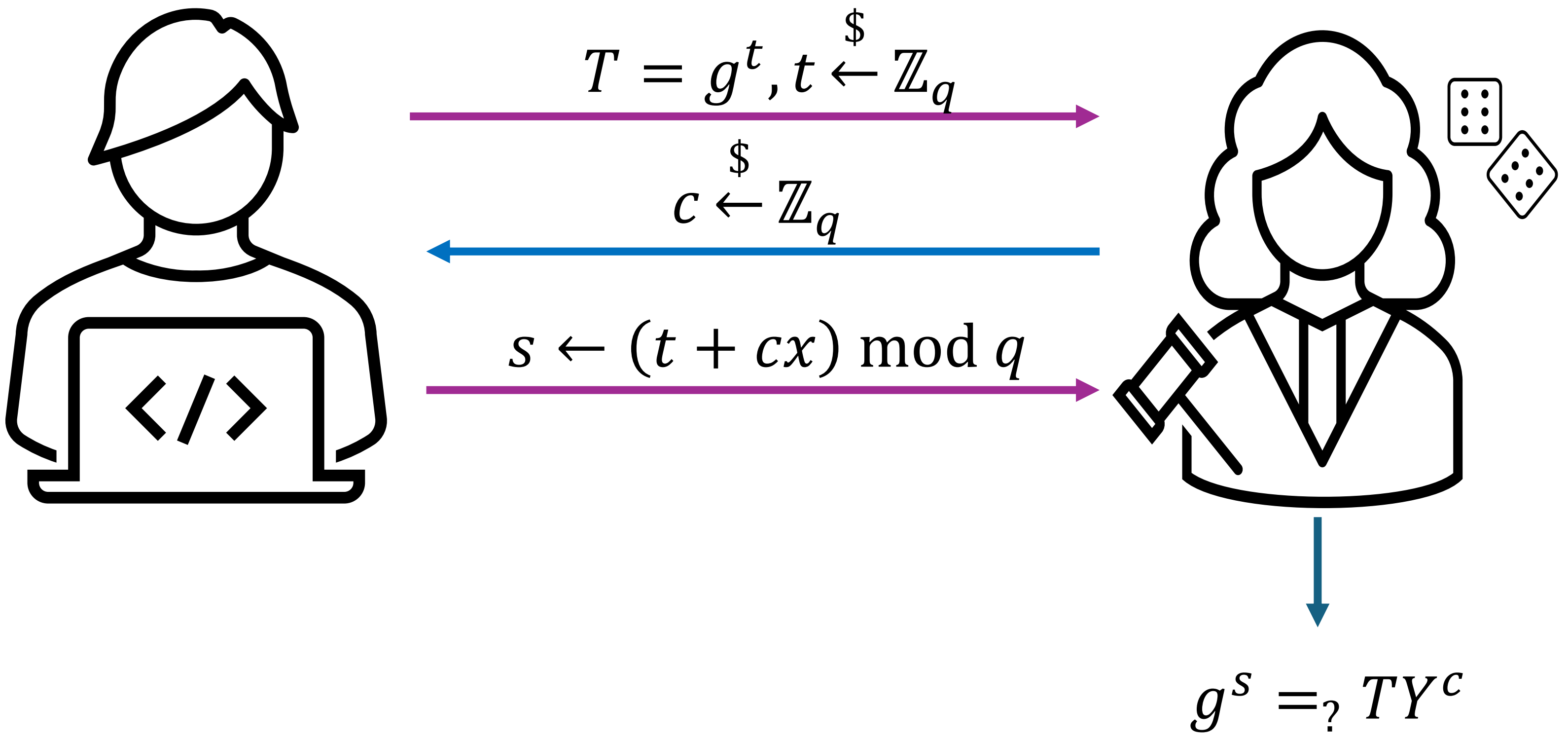


- Proof of Knowledge of a Discrete Logarithm
- $PoK\{x: g^x = Y : Y, g \in \mathbb{G}\}$
- Public Input
  - $\mathbb{G}$ is a cyclic group of prime order q generated by g
  - A group element $Y \in \mathbb{G}$
- Witness
  - $x \in \mathbb{Z}_q$

Schnorr, C. P. (1991). "Efficient signature generation by smart cards". Journal of Cryptology. 4 (3): 161–174. doi:10.1007/BF00196725. S2CID 10976365.

Schnorr's Protocol (II)

$$PoK\{x : g^x = Y : Y, g \in \mathbb{G}\}$$


$T = g^t, t \overset{\$}{\leftarrow} \mathbb{Z}_q$
$c \overset{\$}{\leftarrow} \mathbb{Z}_q$
$s \leftarrow (t + cx) \bmod q$
$g^s =_? TY^c$

Non-interactive Schnorr (**DLPRV**)

- Public input: $g \in \mathbb{G},\ ord(\mathbb{G}) = q, Y \in \mathbb{G}$
- Private input: $x \in\ \mathbb{Z}_q : Y = g^x$

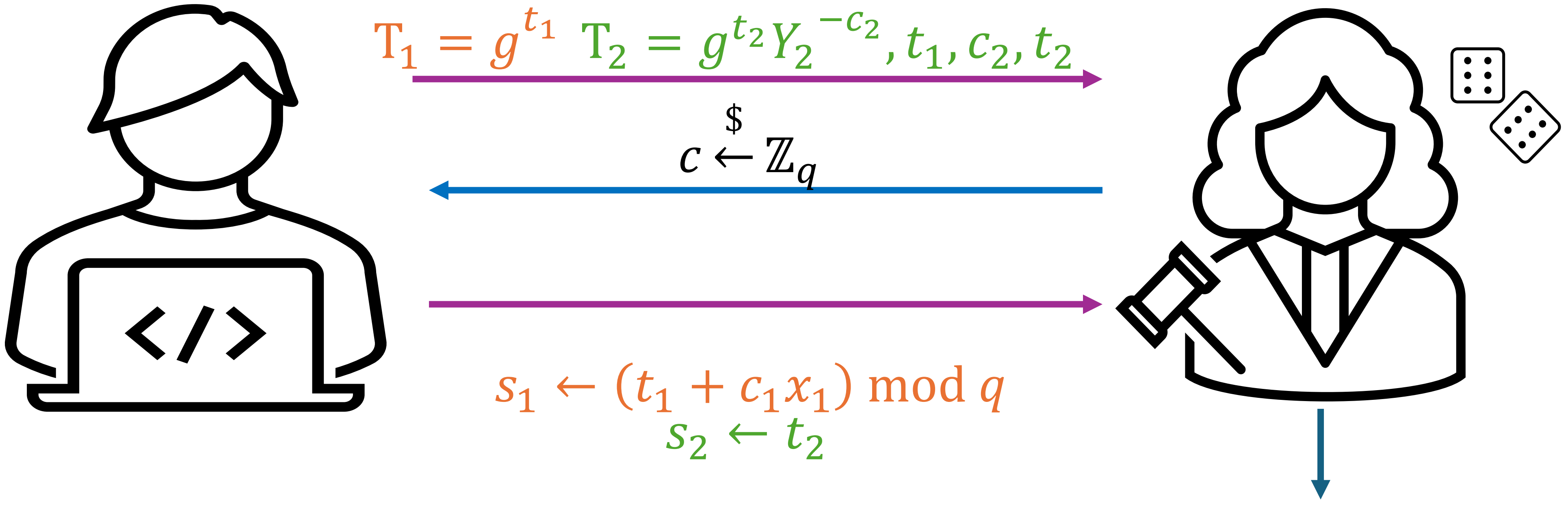
$\boldsymbol{DLPRV}(x, g, Y)$

- Select $t \overset{\$}{\leftarrow} \mathbb{Z}_q$ and compute $T = g^t$
- Compute $\boldsymbol{c} \leftarrow \boldsymbol{H}(\boldsymbol{g}, \boldsymbol{Y}, \boldsymbol{T})$
- Compute $s \leftarrow t + cx$
- The proof is: $\pi = (c, s)$
- **DLVF:** Public verifiability by checking if $c = H(g, Y, g^s Y^{-c})$

$\boldsymbol{DLVF}(g, Y, \pi)$

OR Schnorr (**DJPRV**)

- Proof of knowledge of one out of two DLOGs
- $PoK\{(\boldsymbol{x}_1, x_2): g^{\boldsymbol{x}_1} = Y_1 \textbf{ OR } g^{x_2} = Y_2,\ Y_1, Y_2, g \in \mathbb{G}\}$



$T_1 = g^{t_1}$ $T_2 = g^{t_2} Y_2^{-c_2}, t_1, c_2, t_2$

$c \overset{\$}{\leftarrow} \mathbb{Z}_q$

$s_1 \leftarrow (t_1 + c_1 x_1) \bmod q$

$s_2 \leftarrow t_2$

DJVF

$g^{s_1} =_? T_1 Y_1^{c_1}$ and $g^{s_2} =_? T_2 Y_2^{c_2}$

and $c = c_1 + c_2$

Pitfalls of the Fiat-Shamir Heuristic

- **Weak FS**: Input to hash function contains only commitment
  - $c \leftarrow H(T)$
- **Strong FS**: Input to hash function contains commitment, statement to be proved and all public values generated so far.
  - $\boldsymbol{c \leftarrow H(g, Y, T)}$
- If the prover is allowed to select their statement adaptively then the weak FS yields unsound proofs
- Proofs created using the weak FS have implications to the privacy and verifiability of Helios and other similar voting systems.

Bernhard, Pereira, Warinschi (2012) How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. *ASIACRYPT 2012*

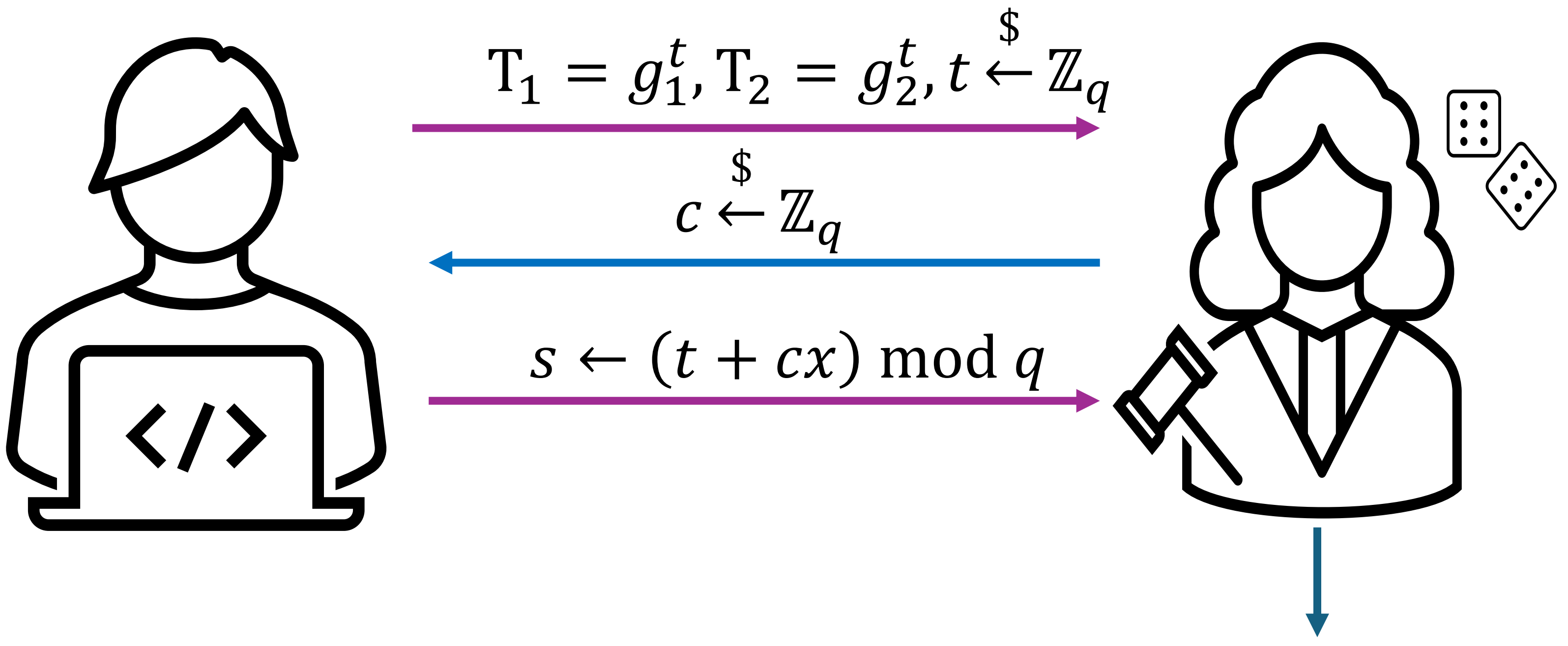
Pitfalls of the Fiat-Shamir Heuristic: An Example

- $\boldsymbol{DLPRV}(x, g, Y)$ proves knowledge of DLOG for a particular $Y \in \mathbb{G}$ which given as input to the prover
- If Y could be selected adaptively (after the proof):
  - Select $T \overset{\$}{\leftarrow} \mathbb{G}$
  - Compute $c \overset{\$}{\leftarrow} H(T)$
  - Select $s \overset{\$}{\leftarrow} \mathbb{Z}_q$
- The tuple (T, c, s) is a proof of knowledge for $Y \ = (g^{-s}T)^{-\frac{1}{c}}$ for which the DLOG is not known!
- Indeed:
  - $\boldsymbol{DLVF}(g, Y, \pi)$ checks in reality if $g^s Y^{-c} = T$ which holds by construction
  - $g^s Y^{-c} = g^s \left((g^{-s}T)^{-\frac{1}{c}}\right)^{-c} = T$

Bernhard, Pereira, Warinschi (2012) How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. *ASIACRYPT 2012*

Chaum – Pedersen protocol (**EQPRV**)

- Proof of knowledge and equality of two DLOGs
- $PoK\{x: g_1^x = Y_1, g_2^x = Y_2 : Y_1, Y_2, g_1, g_2 \in \mathbb{G}\}$


$\mathrm{T}_1 = g_1^t, \mathrm{T}_2 = g_2^t, t \overset{\$}{\leftarrow} \mathbb{Z}_q$
$c \overset{\$}{\leftarrow} \mathbb{Z}_q$
$s \leftarrow (t + cx) \bmod q$

EQVF $g^s =_? T_1 Y_1^c$ and $g^s =_? T_2 Y_2^c$

Pitfalls of the Fiat-Shamir Heuristic (again)

$\boldsymbol{EQPRV}(x, g_1, Y_1, g_2, Y_2)$ proves knowledge and equality of DLOG for a **particular** $Y_1, Y_2 \in \mathbb{G}$ which is given as input to the prover

- If Y_1, g_2, Y_2 could be selected adaptively (after the proof):
    - Select $\alpha, \beta, \gamma, \delta \overset{\$}{\leftarrow} \mathbb{Z}_q$
    - Compute $T_1 \leftarrow g_1^{\alpha}, \mathrm{T}_2 \leftarrow g_1^{\beta}, g_2 \leftarrow g_1^{\gamma}, Y_2 \leftarrow g_1^{\delta}$
    - Compute $c \leftarrow H(T_1, T_2)$
    - Compute $s \leftarrow \frac{\beta + c\delta}{\gamma}$
    - Compute $x \leftarrow \frac{s-a}{c}$ and $Y_1 \leftarrow g^{x}$
    - $x \neq \log_{g_2} Y_2 \leftarrow \delta/\gamma$ but the proof verifies!
- Indeed, $\boldsymbol{EQVF}$ returns true:
    - $g_1^{s} Y_1^{-c} = g_1^{s - c\frac{s-a}{c}} = g_1^{a} = T_1$ and
    - $g_2^{s} Y_2^{-c} = g_1^{\gamma s - \delta c} = g_1^{\gamma \frac{\beta + c\delta}{\gamma} - \delta c} = g_1^{\beta} = T_2$

Bernhard, Pereira, Warinschi (2012) How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. *ASIACRYPT 2012*

Application to modern constructions

- FROZEN HEART (FoRging Of ZEro kNowledge proofs)
  - Girault's proof of knowledge protocol (Schnorr over a composite modulus)
  - Bulletproofs
  - PLONK

Takeaway
The Fiat-Shamir hash computation must include all public values from the zero-knowledge proof statement and all public values computed intermediately the proof (i.e., all random "commitment" values)

Enc+PoK for non-malleability

- Malleability:
  - The ability to transform a valid ciphertext into another (meaningfully related) valid ciphertext without decrypting and encrypting again
- To achieve non malleability the $Enc + PoK$ construction may be used:
  - Append a NIZK PoK of *the encryption randomness* to the ciphertext
- In ElGamal for instance
  - $Enc_Y(m) = (g^r, m \cdot Y^r, c, s)$ where $(c, s) = \boldsymbol{DLPRV}(r, g, g^r)$
  - Before decrypting check if $\boldsymbol{DLVF}\big(g, g^r, (c, s)\big) = 1$
  - Recall that $c = H(g, g^r, g^s(g^r)^{-c})$

Enc+PoK for non-malleability

- If wFS is used, the ciphertext is still malleable, despite the existence of the proof!

- Given $\boldsymbol{c_1} = (R, S, c, s)$
- Create $\boldsymbol{c_2} = (Rg^u, SY^u, c, s + cu), u \overset{\$}{\leftarrow} \mathbb{Z}_q$
- The proof still verifies:
  - $g^{(s+cu)}(Rg^u)^{-c} = g^s R^{-c}$
  - which is valid from the original proof (c, s)
  - But the ciphertext has changed!

Enc+PoK with Strong FS implies NM-CPA

NM-CPA Game

```
$(pk, sk) \leftarrow KGen(1^\lambda)$
$m_0, m_1 \leftarrow A(pk)$
$b \overset{\$}{\leftarrow} \{0,1\}$
$c^* \leftarrow Enc_{pk}(m_b)$
$\mathbf{c} \leftarrow A(pk, c^*)$ // parallel CCA
$for\ c_1 \in \mathbf{c}$:
  $if\ c_i = c^*\ return\ \perp$
  $else\ m_i \leftarrow Dec_{sk}(c_i)$
$b^* \leftarrow A(pk, c^*, \mathbf{m})$
$return\ b = b^*$
```

The contents of the vector $\boldsymbol{c}$ are created independently

Mihir Bellare and Amit Sahai, Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization, CRYPTO' 99

Breaking NM-CPA implies breaking IND-CPA

Hybrid 1:
Use the PoK Simulator when constructing c^*
If the adversary can distinguish the simulated proof then it can break ZK of PoK
Hybrid 2:
In order to construct $\mathbf{c}$, A makes random oracle queries for $\boldsymbol{DLPRV}$. The challenger uses the ZK extractor to retrieve the witness r for the proof and decrypt c_i without sk ($m_i = c_2/pk^{-r}$)
This is the IND-CPA Game

Bernhard, Pereira, Warinschi (2012) How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. *ASIACRYPT 2012*

Plaintext equivalence test (**PET**)

Do two ciphertexts c, c' encrypt the same plaintext?

$$\boldsymbol{c} = (c_1, c_2) = Enc_{pk}(m), \quad \boldsymbol{c'} = (c'_1, c'_2) = Enc_{pk}(m')$$

$$\boldsymbol{c_{PET}} = \frac{\boldsymbol{c}}{\boldsymbol{c'}}$$

$$\boldsymbol{c_{PET,i}} = \left(\frac{\boldsymbol{c}}{\boldsymbol{c'}}\right)^{z_i} \quad \pi_{i1} = EQPRV(z_i)$$

$$\phi = \prod \boldsymbol{c}_{PET,i} = (x, y)$$

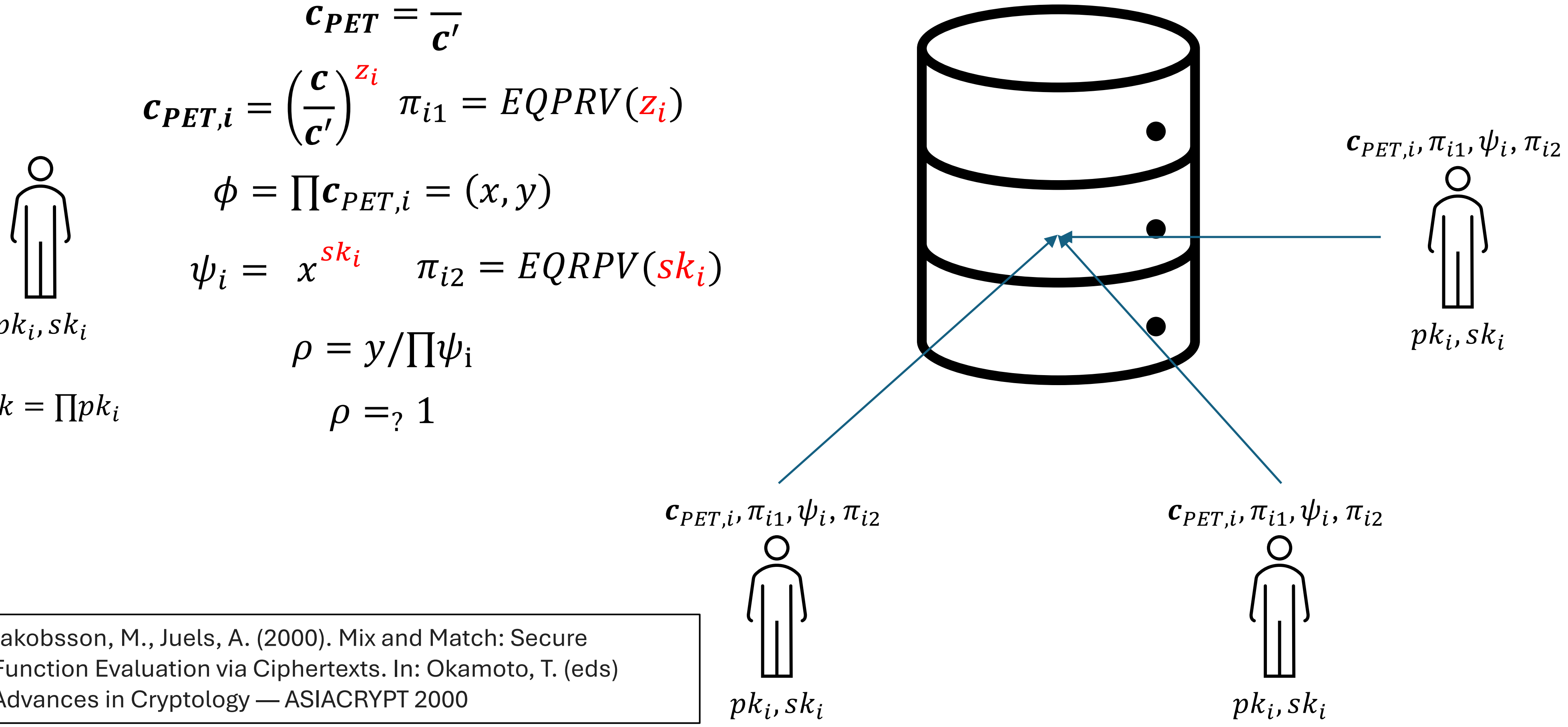
$$\psi_i = x^{sk_i} \quad \pi_{i2} = EQRPV(sk_i)$$

$$\rho = y / \prod \psi_i$$

$$\rho =_? 1$$

pk_i, sk_i

$pk = \prod pk_i$



$\boldsymbol{c}_{PET,i}, \pi_{i1}, \psi_i, \pi_{i2}$
pk_i, sk_i
$\boldsymbol{c}_{PET,i}, \pi_{i1}, \psi_i, \pi_{i2}$
pk_i, sk_i
$\boldsymbol{c}_{PET,i}, \pi_{i1}, \psi_i, \pi_{i2}$
pk_i, sk_i

Jakobsson, M., Juels, A. (2000). Mix and Match: Secure Function Evaluation via Ciphertexts. In: Okamoto, T. (eds) Advances in Cryptology — ASIACRYPT 2000

Blind Signatures

- A set of algorithms $\Pi = (KGen, Sign, Vf)$
- $(sk, vk) \leftarrow KGen(1^\lambda)$
- $\sigma \leftarrow \mathbf{Sign}\langle S(sk), U(m), vk\rangle$
    - **Sign** is a **protocol** and not an algorithm. It consists of:
        - $m' \leftarrow Blind(m, vk)$ initiated by the U
        - $\sigma' \leftarrow Sign(m', sk)$ initiated by the S - $\boldsymbol{Sign}$ is an algorithm
        - $\sigma \leftarrow Unblind(\sigma', vk)$ executed by the U
- Verification: $\{0,1\} \leftarrow Vf(m, \sigma, vk)$
- Correctness:
    - $Vf(m, \text{Sign}\langle S(sk), U(m), vk\rangle, vk) = 1\text{:}\ (sk, vk, prms) \leftarrow KGen(1^\lambda)$

Security Properties

- Blindness
  - The adversary is the signer
  - Goal: Indistinguishability of signatures-signing sessions
- Unforgeability
  - The adversary is the user
  - A blind signature is a forgery (created by the user not the signer)
  - One-more unforgeability
    - The user may not create more signatures than signing sessions

RSA Blind Signatures

- Key generation
  - Like in plain RSA. Finally: $(sk, vk) = (d, (e, n))$
- Signing:
  - $Blind(m, vk) \rightarrow H(m) \cdot r^e \textbf{ mod } n, r \leftarrow Z_n^*$
  - $Sign(m', sk) \rightarrow m'^d \textbf{mod } n \rightarrow \left(H(m)^d r\right) \textbf{ mod } n$
  - $UnBlind(\sigma', vk) \rightarrow \sigma' r^{-1} \textbf{mod } n \rightarrow H(m)^d \textbf{ mod } n$
- Verification:
  - The unblinded signature is a simple RSA signature

Schnorr Blind Signatures

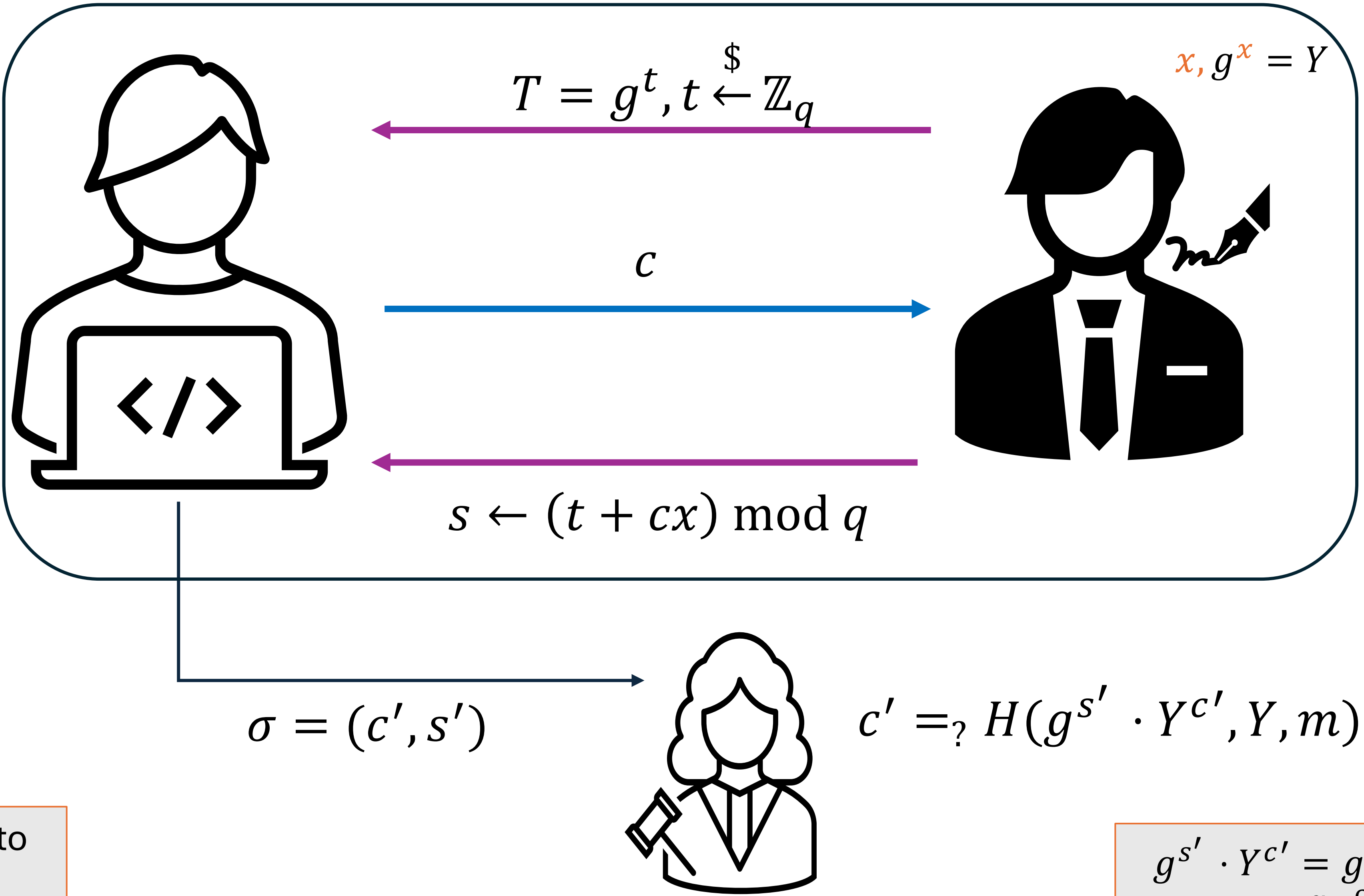
$$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$$

$$T' = T g^{\alpha} Y^{\beta}$$

$$c' \leftarrow H(\boldsymbol{T'}, Y, m)$$

$$c \leftarrow c' + \beta$$

$$s' \leftarrow s + a$$



$x, g^x = Y$
$T = g^t, t \xleftarrow{\$} \mathbb{Z}_q$
c
$s \leftarrow (t + cx) \bmod q$
$\sigma = (c', s')$
$c' =_? H(g^{s'} \cdot Y^{c'}, Y, m)$

α, β are blinding factors to alter (c, s) from (c', s')

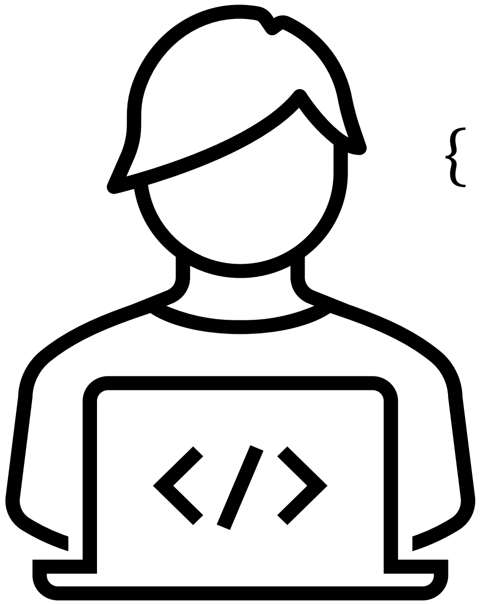
$$g^{s'} \cdot Y^{c'} = g^{s+a} \cdot Y^{\beta + c'} = T g^{\alpha} Y^{\beta} = T'$$

Group-Ring Signatures

- A set of algorithms $\Pi = (KGen, Sign, Vf)$
- $(sk, vk) \leftarrow KGen(1^\lambda)$
- $\sigma \leftarrow \mathbf{Sign}(sk, m, \boldsymbol{R})$
  - **R is a set of public keys**
  - sk corresponds to a $pk \in R$
- Verification: $\{0,1\} \leftarrow Vf(m, \sigma, \boldsymbol{R})$
- Basic idea:
  - Hide the signer inside a group of peers
  - Ring signatures: The group is **ad/hoc**
  - Group signatures: There is a group manager who might manages the group and might trace the signer

Ring Signatures via OR proofs

$R = \{Y_1 . Y_2, \ldots, Y_n\}$ $\quad PoK\{\mathbf{x}_k : g^{x_k} \in R : g^{x_k} = Y_1 \textbf{ OR } g^{x_k} = Y_2, \textbf{OR } \ldots, g^{x_k} = Y_n \}$



$$\mathrm{T}_\kappa = g^{t_\kappa},$$

$$\{\, \mathrm{T}_i = g^{t_i} Y_i^{-c_i}, t_i, c_i \overset{\$}{\leftarrow} \mathbb{Z}_q \} \quad i \in [n]/\{k\}$$

$$\longrightarrow$$

$$c \leftarrow H(R, \boldsymbol{m}, \{T_i\}_{i=1}^{n})$$

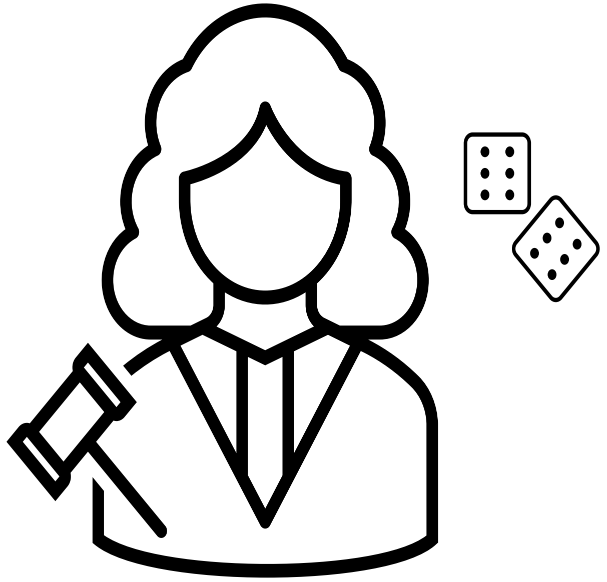
$$\longleftarrow$$

$$\longrightarrow$$

$$c_k \leftarrow c - \sum c_i$$

$$s_k \leftarrow (t_k + c_k x_k) \bmod q$$

$$s_i \leftarrow t_i, i \in [n]/\{k\}$$



$$\downarrow$$

$$\forall i \in [n]\ g^{s_i} =_? T_i Y_i^{c_i} \quad \text{και } c =_? \textstyle\sum_i c_i \bmod \mathrm{q}$$

Linkable Ring Signatures

- Two signatures by the same signer can be identified as such, but the signer remains anonymous.
- Enabled by including a linking tag in the signature
  - A function of the secret key
- Eliminate double voting
- Enable revoting

Liu, J.K., Wei, V.K., Wong, D.S. (2004). Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds) Information Security and Privacy. ACISP 2004. Lecture Notes in Computer Science, vol 3108. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-27800-9_28

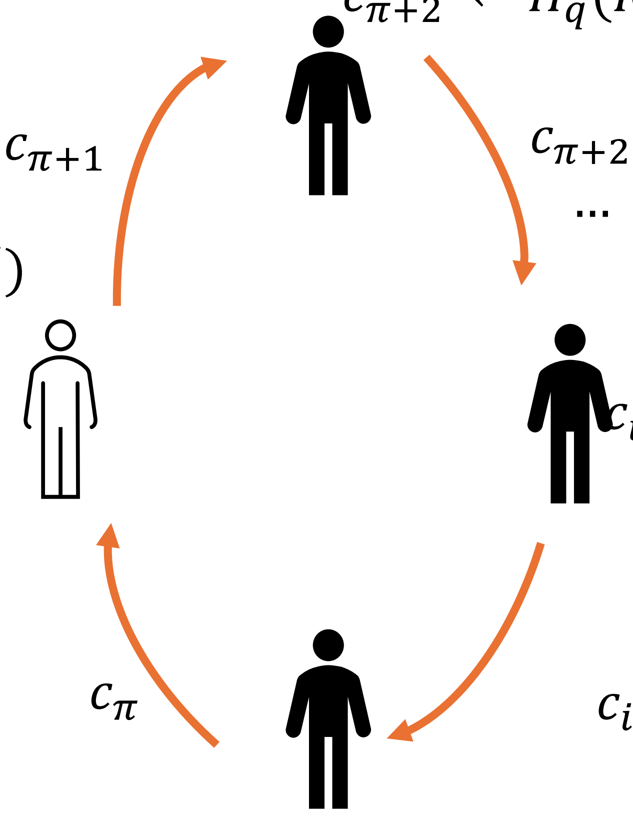
Linkable Ring Signatures

$R = \{Y_1, Y_2, \dots, Y_n\}$

$$s_{\pi+1} \overset{\$}{\leftarrow} \mathbb{Z}_q$$
$$c_{\pi+2} \leftarrow H_q(R, m, tag, g^{s_{\pi+1}} Y_i^{-c_{\pi+1}}, h^{s_{\pi+1}} tag^{-c_{\pi+1}})$$

$$t \overset{\$}{\leftarrow} \mathbb{Z}_q$$
$$c_{\pi+1} \leftarrow H_q(R, m, tag, g^t, h^t)$$

x_π, Y_π
$h \leftarrow H_{\mathbb{G}}(R)$
$tag \leftarrow h^{x_\pi}$



$$s_i \overset{\$}{\leftarrow} \mathbb{Z}_q$$
$$c_{i+1} \leftarrow H_q(R, m, tag, g^{s_i} Y_i^{-c_i}, h^{s_i} tag^{-c_i})$$

$$s_\pi \leftarrow t + c_\pi x_\pi$$
$$\sigma = (c_1, s_1, \cdots, s_n, tag)$$

Public Verification:
$\sigma = (c_1, s_1, \cdots, s_n, tag)$
Recompute $h \leftarrow H_{\mathbb{G}}(R)$
Compute for $i = 1 \dots n$
$c_{i+1} \leftarrow H_q(R, m, tag, g^{s_i} Y_i^{s_i}, h^{s_i} tag^{s_i})$
Check if
$$c_1 = c_{n+1}$$

$$s_{\pi-1} \overset{\$}{\leftarrow} \mathbb{Z}_q$$
$$c_\pi \leftarrow H_q(R, m, tag, g^{s_{\pi-1}} Y_i^{-c_{\pi-1}}, h^{s_{\pi-1}} tag^{-c_{\pi-1}})$$

Security Properties

- Unforgeability
  - No one can create a signature without knowing a secret key
- Anonymity
  - No-one can identify which ring member is the actual signer
- Linkability
  - Two signatures from the same signer are can be linked together
- Non-Slanderability
  - Given a signature no one can create a signature that links to it except for the original signer

Designated-Verifier Signatures

- Restrict the public verifiability of digital signatures
- Only a specific entity, *designated during signature creation,* can be sure about the signer of a message
    - Designation – inclusion of its public key in the signature algorithm
- Main idea:
    - Create an OR-proof of the statement:
        - **I know the signer's private key OR the designated verifier's private key**
    - The designated verifier can simulate proofs
        - Extra functionality **Simulate**
    - Other entities cannot be sure of the actual signer

Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. "Designated Verifier Proofs and Their Applications". In: *EUROCRYPT '96*

Designated-Verifier Signatures

- Strong DV Signatures
  - The private key of the DV is required for verification
- Use in e-voting:
  - Deniability for coercion-resistant and receipt-free schemes
  - The voters can simulate proofs that they followed the instructions of the coercer

- Security Properties
  - Unforgeability
    - Only the DV can ‘forge’ signatures
  - Non-Transferability
    - The public cannot tell if a signature originates from the signer or the DV

Designated-Verifier Schnorr

Proof

$PoK_V\{x : g^x = Y : Y, Y_V, g \in \mathbb{G}\}$

$T = g^t, t \xleftarrow{\$} \mathbb{Z}_q \quad Z = g^w Y_V^r, w, r \xleftarrow{\$} \mathbb{Z}_q$ $\longrightarrow$

$\longleftarrow$ $c \leftarrow H(Y, Y_V, T, Z)$

$s \leftarrow (t + (c + w)x) \bmod q$ $\longrightarrow$

$\downarrow$ (w, r, T, s)

$Z = g^w Y_V^r$

$c \leftarrow H(Y, Y_V, T, Z)$

$g^s =_? T Y^{c+w}$

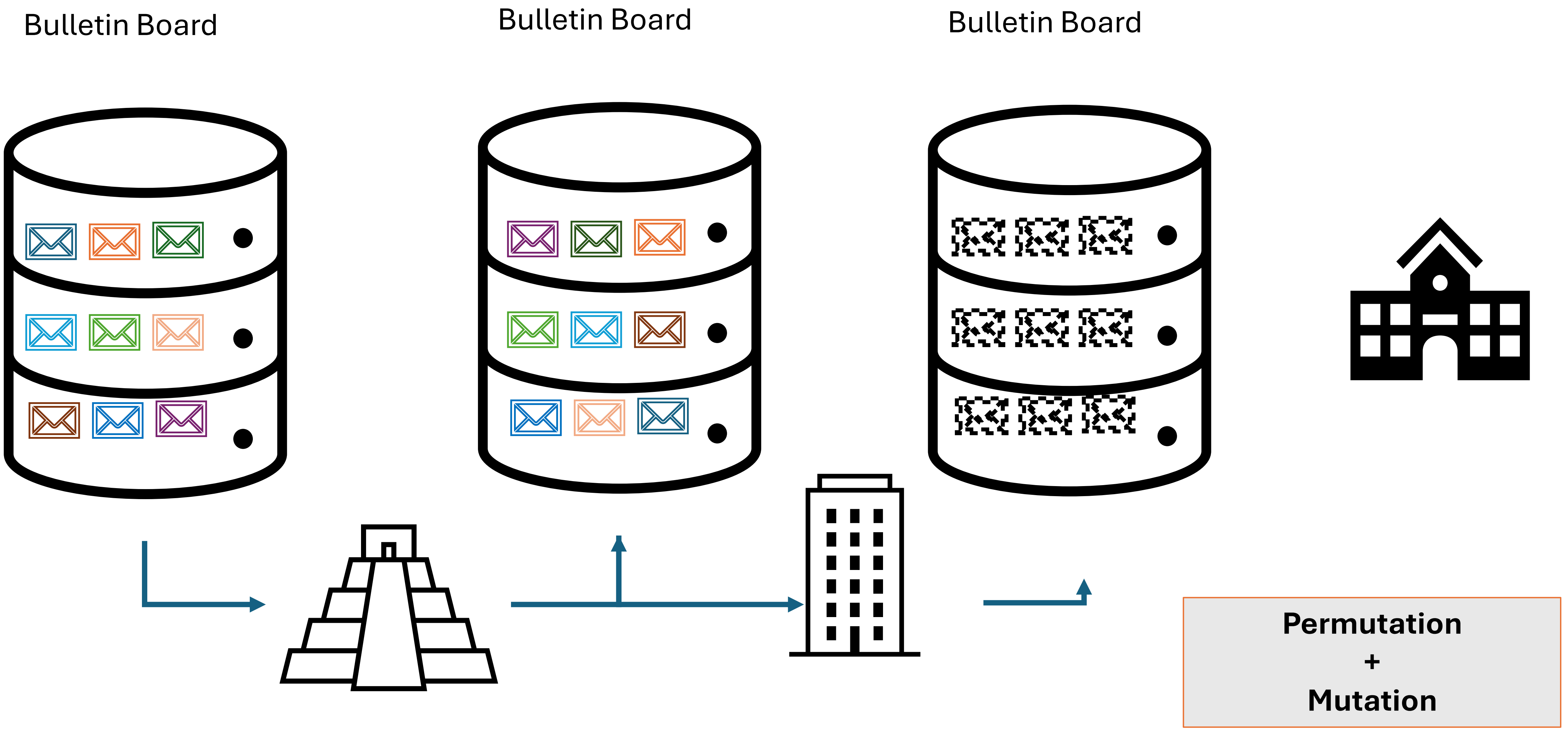
Simulation

$$T \leftarrow g^s Y^{-\beta}, \qquad Z \leftarrow g^a, \qquad a, \beta, s \xleftarrow{\$} \mathbb{Z}_q$$

$$c \leftarrow H(Y, Y_V, T, Z)$$
$$r \leftarrow (a - w) \cdot x_v^{-1}$$
$$w \leftarrow (\beta - c)$$

Return (w, r, T, s)

Mixnets



Permutation
+
Mutation

RSA Decryption Mixnets

- Each mixer i has a pair of RSA keys (sk_i, pk_i)
- The voter encrypts their choice using the public RSA keys of the mixers in reverse
  - $b_i = Enc_1(Enc_2(\ldots Enc_m(v_i) \ldots))$
  - $L_0 = (b_i)_{i=1}^{n}$
- Each mixer permutes the list of ballots using a random permutation π_i
- and decrypts using their private key (mutation)
- The first mixer will append to the BB:
  - $L_1 = \left(Dec_1(b_i)\right)_{\mathrm{i}=\pi_1^{-1}(1)}^{\pi_1^{-1}(n)}$

RSA Decryption Mixnets

- This process is repeated for every mixer
- In the end, the BB contains

$$L_m = (v_i)_{\mathrm{i}=\pi_m^{-1}\circ\cdots\circ\pi_1^{-1}(1)}^{\pi_m^{-1}\circ\cdots\circ\pi_1^{-1}(n)}$$

- Remarks:
  - The permutation could simply be to sort the encryptions as binary string
  - The last mixer knows the plaintext but not the voter identity
  - One honest mixer should be enough for security
    - Mixers should be entities with conflicting interests
  - Computationally expensive for the voter: $O(m)$ encryptions
  - Allows counting through the use of complex voting rules

ElGamal Decryption Mixnets

- Each mixer M_j has a key pair: $\left(sk_j, pk_j\right) = (x_j, g^{x_j})$
- The combined public key of the mixnet is $Y = \prod_j pk_j = g^{\sum x_j}$
- The voter encrypts their choice using Y
    - $b_{i0} = Enc_Y(v_i) = (g^{r_{i0}}, v_i Y^{r_{i0}})$
- Each M_j removes an encryption layer using their private key
    - $b_{ij} = Dec_{x_j}\left(b_{ij-1}\right)$
    - Applies new randomness r_{ij}
    - $L_j = \{b_{ij}\}_{i=1}^{n} = \left\{(g^{\sum_{k=0}^{j} r_{ik}}, v_i g^{\sum_{k=j+1}^{m} x_k \sum_{k=0}^{j} r_{ik}})\right\}_{i=1}^{n}$
    - Permutes using π_j

ElGamal Reencryption Mixnets

- Each mixer M_j reencrypts and permutes the ballot list using Y
- On input $L_{j-1} = \{Enc_Y(v_i, r_i)\}_{i=1}^{n}$
- Selects $\left\{r_{ij} \overset{\$}{\leftarrow} \mathbb{Z}_q\right\}_{i=1}^{n}$
- Computes
  - $L_j = \left\{Enc_Y\left(v_i, r_{ij}\right) \cdot Enc_Y\left(1, r_{ij}\right)\right\}_{i=1}^{n} = \left\{\left(g^{\sum_{k=0}^{j} r_{ik}}, v_i Y^{\sum_{k=0}^{j} r_{ik}}\right)\right\}_{i=1}^{n}$
- Permutes using π_j
- All mixers jointly decrypt after L_m has been posted

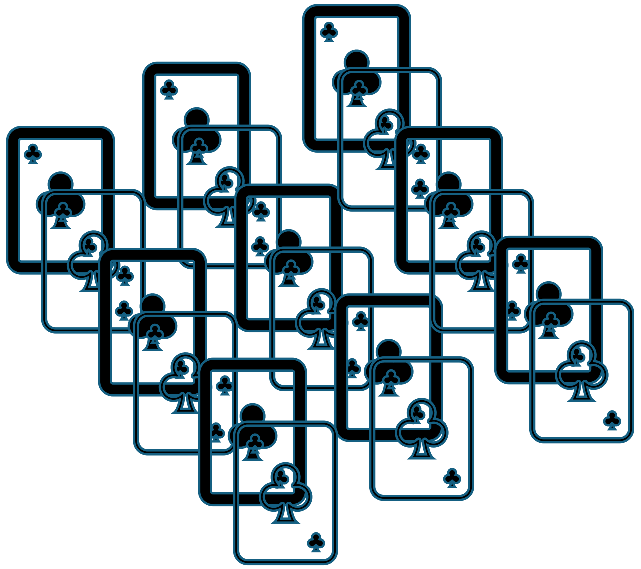
The tagging attack

- A generic attack applicable to all types of anonymous channels!
- Adversarial goal: reveal the input of V_i with the help of a corrupted user V_j willing to sacrifice their input
- The adversary
  - Retrieves the initial input of V_i: $c_{i0} = (g^r, v_i Y^r)$
  - Selects $\tau \overset{\$}{\leftarrow} \mathbb{Z}_q$ and computes $c_{i0}^{\tau} = (g^{r\tau}, v_i^{\tau} Y^{r\tau})$
  - Replaces V_j's input with c_{i0}^{τ}
- The output of the mixnet contains both v_i, v_i^{τ}
- The adversary computes for all outputs $x \rightarrow x^{\tau}$ and checks for duplicates

Pfitzmann, B.: Breaking an efficient anonymous channel. In: EUROCRYPT'94. pp. 332–340 (1995)

Verifiable mixnets – Proofs of Shuffles

- Protect against corrupted mixers that aim to omit or alter inputs
- The mixer provides a proof of (correct) shuffle that:
  - No plaintexts were *modified*
  - No ciphertexts were *removed* or *inserted*
  - The output ciphertexts are only a reencryption and permutation of the input ciphertexts.
- Without revealing:
  - The permutation π
  - The reencryption factors r_i
- Many solutions in the literature



A simple 2×2 verifiable shuffle

- Input
  - $c_0 = Enc_Y(m_0, r_0), c_1 = Enc_Y(m_1, r_1)$
- Output
  - $c_0' = ReEnc(c_b) = Enc_Y(m_b, r_b{}'), c_1' = ReEnc(c_{1-b}) = Enc_Y(m_{1-b}, r_{1-b}{}')$
- Proof that $c_i' = ReEnc(c_i)$
  - Prove that they encrypt the same message
  - If $c_i = (G, mR)$ then $c_i{}' = (G', mR')$
  - This means that $DL_g(G \cdot G'^{-1}) = DL_Y(R \cdot R'^{-1})$
  - Use the Chaum – Pedersen Protocol
- Proof of correct shuffle
  - Prove that $\{c_0', c_1'\}$ is a shuffle of $\{c_0, c_1\}$
  - Prove that $c_i' = ReEnc(c_i)\ AND\ c_{1-i}' = ReEnc(c_{1-i})\ OR\ c_i' = ReEnc(c_{1-i})\ AND\ c_{i-1}' = ReEnc(c_i)$
  - Composition of Chaum – Pedersen Protocols

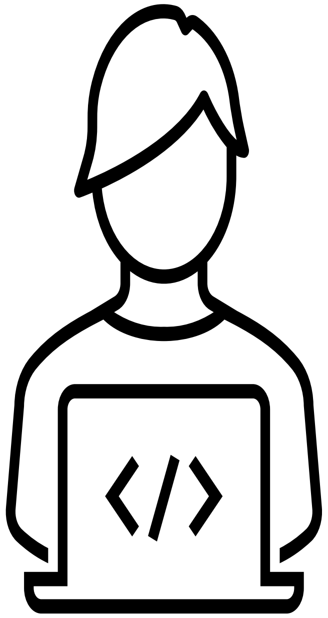
Bayer – Groth Proof of Shuffle

- Public Input
  - Two sets of ciphertexts $C_1, \ldots, C_n$ and $C_1', \ldots, C_n{}'$ in a group $\mathbb{G}$ of prime order q
  - Encrypted with pk
- Private input $\pi, \boldsymbol{\rho} = (\rho_1, \ldots, \rho_n)$ such that
  - $C_i' = C_{\pi(i)} \cdot Enc_{pk}(1, \rho_i)$
- Proof of Knowledge of Permutation
  - Product Argument: A set of committed values has a particular product
- Proof of Knowledge of Reencryption Factors
  - Mult exponentiation argument: The product of a set of ciphertexts raised to a set of committed exponents yields a particular ciphertext

Bayer, S., Groth, J. (2012). Efficient Zero-Knowledge Argument for Correctness of a Shuffle–EUROCRYPT 2012

Bayer – Groth Proof of Shuffle

9 round HVZK argument



$$\xrightarrow{\quad Commit(\pi(1), \dots, \pi(n)) \quad}$$

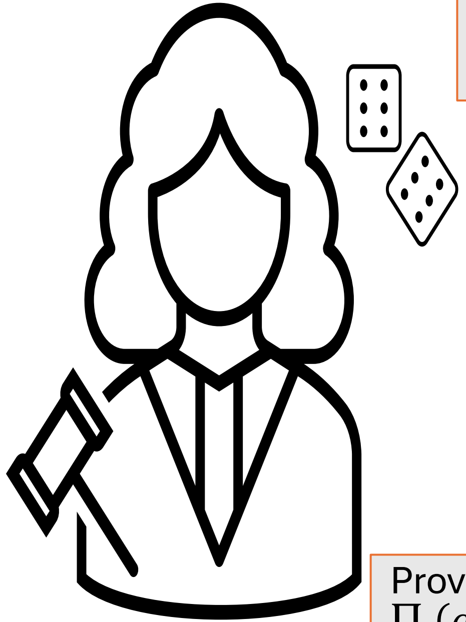
$$\xleftarrow{\quad x \overset{\$}{\leftarrow} \mathbb{Z}_q \quad}$$

$$\xrightarrow{\quad Commit(x^{\pi(1)}, \dots, x^{\pi(n)}) \quad}$$

$$\xleftarrow{\quad c, z \overset{\$}{\leftarrow} \mathbb{Z}_q \quad}$$

Prove that $\prod_i (d_i - z) = \prod_i (x^i + ic - z)$ using the product argument $\longrightarrow$

Prove that $Enc_{pk}(1, r) \; \prod_i C'^{x^{\pi(i)}}_i = \prod_i C_i^{x^i}$ using the multiexponentiation argument $\longrightarrow$



The prover must convince the verifier that the same permutation has been used for $1, \dots, n$ and $x^1, \dots, x^n$

Prove that the $P(k) = \prod_i (d_i - k) - \prod_i (x^i + ic - k)$ is the zero polynomial

Schwartz-Zippel lemma: This can be cheated with negligible probability if the permutation is not known

Bayer – Groth Proof of Shuffle

- State of the art in proof size $O(\sqrt{n})$
- Verification time $O(n)$
- Prover time $O(log(\sqrt{n})n)$
- Main trick for efficient communication complexity:
  - Arrange the input ciphertexts into a $k \cdot l$ matrix where $k = O(\sqrt{n})$
  - Use Generalised Pedersen Commitment to commit to columns
- First prover message
  - Send $\mathbf{cm}_{\Pi} = GPC(\boldsymbol{\pi}_{\mathbf{k}}, \mathbf{r})$ were $\mathbf{r} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$ and $\mathrm{U}_k\, \boldsymbol{\pi}_{\mathbf{k}} = \boldsymbol{\pi}$
- Second prover message
  - Send $\mathbf{cm}_{\mathbf{X}} = GPC(\mathbf{x}^{\boldsymbol{\pi}_{\mathbf{k}}}, \mathbf{s})$ were $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^k$ and $\mathrm{U}_k\, \boldsymbol{\pi}_{\mathbf{k}} = \boldsymbol{\pi}$
  - The permutation was fixed before the prover saw x

- **Generalized Pedersen Commitment**
- Commitment to a vector $\mathbf{m} = (m_1, \dots, m_n)$
- $\mathbb{G}$ is a cyclic group of prime order q generated by $g_1, \dots, g_n, h$
  - $GPC(\mathbf{m}, r) = h^r \prod_i g_i^{m_i}$

Bayer, S., Groth, J. (2012). Efficient Zero-Knowledge Argument for Correctness of a Shuffle–EUROCRYPT 2012

Bayer – Groth Proof of Shuffle

- Third message: Both prover and verifier compute
  - $\mathbf{cm}_{-z} = \mathrm{GPC}(-\mathbf{z}, \mathbf{0})$
  - $\mathbf{cm_D} = \mathbf{cm}_{\Pi}^{c} \otimes \mathbf{cm_X} = \mathrm{GPC}(c \cdot \boldsymbol{\pi}(\mathbf{i}) + \mathbf{x}^{\boldsymbol{\pi}(\mathbf{i})})$ which is a commitment to $c\pi(i) + x^{\pi(i)}$ with randomness $cr_i + s_{\mathrm{i}}$
  - The verifier does not know $\pi(i), r_i, s_{\mathrm{i}}$ but can compute the values homomorphically
  - $\mathbf{cm}_{\boldsymbol{D}} \otimes \mathbf{cm}_{-\boldsymbol{z}} = \mathrm{GPC}(\mathbf{d} - \mathbf{z})$ where $d_i = c\ \pi(i) + x^{\pi(i)}$
  - Use the product argument to show knowledge of d_i, r_i, s_i such that:
    - $\prod_i (d_i - z) = \prod_i (x^i + ic - z)$ a polynomial and its permutation in z - identical roots
    - The value $\prod_i (x^i + ic - z)$ can be computed by the verifier

Bayer, S., Groth, J. (2012). Efficient Zero-Knowledge Argument for Correctness of a Shuffle–EUROCRYPT 2012

Bayer – Groth Proof of Shuffle

- Third message: The prover computes
    - $\rho \leftarrow \boldsymbol{\rho} \odot \mathbf{s}$
    - $\mathbf{C}^{\mathbf{x}} = Enc(1, \rho) \cdot \mathbf{C}'^{\mathbf{x}_\pi}$ where $\mathbf{x} = (x^1, \ldots, x^n)$
    - The verifier can compute $\mathbf{C}^{\mathbf{x}}$
    - Using the multi exponentiation argument it convinces the verifier that $Enc(1, \rho) \cdot \mathbf{C}'^{s}$ was computed correctly
- Note that because of the homomorphic properties
    - $\prod_i m_i^{x^i} = \prod_i m_i'^{x^i} \Rightarrow \log \sum (m_i)\, x^i = \log \sum \left(m_{\pi^{-1}(i)}{}'\right) x^i$
    - This means that **wvhp** $m_{\pi(i)} = m_i{}'$
- The shuffle was performed correctly

Voting Paradigms

Helios and extensions

JCJ – Coercion Resistance

Voting with blind/ring signatures

OpenVote

Helios

Helios' Facts

- Elections in the browser
  - Open-Audit: Everyone has access to all election data for verifiability
  - **Trust no one for integrity – trust the server for privacy**
  - Low coercion environments
- 2.000.000 votes cast so far
  - ACM, IACR and university elections
  - Can be used online https://vote.heliosvoting.org/ or deployed locally
- Based on:
  - Verifiable mixnets – Helios 1.0 (Sako-Killian, Eurocrypt 95)
  - **Homomorphic tallying – Helios 2.0 (Cramer-Genaro-Shoenmakers, Eurocrypt 97)**
  - **Benaloh Challenge**
- Many variations
  - Belenios (Helios-C)
  - Zeus

Ben Adida. 2008. Helios: web-based open-audit voting. In Proceedings of the 17th conference on Security symposium (SS'08). USENIX Association, USA, 335–348.

Participants

- **Election administrator:** Create the election, add the questions, combine partial tallies
- **BB - Bulletin' Board**: Maintain votes (**Ballot Tracking Center**) and audit data
- **TA - Trustees (Talliers)**: Partially decrypt individual (in Helios 1.0) or aggregated (in Helios 2.0) ballots
- **RA - Registrars (Helios-C):** Generate cryptographic credentials for voters
- $EA = (RA, TA, BB)$
- **Eligible voters** optionally identified by random alias or external authentication service (Google, Facebook, LDAP)
  - Authenticated channel between voter and BB (username, password)

Auditing Process

- Individual Verifiability
  - Cast as intended
    - After ballot creation (encryption) but before authentication, each voter can choose if they will audit or cast the ballot.
    - **On audit:** Helios releases the encryption randomness and the voter can recreate the ballot using software of their choice.
    - An audited ballot cannot be submitted.
  - Recorded as cast
    - Each encrypted ballot and related data are hashed to a tracking number.
    - Every voter can check if the assigned number exists in the Ballot Tracking Center (BTC).

Auditing Process

- Universal Verifiability
  - Tallied as recorded - Every interested party may
    - Retrieve ballots from BTC
    - Compare identities with eligible voters (if applicable)
    - Recompute tracking numbers
    - Aggregate the ballots and check equality with official encrypted tally before decryption
  - Verify decryption proofs

Formal Description: Setup

- Executed by the Election Administrator
- Creates cryptographic groups, defines message space etc.
- Reusable for many elections

$$Setup(1^\lambda) = \left\{ \begin{array}{c} \mathbb{G}, q, g \\ H_q : \{0,1\} \to \mathbb{Z}_q \\ (\boldsymbol{DLPRV}(x, g, Y)\,, \boldsymbol{DLVF}(g, Y, \pi)\,) \\ (\boldsymbol{EQPRV}(x, g_1, Y_1, g_2, Y_2), \boldsymbol{EQVF}(g_1, Y_1, g_2, Y_2, \pi)) \\ (\boldsymbol{DJPRV}(x_1, x_2, g, Y_1, Y_2), \boldsymbol{DJVF}(g, Y_1, Y_2, \pi)) \\ BB \leftarrow \emptyset \end{array} \right.$$

Formal Description: SetupElection

- The members of the TA cooperate to create their **joint** public key
  - Compute member key pair: $sk_i \overset{\$}{\leftarrow} \mathbb{Z}_q, pk_i \leftarrow g^{\mathrm{sk}_i}$
  - Publish $pk_i, DLPRV(sk_i, g, pk_i)$
  - Compute election public key: $pk \leftarrow \prod_i pk_i$
- Create list of eligible voters V_l
- Create list of candidates $CS = \{0,1\}$ (for simplicity)
- Publish everything into BB
  - $BB \Leftarrow \{pk_i, pk, V_l, CS\}$

Formal Description: Voting

Vote(i,v):

$v \in \{g^0, g^1\}$

$Enc_{\mathrm{pk}}(g^v) \to (g^r, g^v \cdot \mathrm{pk}^r) = (R, S)$

$EQPRV(r, g, R, pk, S)\ OR\ EQPRV(r, g, R, pk, Sg^{-1}) \to \pi_V$

$\mathrm{b} = (R, S, \pi_V)$

Valid(i,b):

Return 1 if $i \in V_l$ and $EQVF(\pi_V) = 1$

Append(I,b):

$BB \leftarrow (i, b)$ if $Valid(b) = 1$

VerifyVote(i,b,BB):

Return 1 if $b \in BB$ **and** $Valid(i, b) = 1$

Publish(BB):

Return $\mathrm{PBB} = \{b\}$ i.e. remove id’s from ballots and keep one ballot per voter id

Occurs after all voters have voted

Formal Description: Tally

```
Tally(PBB, $sk_i$):
    Validate all proofs in $PBB$
    Compute $(R_\Sigma, S_\Sigma) \leftarrow \prod b$ for all $b \in PBB$
    Distributed Decryption of $(R_\Sigma, S_\Sigma) \rightarrow g^t$
    Each $TA_i$
        posts $\left(D_i = R_\Sigma^{sk_i}, EQPRV(sk_i, g, pk_i, R_\Sigma, D_i)\right)$
        computes $\frac{S_\Sigma}{\prod_i D_i} \rightarrow g^t$
        solves small DLOG to get $t$
        posts $\pi_T = EQPRV(sk_i, g, pk_i, R_\Sigma, S_\Sigma \cdot g^{-t})$
```

Formal Description: Verify

Verify(BB,PBB, t, π_T):

Check correct construction of PBB

- Only last ballot kept
- All kept ballots belong to eligible voters
- All kept ballots had valid proofs

Recompute $(R_\Sigma, S_\Sigma) \leftarrow \prod b$ for all $b \in PBB$

Verify π_T

Attacks by using wFS: Denial of Service

- In the proof $EQPRV(sk_i, g, pk_i, R_\Sigma, D_i)$ a malicious TA_i can cheat by **first creating the proof** and **then adaptively selecting D_i**
  - Compute $T_1 \leftarrow g^a, T_2 \leftarrow g^b$ where $a, b \overset{\$}{\leftarrow} \mathbb{Z}_q$
  - wFS: $c \leftarrow H(T_1, T_2)$
  - Compute $s \leftarrow a + c \cdot sk_i$
  - Select $D_i \leftarrow (R_\Sigma^{-s} T_2)^{-c^{-1}}$
- The proof (c, s) verifies
  - $g^s pk_i^{-c} = T_1$ and $R_\Sigma^s D_i^{-c} = R_\Sigma^s R^{-s} T_2 = T_2$ but $\log_{R_\Sigma} D_i = -s - c^{-1} \log_{R_\Sigma} T_2 \quad \neq sk_i$
- What does this mean?
  - Tally decryption will yield a random group element instead of g^t
  - Efficient computation of t (assumed to be small DLOG) will not be feasible!

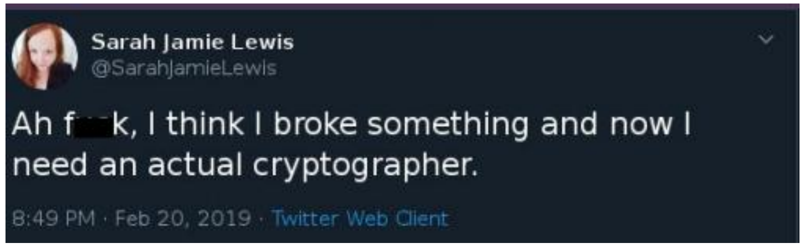
Attacks by using wFS: Undetectably alter result

- Goal: Announce election result $t \neq t'$
- Assumptions
  1. All TA_i's are corrupted – corrupted TA
  2. The TA can eavesdrop on the voter-selected encryption randomness
     - Realistic assumption if the voting device is corrupt
  3. Corrupt a single voter to cast the last vote
- The TA creates a 'proof' of correct 'tallying' before tallying
  1. Compute $T_1 \leftarrow g^a, T_2 \leftarrow g^b$ where $a, b \overset{\$}{\leftarrow} \mathbb{Z}_q$
  2. wFS: $c \leftarrow H(T_1, T_2)$
  3. Compute $s \leftarrow a + c \cdot sk$
- All voters vote except for the corrupt voter
  1. The current result is t and encrypted as $(R, S) = (g^{\Sigma r}, g^t pk^{\Sigma r})$
  2. By assumption 2: $\sum r$ is know to the TA
  3. The TA can compute t before the corrupt voter

Attacks by using wFS: Undetectably alter result

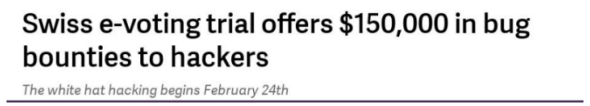
- The TA selects $r' \leftarrow \frac{b+c(t-t')}{s-c\cdot sk}$
- Using the corrupt voter the TA casts the ballot $(g^{r'-\Sigma r}, g^0 pk^{r'-\Sigma r})$ which is a valid ballot
- The current encrypted tally is $(R', S') = (g^{r'}, g^t \cdot pk^{r'})$
- The encrypted tally does not change **but the proof $(\boldsymbol{c}, \boldsymbol{s})$ also verifies for $\boldsymbol{t'}$**
- $g^s pk^{-c} = T_1$ **(nothing has changed here)**
- $R'^s (S' \cdot g^{-t'})^{-c} = g^{sr'-ct-c\cdot sk\cdot r'+ct'} = g^{r'(s-c\cdot sk)-c(t-t')} = g^b = T_2$
- As a result, the corrupt TA can announce t' for the election result and everyone will be convinced by the proof.

NSW Electoral Commission iVote and Swiss Post e-voting


Sarah Jamie Lewis
@SarahJamieLewis

Ah f[redacted]k, I think I broke something and now I need an actual cryptographer.

8:49 PM · Feb 20, 2019 · Twitter Web Client


Swiss e-voting trial offers $150,000 in bug bounties to hackers

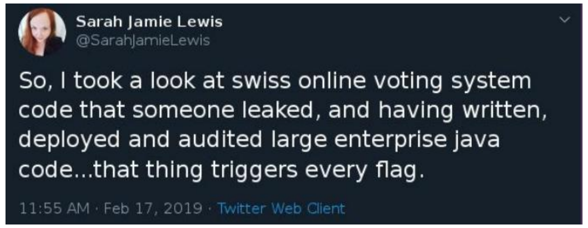
The white hat hacking begins February 24th

Similar attacks to other voting schemes

- S. J. Lewis, O. Pereira, and V. Teague, “How not to prove your election outcome: The use of non-adaptive zero knowledge proofs in the Scytl-SwissPost Internet voting system, and its implications for decryption proof soundness”
- R. Haenni, “Swiss post public intrusion test: Undetectable attack against vote integrity and secrecy”

We broke it too

Feb 20, 2019, 8:59 PM


Sarah Jamie Lewis
@SarahJamieLewis

So, I took a look at swiss online voting system code that someone leaked, and having written, deployed and audited large enterprise java code...that thing triggers every flag.

11:55 AM · Feb 17, 2019 · Twitter Web Client

Helios Extensions

Everlasting Privacy

Receipt Freeness

Eligibility Verifiability

Everlasting privacy

- Ballot secrecy is provided through encryption schemes
- Protection relies on computational hardness assumptions
- What if these assumptions are broken?

- Vote contents might be useful to a future oppressive government
- But such a regime might also use insider information
- This threat might constitute an indirect coercion attempt
- The need for verifiability makes election data publicly available

- Helios does not have everlasting privacy!
- The functionality **Publish(BB)** releases the encrypted ballots
- An unbounded adversary can decrypt them!

Approaches to everlasting privacy

- **Perfectly Hiding Commitments**
  - Instead of encryption
  - But: Counting requires the openings.
  - How do voters send them?
    - Through Private Channels
      - Encrypted
      - Directly sent to the authorities
    - Not available to a future attacker
      - Unless they control part of the authorities
    - **Practical Everlasting Privacy**

- **Anonymous casting**
  - Disassociate identity from ballot
  - Use anonymous credentials to signal ballot eligibility or validity
    - Blind signatures
    - Ring signatures
  - An important advantage:
    - No trust required for privacy!

- Haines, T., Mueller, J., Mosaheb, R., & Pryvalov, I. (2023). SoK: Secure E-Voting with Everlasting Privacy. In Proceedings on Privacy Enhancing Technologies (PoPETs).
- Grontas, P., Pagourtzis, A. Anonymity and everlasting privacy in electronic voting. Int. J. Inf. Secur. 22, 819–832 (2023). https://doi.org/10.1007/s10207-023-00666-2

Adding everlasting privacy to Helios

Voters:

- Instead of encryption, use committments
  - $v \in \{0,1\}$,
  - $c = Commit(v, s) \rightarrow (g^{v} \cdot h^{s})$
  - $c_1 = Enc_{pk}(v) \rightarrow (g^{r_1}, g^{v} pk^{r_1})$
  - $c_2 = Enc_{pk}(s) \rightarrow (g^{r_2}, g^{s} pk^{r_2})$
- Proof of validity of v
- Proof that v, s are the same in c, c_1, c_2
- Post c in BB
- Send c_1, c_2 to TA through **private channels**

Talliers:

- Compute
  - $\prod_{v \in V} c$. Yields $\mathbf{c} = Commit(\sum v, \sum s)$
  - $\prod_{v \in V} c_1$. Yields $\mathbf{c_1} = Enc_{pk}(\sum v)$
  - $\prod_{v \in V} c_2$. Yields $\mathbf{c_2} = Enc_{pk}(\sum s)$
- Posts decryptions of $\boldsymbol{c_1}, \boldsymbol{c_2}$
- Everyone can validate the commitment $\mathbf{c}$

Do you see a problem?

Denise Demirel, J Van De Graaf, and R Araújo. "Improving Helios with Everlasting Privacy Towards the Public". In: *EVT/WOTE'12 Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections* (2012).

Adding everlasting privacy to Helios

- $Enc_{pk}(\sum s) = (g^{\sum r_2}, g^{\sum s} pk^{\sum r_2})$
- Need to solve DLP to get $\sum s$.
- This is not feasible!
  - Randomness is not in the same range as the result
- **Solution:**
  - **Use Paillier cryptosystem**
  - Encryption in the exponent
  - DLP for free!

Denise Demirel, J Van De Graaf, and R Araújo. “Improving Helios with Everlasting Privacy Towards the Public”. In: *EVT/WOTE’12 Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections* (2012).

Josh Benaloh and Dwight Tuinstra. “Receipt-free secret-ballot elections (extended abstract)”. In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing - STOC ’94*. ACM Press, 1994, pages 544– 553.

Receipt-Freeness

- Extensions for privacy against malicious voters
  - Voters that wish to sell their vote
- The attack scenario:
  - A voter agrees to sell their vote before the election
  - Proceeds to vote on their own
  - **The buyer does not monitor the voter when casting the ballot**
  - The voter presents evidence *after* voting to receive payment

A voting system is receipt free if a malicious voter cannot prove how they voted even if the want to

Helios is not receipt-free

- The malicious voter will offer as evidence:
  - the encryption randomness r
  - the position of the claimed ballot b in the BB
- The buyer will:
  - Encrypt the claimed choice with r
  - Compare with b
- Revoting does not help against coercion resistance
  - The published BB contains the final version of b

Adding receipt - freeness

- **Main idea:** The voter is not the sole contributor of encryption randomness for the ballot
  - They do not know the final randomness used - the voter – generated randomness as receipt is spoiled!
- A rerandomization authority reencrypts the ballot
  - Trusted for receipt-freeness
  - Not trusted for integrity/verifiability and privacy
- Sends a proof of correct reencryption to the voter
  - Use of designated verifier proofs
  - The voter (DV) cannot use it to convince the voter buyer

Martin Hirt and Kazue Sako. “Efficient receipt-free voting based on homomorphic encryption”. In. EUROCRYPT’00

Adding receipt – freeness

- Each voter has a private-public key pair (sk_V, pk_V).
- They encrypt their ballot deterministically (i.e. $r = 0$) and send it to the EA
- The EA is split into $EA_1, \cdots, EA_n$ which operate a verifiable mixnet
  - Each vote is shuffled and reencrypted
  - Public proof of correct shuffling
- Each authority privately proves to each voter how the list was shuffled and reencrypted
  - The proof uses pk_V so it is designated-verifier
  - The voter can pinpoint their ballot in their final list to verify it, but they cannot prove to a vote seller its position
    - Non-transferability

Martin Hirt and Kazue Sako. “Efficient receipt-free voting based on homomorphic encryption”. In. EUROCRYPT’00

Eligibility verifiability

- Anyone can verify that:
  - Every ballot was cast by a voter with the right to vote
  - No voter cast more than two counted ballots
  - Prevent **ballot stuffing**
- A simple solution:
  - Equip voters with credentials (PKI)
  - Sign encrypted ballots
  - Keep only one ballot / public key
  - Verify against eligible voter list

Belenios: Helios with credentials

- Extension to provide eligibility verifiability
- Adds a registration (credential) authority
- The BB generates login information for the voters (username, password)
- The voters receive both credentials $\langle (pk_i, sk_i), (uid, pwd) \rangle$using a private channel
- The voter logins to the BB using (uid, pwd)
- The ballot consists of
  - Vote encryption c
  - NIZK proof π of vote validity
  - A signature on c

Belenios: A Simple Private and Verifiable Electronic Voting System. Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. In Foundations of Security, Protocols, and Equational Reasoning, pp. 214-238, 2019.

Belenios: Helios with credentials

- The BB keeps one ballot per (id, pk)
  - Last one if multiple exist
- The BB checks signatures and proofs
- The voters check that their ballots appear on the BB (individual verifiability)
- Ballot stuffing can occur only if both the BB and the RA are corrupt
  - Stuffed ballots need to have both a vk and an id
- Eligibility verifiability:
  - Everyone can check that a ballot comes from a valid voter
  - But: This reveals who abstained - illegal in some countries

Belenios: A Simple Private and Verifiable Electronic Voting System. Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. In Foundations of Security, Protocols, and Equational Reasoning, pp. 214-238, 2019.

Private eligibility verifiability (KTV-Helios)

- Participation privacy + Universal verifiability
- **Main idea: Add null votes + vote update capabilities**
- Voting proxies:
  - Entities that add null votes for a voter
- Properties of null votes:
    - They do not add to the result
    - They are indistinguishable from regular votes
    - Proofs that each vote is either a null vote or a normal vote
    - Anonymous casting
- Also provide (some degree) of receipt freeness
  - The voter may prove that he cast c, but..
  - If there exists another ballot c' cast for them, they cannot prove that
    - $c' \neq c'' \cdot c^{-1}$ (which updates their true ballot to c'')

Kulyk, O., Teague, V., Volkamer, M. Extending Helios Towards Private Eligibility Verifiability. Vote-ID 2015.

BeleniosRF: Belenios with receipt-freeness

- Use a rerandomizing server
  - Rerandomizes all the ballots before publishing them to the BB
  - This breaks the validity of signatures!
- Solution: Signatures on Randomizable Ciphertexts
  - Given a ciphertext, signature pair (c, σ)
    - Rerandomize the ciphertext to c'
      - Without the decryption key
    - Adapt the signature so that it publicly verifies for c'
      - Without the signing key

Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: A non-interactive receipt-free electronic voting scheme. In *23rd ACM Conference on Computer and Communications Security (CCS'16)*, pages 1614–1625, Vienna, Austria, 2016.

BeleniosRF: Belenios with receipt-freeness

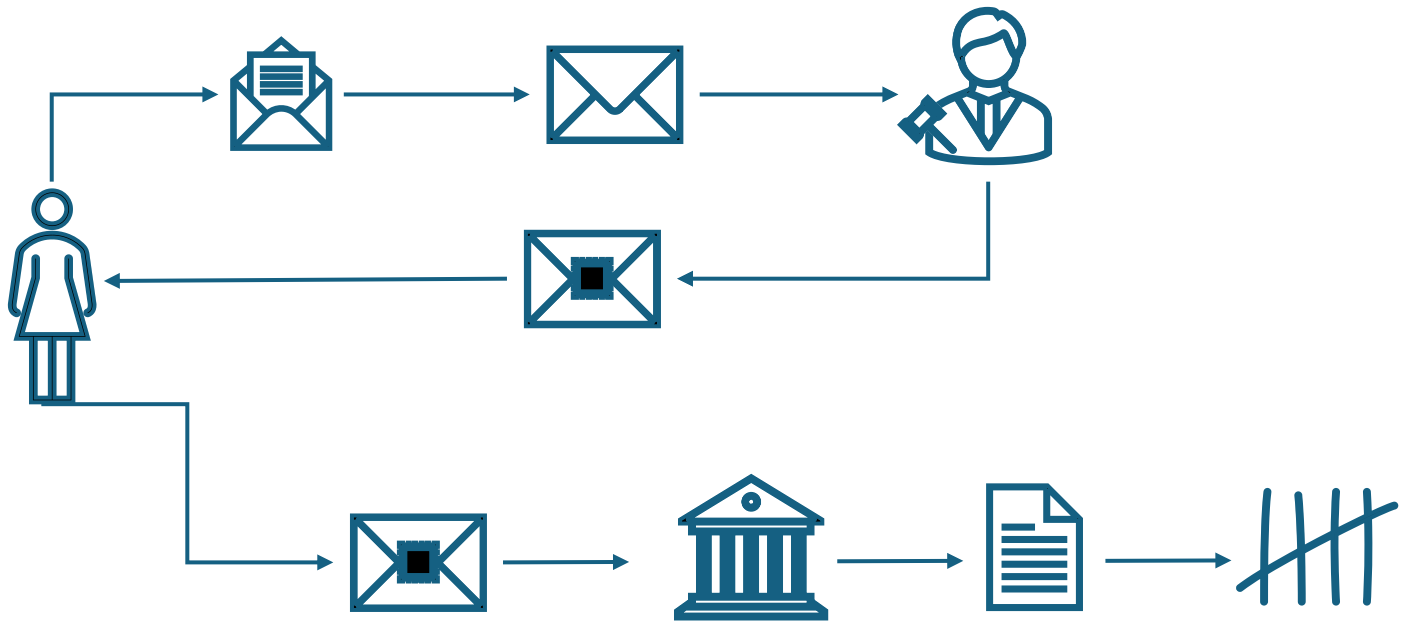
- No need for proofs of correct rerandomization for RF
  - The EA rerandomizes the ciphertexts and adapts the signatures of validity
- Security:
  - Rerandomization appears as fresh encryption
  - One-more unforgeability: The signer can create signatures on *messages* they have never seen
- Is it enough?
  - The voter might sell their keys and passwords!

Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In *Public Key Cryptography - PKC 2011*

Voting With Blind Signatures

A practical secret voting scheme for large scale elections

- **Main idea:** How would real-world elections work if the identity validation took place in a different physical space than counting?



Assumptions:
- Voters have cryptographic key pairs
- Voters can send two messages
- Access to an anonymous channel

Fujioko A, Okamoto T, Ohta T (1992) A practical secret voting scheme for large-scale elections. In: Proceedings of advances in cryptology, AUSCRYPT’92, Springer, pp 244–260

A practical secret voting scheme for large scale elections (FOO)

- Preparation $\mathrm{V_i}$
  - Select and commit to the vote
    - $b_i = Commit(v_i, r_i)$
  - Blind ballot for pk_{EA}
    - $bb_i = Blind(b_i, pk_{EA})$
  - Sign with voter sk_{v_i}
    - $sbb_i = Sign(sk_{v_i}, bb_i)$
  - Send to the EA
    - (id, bb_i, sbb_i)

- Authorisation by the EA
  - Check voter eligibility and previous authorization requests for double voting
  - If everything is ok sign the blind ballot and return it to the voter
    - $sbb_i' = Sign(sk_{EA}, bb_i)$
  - Announce total number of eligible voters by publishing to the BB
    - $T = \{id, bb_i, sbb_i'\}$

A practical secret voting scheme for large scale elections (FOO)

- Voting **Phase 1**
  - Unblind the ballot signature
    - $sb_i = Unblind(sbb_i')$
  - Send ballot and signature to the BB through **an anonymous channel**
    - (b_i, sb_i)
  - Eligibility is publicly verifiable by verifying the EA signature
  - Everybody can create a list of eligible ballots and verify it against T

- Voting **Phase 2**
  - **After everyone has voted!**
  - Send decommitment values over an anonymous channel
    - (v_i, r_i)
- **(Public)** Counting Phase
  - Verify all commitments
  - Verify eligibility
  - Compute tally using successfully verified decommited values

A practical secret voting scheme for large scale elections (FOO)

- Voting **Phase 1**
  - Unblind the ballot signature
    - $sb_i = Unblind(sbb_i')$
  - Send ballot and signature to the BB through **an anonymous channel**
    - (b_i, sb_i)
  - Eligibility is publicly verifiable by verifying the EA signature
  - Everybody can create a list of eligible ballots and verify it against T

- Voting **Phase 2**
  - **After everyone has voted!**
  - Send decommitment values over an anonymous channel
    - (v_i, r_i)
- **(Public)** Counting Phase
  - Verify all commitments
  - Verify eligibility
  - Compute tally using successfully verified decommited values

Voting with Blind Signatures: Discussion

- **Privacy**
  - Commitment schemes
  - Blind signatures
  - Anonymous channel
- Major difference with Helios
  - **There is no need to require trust in the server for privacy!**

- **Verifiability**
  - Individual
    - Existence of the signed ballots and decommitments in the BB
  - Universal
    - Counting can be replicated
    - No secret keys involved
  - Elibigility
    - Based on the unforgeability of the blind signature scheme

But: Voting is a two-step process in different protocol phases

Voting With Ring Signatures

LSAG Voting

- Remove the authority from the FOO scheme
- All voters have cryptographic key pairs
- There is a **reliable** repository of identities and public key pairs
  - Who creates it?
- **Voting phase:**
  - Each voter picks v_i and signs it using a LSAG scheme
    - The ring is selected from the public repository of identities
  - The ballot is (v_i, σ_i)
  - The ballot is posted via an anonymous channel
- **Tallying phase:**
  - Everyone can retrieve the ballots from the BB and verify the signatures by retrieving the identities from the repository

Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. "Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract)". In: *ACISP 2004*. Vol. 3108. LNCS. 2004, pp. 325–335. doi: 10. 1007/978-3-540-27800-9_28.

LSAG Voting

- **Privacy**
  - Anonymous channel
  - Ring anonymity
- **Verifiability**
  - The counting process can be performed by everyone
  - The linkability property of the LSAG prevents double voting

Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. “Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract)”. In: *ACISP 2004*. Vol. 3108. LNCS. 2004, pp. 325–335. doi: 10. 1007/978-3-540-27800-9_28.

Open Vote Network

Decentralised Voting

A different paradigm

- Large scale election
  - Authorities involved
    - mixing,
    - tallying,
    - BB maintenance
  - Some trust required
  - Each voter is only interested in casting their ballot
    - Vote & Go

- Boardroom voting
  - No entity is special
  - Conducted by the voters themselves
    - They may send other messages except their votes
  - Private channels lead to disputes
  - Robustness is important
    - A voter should not disrupt an election

Anonymous Voting by 2-Round Public Discussion

- **Setup**
  - Select a group $\mathbb{G}$ of prime order q
- **Preparation**
  - Each of n voters V_i samples $x_i \overset{\$}{\leftarrow} \mathbb{Z}_q$
- **Commitment**
  - Each V_i broadcasts $\langle g^{x_i}, DLPRV(x_i, g, g^{x_i})\rangle$
  - When every voter is finished everyone computes
    - $Y_i = \frac{\prod_{j=1}^{i-1} g^{x_j}}{\prod_{j=i+1}^{n} g^{x_j}} = g^{y_i}$ for some $y_i \in \mathbb{Z}_q$
- **Voting**
  - Each V_i selects $v_i \in \{0,1\}$ and broadcasts
    - $Y_i^{x_i} g^{v_i}$
- **Self-Tallying**
  - Everyone computes
    - $\prod_{i=1}^{n} Y_i^{x_i} g^{v_i} = g^{\sum_i v_i}$
  - Solve simple DLOG

Hao, Feng, Peter Y. A. Ryan and Piotr Zielinski. "Anonymous voting by two-round public discussion." *IET Inf. Secur.* 4 (2010): 62-67.

Protocol Magic

- Correctness
  - $\sum_i x_i y_i = \sum_{i=1}^{n} \sum_{j=1}^{i-1} x_i x_j - \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i x_j = 0$
- Intuition

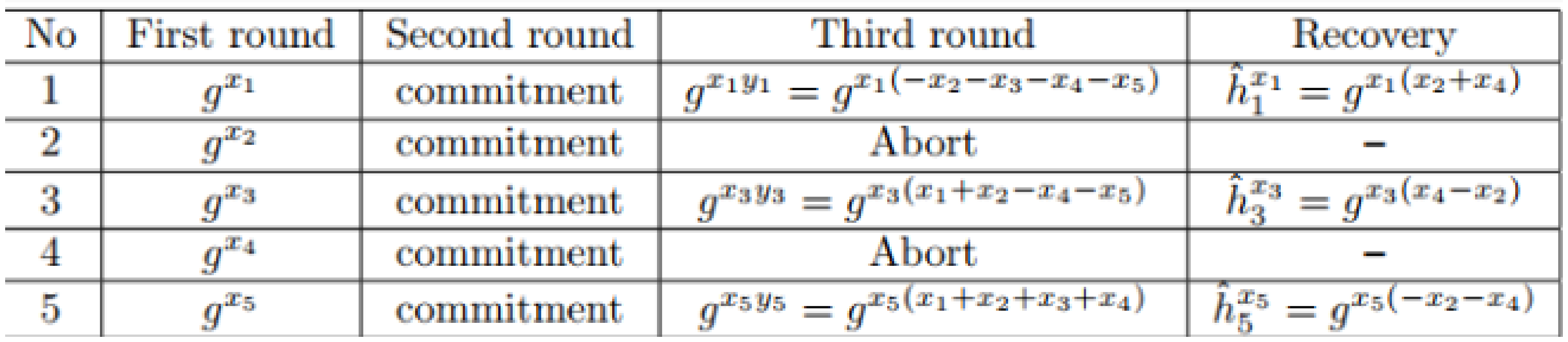
	x_1	x_2	x_3	x_4
x_1		-	-	-
x_2	+		-	-
x_3	+	+		-
x_4	+	+	+	

Robustness - Fairness

- The protocol is not robust
  - If a voter that has participated in the **commitment round** does not participate in the **voting round** the result cannot be computed
- The protocol is not fair
  - The last voter learns the result before the rest
  - They can adapt their vote for a favorable result
- Solution: A recovery round

Recovery Round

- L: The set of voters that have performed both rounds
  - They participate in one more round in order to post cancellation tokens
  - They compute $Z_i \leftarrow \frac{\prod_{j\in[i+1,n]\setminus L} g^{x_i}}{\prod_{j\in[1,i-1]\setminus L} g^{x_i}}$
  - The cancellation token is $(Z_i^{x_i}, \boldsymbol{DLPRV}(x_i, g, Z_i))$
  - They are used to remove the commitments of the players that did not vote
- Tallying becomes
  - $\prod_{i=1}^{n} Y_i^{x_i} g^{v_i} \cdot \prod_{i\in L} Z_i^{x_i} = g^{\sum_{i\in L} v_i}$



No	First round	Second round	Third round	Recovery
1	g^{x_1}	commitment	$g^{x_1 y_1} = g^{x_1(-x_2-x_3-x_4-x_5)}$	$\hat{h}_1^{x_1} = g^{x_1(x_2+x_4)}$
2	g^{x_2}	commitment	Abort	–
3	g^{x_3}	commitment	$g^{x_3 y_3} = g^{x_3(x_1+x_2-x_4-x_5)}$	$\hat{h}_3^{x_3} = g^{x_3(x_4-x_2)}$
4	g^{x_4}	commitment	Abort	–
5	g^{x_5}	commitment	$g^{x_5 y_5} = g^{x_5(x_1+x_2+x_3+x_4)}$	$\hat{h}_5^{x_5} = g^{x_5(-x_2-x_4)}$

Implementation on the Blockchain

- Ethereum
- Smart Contracts for
  - Registration (using the accounts of the voter)
  - Voting
  - Tallying
- Restrictions
  - integers of 256 bits
  - expensive cryptographic computations
  - one vote or six registrations per block
  - small number of allowed local variables
  - order of transactions in a block and timers
- Linear Complexity for Tally And Vote
- Maximum number of voters: 50
- Cost/voter: 0.73$ (2017)

Patrick McCorry, Siamak Shahandashti, and Feng Hao, A smart contract for boardroom voting with maximum voter privacy, pp. 357–375, 01 2017.

Improvements

- Organize voters in Merkle Tree
  - only the root is stored (256 bits)
- Instead of voter list a voter provides a proof of membership
- Tally off-chain by an untrusted tallier
- Publish computation trace in Merkle Tree
- Subject to verification

Mohamed Seifelnasr and Hisham Galal, Scalable open-vote network on ethereum, pp. 436–450, 08 2020

Voting on the blockchain

- Conceptual similarity between blockchain and the BB
    - Append-only
    - Broadcast channel
    - No central authority - anyone can be a miner (given enough computing power)
    - Pseudonymity
- Good for universal/individual verifiability (recorded as cast)
- But...
- Registration/authentication/eligibility verifiability are inherently centralized
- Does not help with verifying voter intent
- Does not help with coercion-resistance / receipt-freeness
- Intensifies threats associated with everlasting privacy
- Is it actually decentralized? (concentration of mining power)

Voting on the blockchain

- To sum up... ‘*using Blockchain for voting solves a small part of the problem with an unnecessarily big hammer*’ (Ben Adida, 2017)
- However...
- …it might be useful for different types of elections
  - new election paradigms on a smaller scale
  - blockchain governance

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

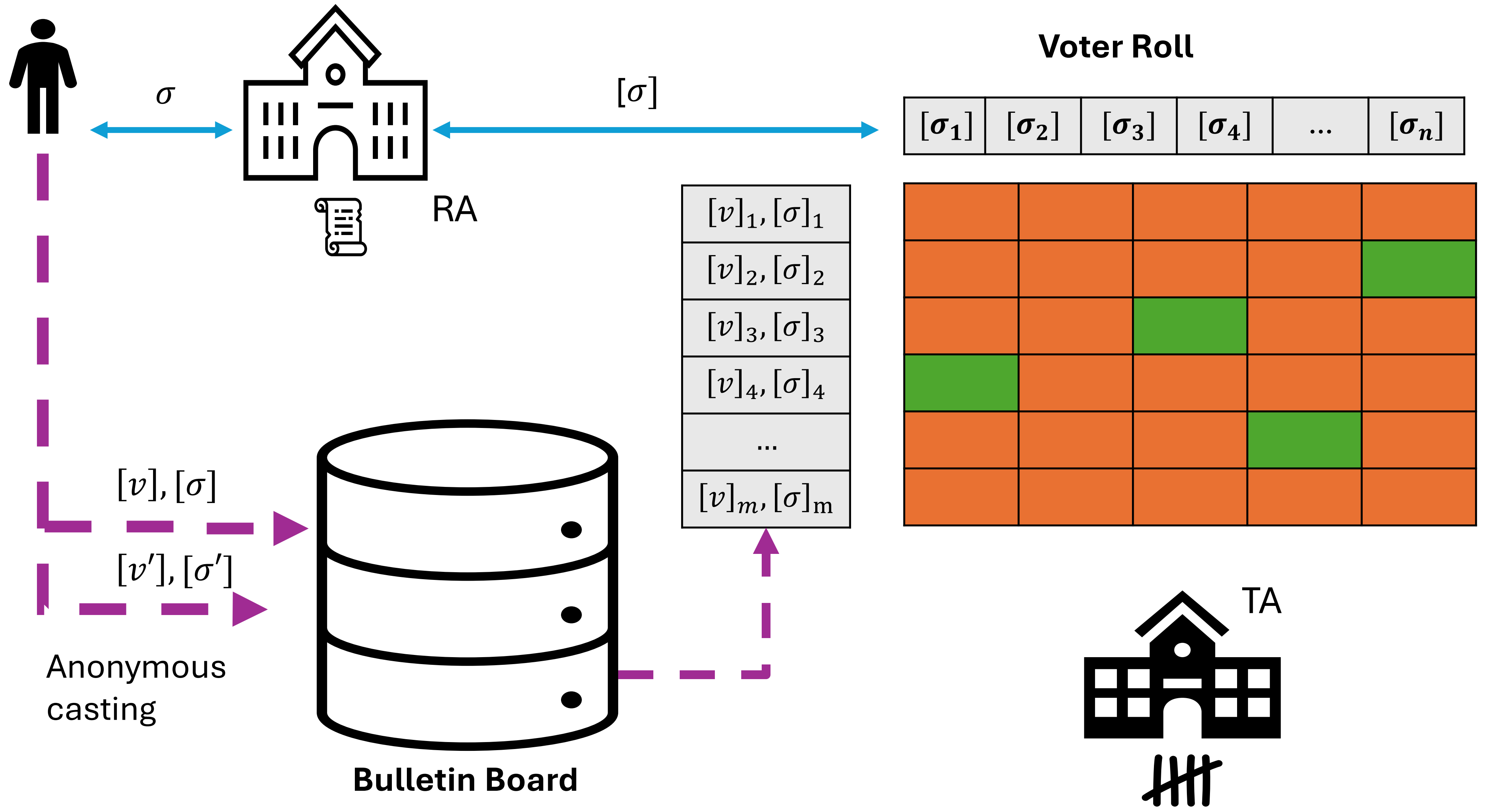
- The voter registers their real credential
    - Untappable registration – occurs once but may be reused
    - The voter may create the credential either alone or together with an authority
- The authorities publishes all real credentials in encrypted form
    - Voter roll
- Coercion Attack
    - The 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 σ^z (credential fingerprint)
- The pair (σ^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 σ^z and σ^{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 σ 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 β 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

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

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

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

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**

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

$\boldsymbol{OCorrupt(i)}$

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

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

```
$BB \Leftarrow (i, b)$
```

Weak verifiability

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

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

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

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

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

```
$pp \leftarrow VS.Setup(1^\lambda)$
$pk_{EA}, sk_{EA}, BB \leftarrow VS.SetupElection(pp)$
$(CS, BB, T_A, \pi_{T_A}) \leftarrow A^{ORegister,OCorrupt,OVote}(pp, pk_{EA})$
$if\ T_A = \perp\ \mathbf{or}\ VS.Verify(CS, BB, T_A, \pi_{T_A}) = 0\ then$
    $return\ 0$
$V_{CH} \leftarrow \{(i_{CH}, v_{CH}, b_{CH})\}_{i=1}^{n_{CH}}$ //voters who performed verification
$if\ \exists \{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}\ where\ 0 \le n_{V_{CR}} \le |V_{CR}|$
    $\mathbf{and}\ T_A = result(\{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}) \oplus result(\{v_i \mid i \in V_H - V_{CH}\}_{i=1}^{n_{V_H} - n_{CH}})$
        $\oplus\ result(\{v_i \mid i \in V_{CH}\}_{i=1}^{n_{CH}})$
    $return\ 0$
$return\ 1$
```

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

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

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

```
$pp \leftarrow VS.Setup(1^\lambda)$
$pk_{EA}, sk_{EA}, BB \leftarrow VS.SetupElection(pp)$
$(\boldsymbol{CS}, \boldsymbol{T_A}, \boldsymbol{\pi_{T_A}}) \leftarrow \boldsymbol{A^{OCorrupt,OVote,OCast}(pp, pk_{EA})}$
$if\ T_A = \perp\ \mathbf{or}\ VS.Verify(CS, BB, T_A, \pi_{T_A}) = 0\ then$
    $return\ 0$
$V_{CH} \leftarrow \{(i_{CH}, v_{CH}, b_{CH})\}_{i=1}^{n_{CH}}$ //voters who performed verification
$if\ \exists\{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}\ where\ 0 \le n_{V_{CR}} \le |V_{CR}|$
    $\mathbf{and}\ T_A = result(\{v_j \mid j \in V_{CR}\}_{j=1}^{n_{V_{CR}}}) \oplus result(\{v_i \mid i \in V_H - V_{CH}\}_{i=1}^{n_{V_H}-n_{CH}})$
                $\oplus result(\{v_i \mid i \in V_{CH}\}_{i=1}^{n_{CH}})$
    $return\ 0$
$return\ 1$
```

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

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**
  - $$Pr\left[\begin{array}{c}(T,\pi) \leftarrow VS.Tally(\{b_1, \dots, b_n\}); \\ b_i = VS.Vote(i, v_i, sk_i), v_i \in CS \text{ and } VS.IsValid(b_i) = 1\ \forall i \\ \text{and } VS.Verify(T, \pi, \{b_1, \dots, b_n\}) = 1 \\ \text{and } T = result(\{v_1, \dots v_n\})\end{array}\right] = 1$$
- **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**

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

- **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, π_T is the output of A in $\mathbf{Weak-Ver_{VS,A}}$
- $Verify(BB, T, \pi_T) = 1$ and $T \neq \bot$
- Since the BB is honest:
  - $\forall b \text{ in } BB: IsValid(b) = 1$ (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 $Tally(\mathrm{BB}) = Tally(BB_1) \oplus Tally(BB_2)$

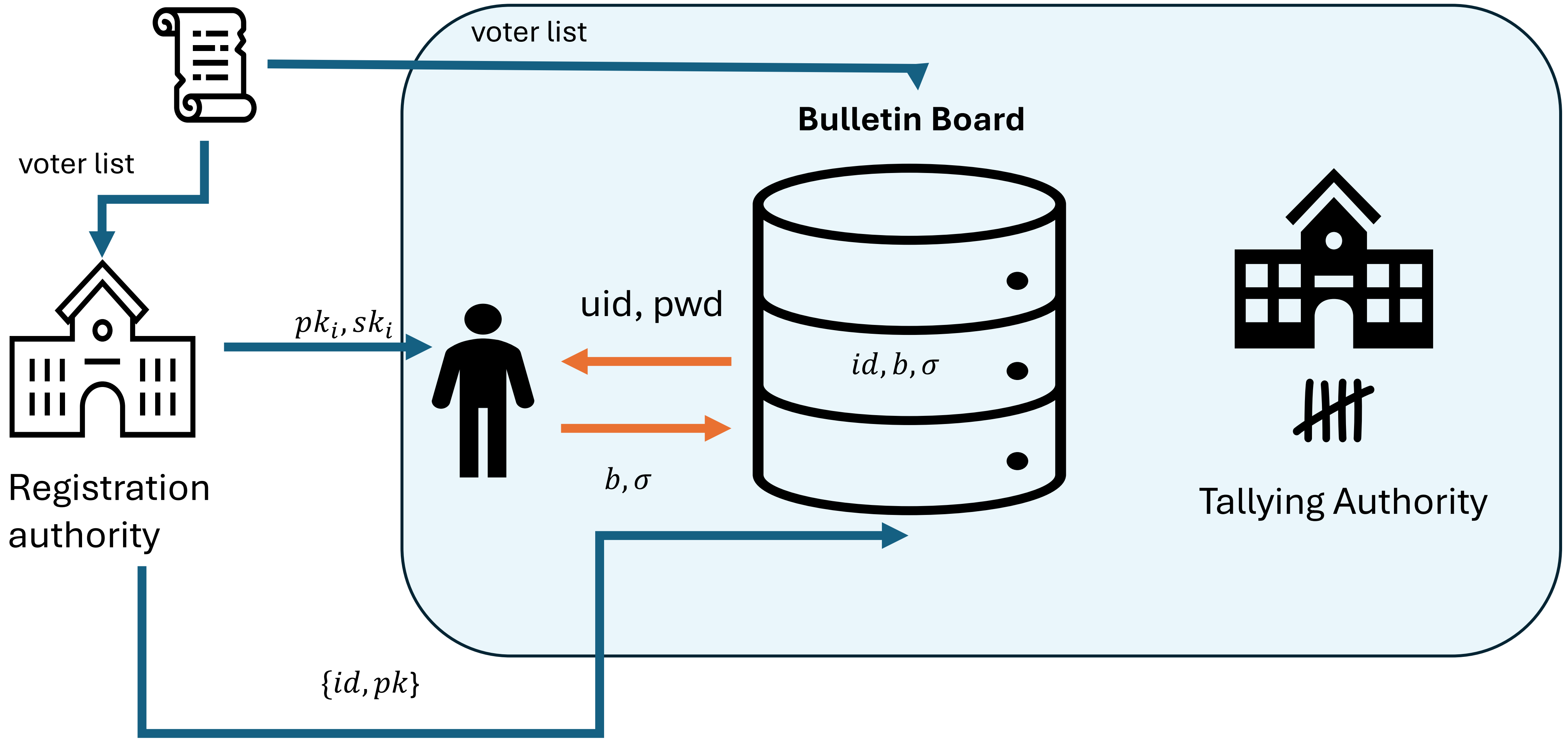
Weak verifiability for Helios

- 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^{σ} 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^{σ} against VS^{σ} **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^{σ} is well-formed, since it passes $Verify^{\sigma}$.
- This means that all ballots in BB^{σ} are valid. Thus:
- For every honest vote that has a corresponding ballot $b_H = (pk_u, a_H, \sigma)$ in BB_{σ} 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^σ corresponds to an honest vote (output of Vote)

If not: since it is placed in BB^σ it must have a valid signature.

Since σ 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^σ with the same credential. But BB^σ is well-formed.

Or: A^σ 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 BBs
  - 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

$\boldsymbol{OBoard(b)}$

```
$return\ Publish(BB_b)$
```

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

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

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

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

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

$\boldsymbol{OTally(b)}$

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

The BPRIV definition

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

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

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

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

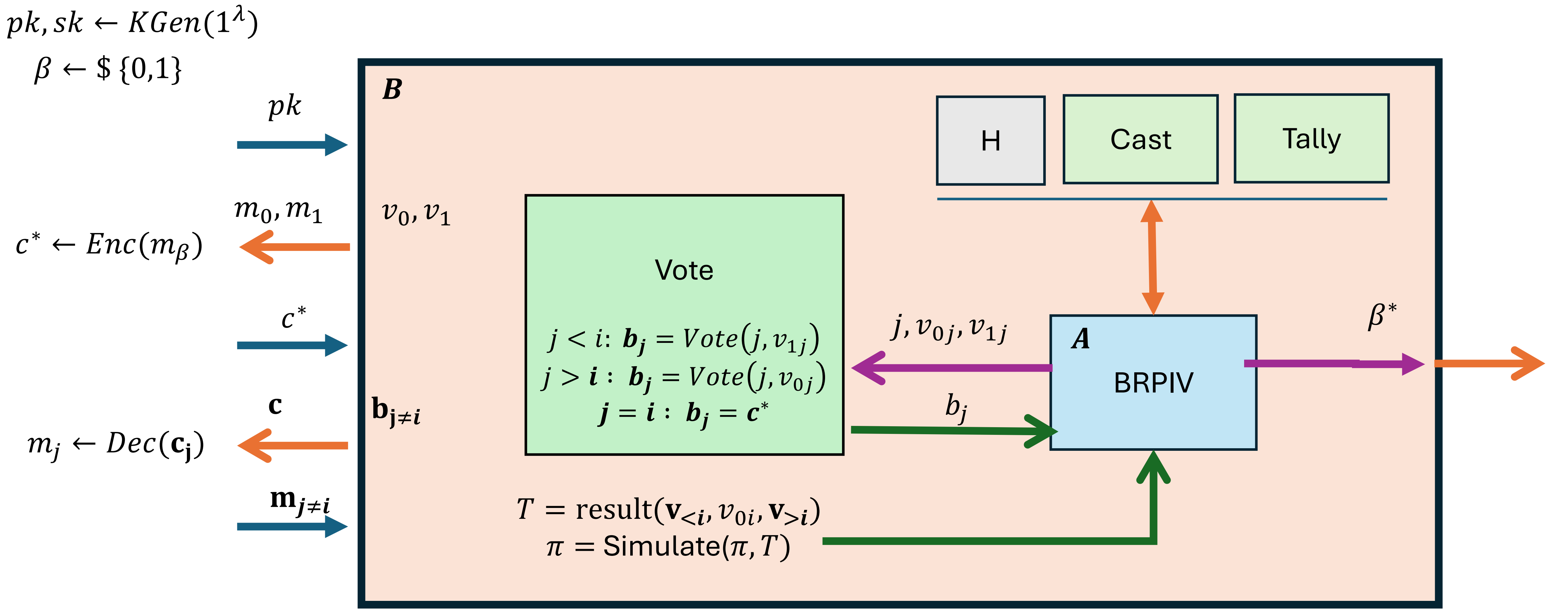
Helios is BPRIV

- 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 $\Rightarrow Game_{2,i-1} \approx Game_{2,i}$)


$pk, sk \leftarrow KGen(1^\lambda)$
$\beta \leftarrow \$ \{0,1\}$
pk
m_0, m_1
$c^* \leftarrow Enc(m_\beta)$
c^*
$\mathbf{c}$
$m_j \leftarrow Dec(\mathbf{c_j})$
$\mathbf{m}_{j\neq i}$
$\boldsymbol{B}$
v_0, v_1
$\mathbf{b_{j\neq i}}$
H
Cast
Tally
Vote
$j < i$: $\boldsymbol{b_j} = Vote(j, v_{1j})$
$j > \boldsymbol{i}$: $\boldsymbol{b_j} = Vote(j, v_{0j})$
$\boldsymbol{j} = \boldsymbol{i}$: $\boldsymbol{b_j} = \boldsymbol{c}^*$
j, v_{0j}, v_{1j}
b_j
$\boldsymbol{A}$
BRPIV
β^*
$T = \text{result}(\mathbf{v}_{<i}, v_{0i}, \mathbf{v}_{>i})$
$\pi = \text{Simulate}(\pi, T)$