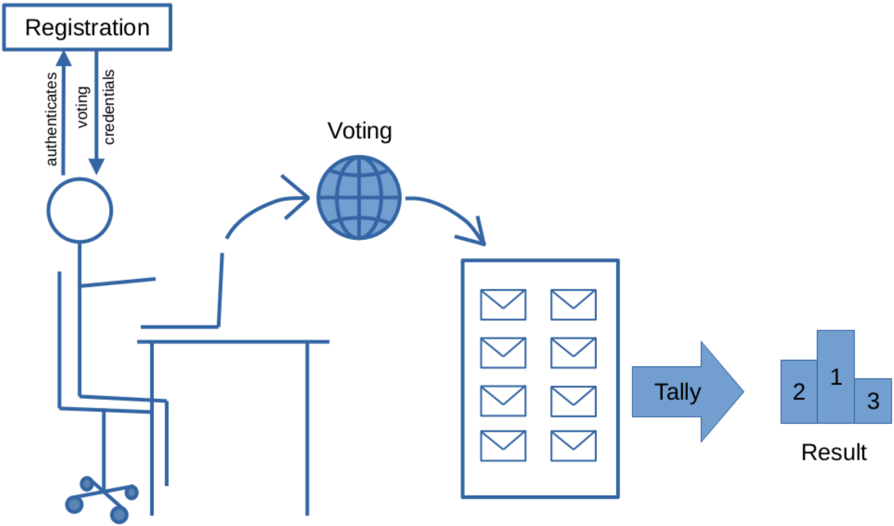
Coercion resistance in electronic voting: design and analysis

Quentin Yang
Under the supervision of Véronique Cortier and Pierrick Gaudry

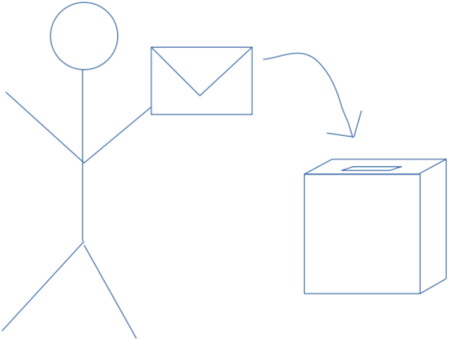
LORIA, 23 June 2023

Table of Contents

What is electronic voting?

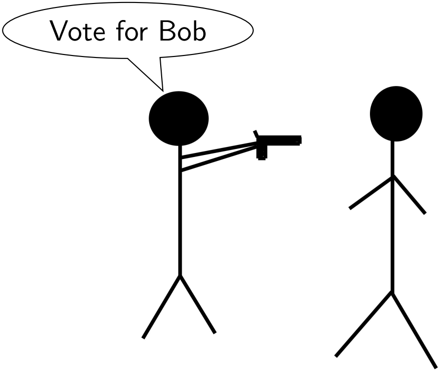


The security goals in electronic voting



Security properties
• Confidence in the result ✓Eligibility ✓Cast as intended ✓Recorded as cast ✓Tallied as recorded • Vote secrecy

A more advanced property: coercion-resistance



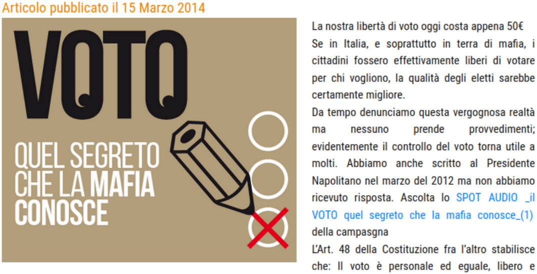
voteselling.onion

Sell your vote!
Easy money!

Fast and secure!

Offer ends in 23 : 58 : 37

Vote buying in practice


Articolo pubblicato il 15 Marzo 2014

La nostra libertà di voto oggi costa appena 50€

Se in Italia, e soprattutto in terra di mafia, i cittadini fossero effettivamente liberi di votare per chi vogliono, la qualità degli eletti sarebbe certamente migliore.

Da tempo denunciamo questa vergognosa realtà ma nessuno prende provvedimenti; evidentemente il controllo del voto torna utile a molti. Abbiamo anche scritto al Presidente Napolitano nel marzo del 2012 ma non abbiamo ricevuto risposta. Ascolta lo SPOT AUDIO _il VOTO quel segreto che la mafia conosce_(1) della campasgna

L'Art. 48 della Costituzione fra l'altro stabilisce che: Il voto è personale ed eguale, libero e

Petition to stop vote buying in Italia (Libero Futuro, 2012)


Le Monde

Consulter le journal

Se connecter S'abonner

ACTUALITÉS PRÉSIDENTIELLE 2022 ÉCONOMIE VIDÉOS DÉBATS CULTURE M LE MAG SERVICES

INTERNATIONAL · LETTRES DE

Partage

« Beaucoup d'électeurs sont prêts à vendre leur voix » : l'achat de votes, un fléau bulgare

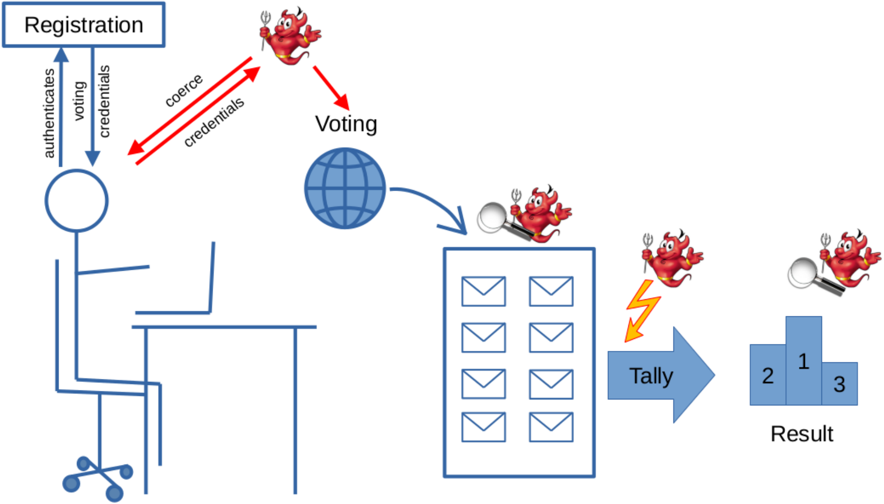
A la veille des élections législatives en Bulgarie du 4 avril, une étude confirme une pratique endémique et partagée par tous les partis du pays : payer les électeurs pour s'assurer leur voix.

Par Jean-Baptiste Chastand (Vienne, correspondant régional)

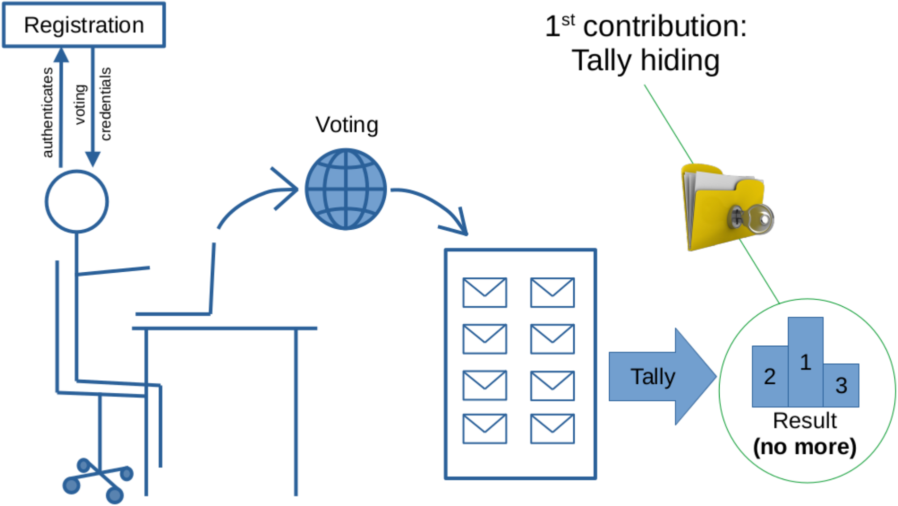
Publié le 02 avril 2021 à 00h14 - Mis à jour le 02 avril 2021 à 12h18 · Lecture 4 min.

Article about vote buying in Bulgaria (Le Monde, 2021)

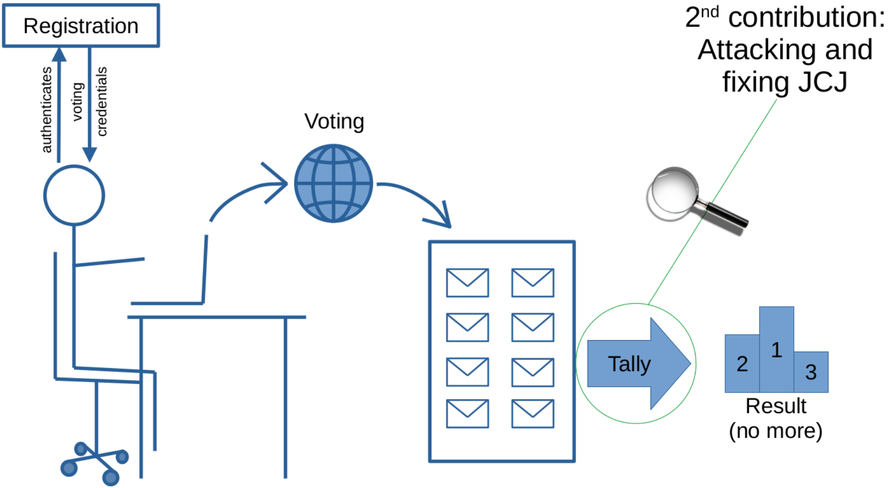
The adversary in coercion-resistance



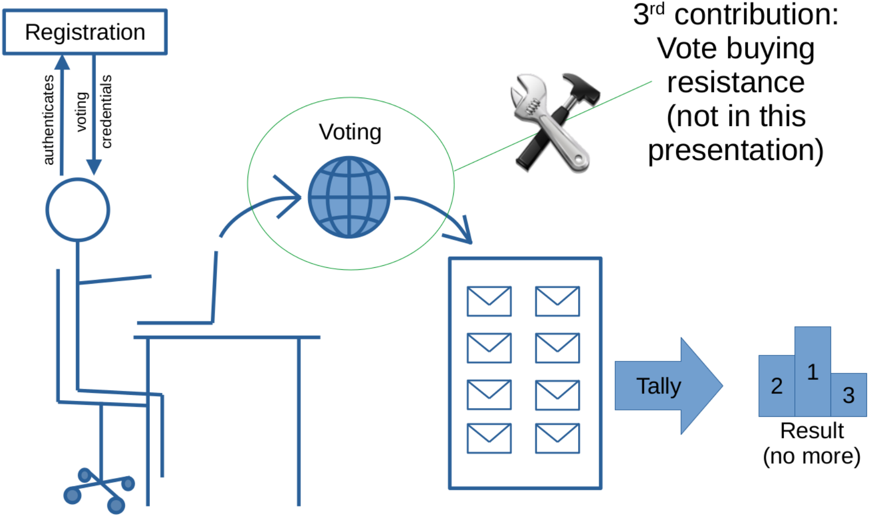
Our contributions



Our contributions

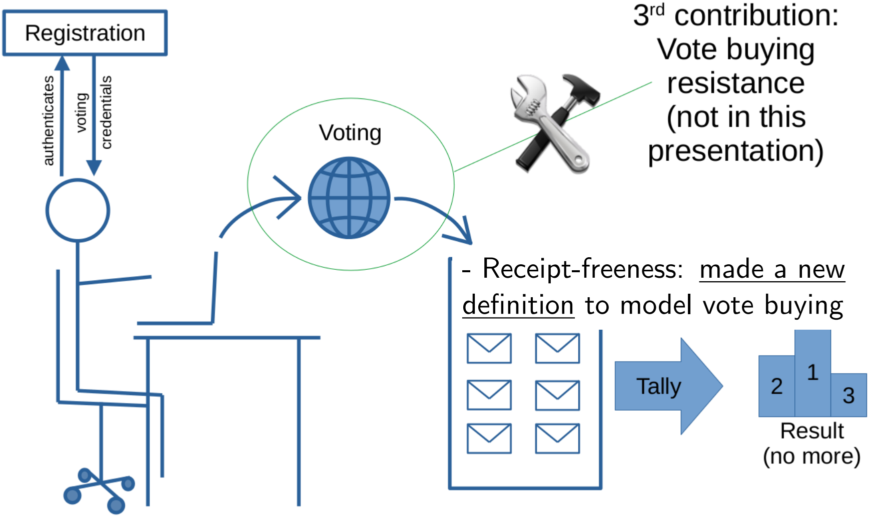


Our contributions



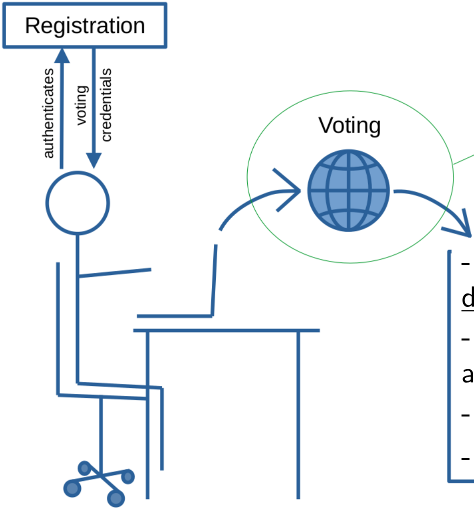
3rd contribution:
Vote buying
resistance
(not in this
presentation)

Our contributions



- Receipt-freeness: made a new definition to model vote buying

Our contributions



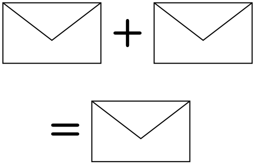
3rd contribution: Vote buying resistance (not in this presentation)

- Receipt-freeness: made a new definition to model vote buying
- Designed a voting protocol to achieve receipt-freeness
- Provided security proofs
- And more...

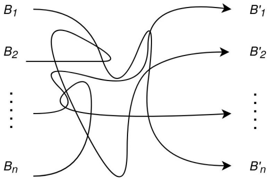
Outline

Two main strategies to compute the tally

Homomorphic tally



Mixnet



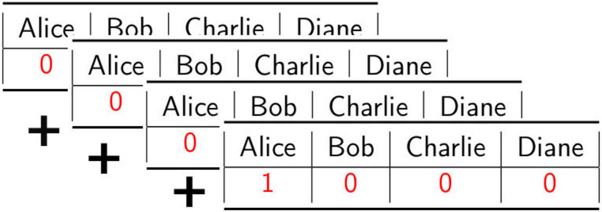
The homomorphic tally

Candidates : Alice, Bob, Charlie, Diane

Ballots :

Alice	Bob	Charlie	Diane
0	0	1	0

Tally :



=

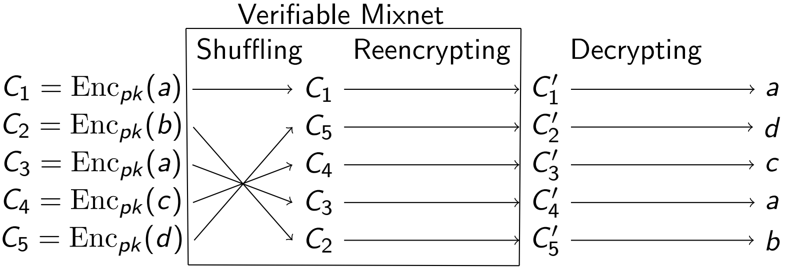
Alice	Bob	Charlie	Diane
1	2	1	0

decrypt

Result :

Alice	Bob	Charlie	Diane
1	2	1	0

The mixnet



A mixnet reveals the multiset of the voting options chosen.

Why tally hiding?

Some voting systems (Condorcet, STV) let you choose any permutation of the candidates.

1. Alice
2. Bob
3. Charlie
4. Diane
5. Eve

⋮

1. Alice
2. Eve
3. Diane
4. Greg
5. Bob

⋮

1. Alice
2. Kolmogorov
3. Zarathustra
4. Leto
5. Puccini

⋮

1. Eve
2. Bob
3. Freya
4. Diane
5. Alice

⋮

1. Eve
2. Charlie
3. Alice
4. Bob
5. Diane

⋮

Why tally hiding?

Some voting systems (Condorcet, STV) let you choose any permutation of the candidates.

1. Alice
2. Bob
3. Charlie
4. Diane
5. Eve

⋮

1. Alice
2. Eve
3. Diane
4. Greg
5. Bob

⋮

1. Alice
2. Kolmogorov
3. Zarathustra
4. Leto
5. Puccini

⋮

unlikely permutation

1. Eve
2. Bob
3. Freya
4. Diane
5. Alice

⋮

1. Eve
2. Charlie
3. Alice
4. Bob
5. Diane

⋮

Why tally hiding?

Some voting systems (Condorcet, STV) let you choose any permutation of the candidates.

1. Alice
2. Bob
3. Charlie
4. Diane
5. Eve

⋮

1. Eve
2. Bob
3. Freya
4. Diane
5. Alice

⋮

1. Alice
2. Eve
3. Diane
4. Greg
5. Bob

⋮

1. Eve
2. Charlie
3. Alice
4. Bob
5. Diane

⋮

1. Alice
2. Kolmogorov
3. Zarathustra
4. Leto
5. Puccini

⋮

unlikely permutation

Italian Attack!
(vote buying)

Why tally hiding?

Some voting systems (Condorcet, STV) let you choose any permutation of the candidates.

1. Alice
2. Bob
3. Charlie
4. Diane
5. Eve

⋮

1. Eve
2. Bob
3. Freya
4. Diane
5. Alice

⋮

1. Alice
2. Eve
3. Diane
4. Greg
5. Bob

⋮

1. Eve
2. Charlie
3. Alice
4. Bob
5. Diane

⋮

1. Alice
2. Kolmogorov
3. Zarathustra
4. Leto
5. Puccini

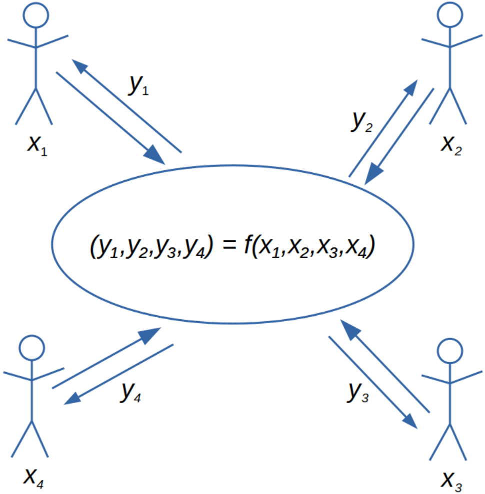
⋮

unlikely permutation

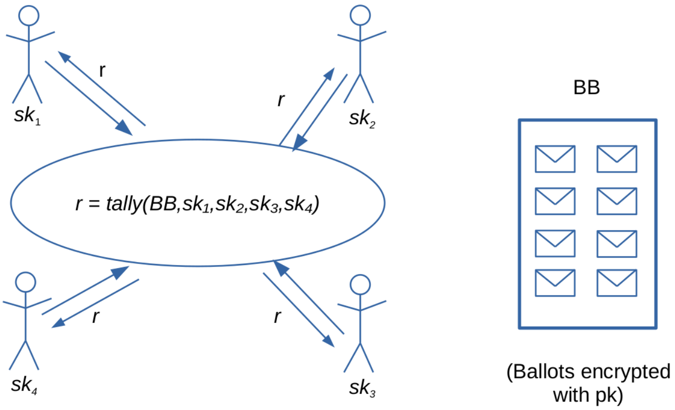
Italian Attack!
(vote buying)

Tally hiding: Alice wins.

Multi-party computation



Multi-party computation



Tally-hiding in the literature

Single choice voting: Voters select one candidate or list.

☐	Ape
☐	Beaver
☑	Capybara
☐	Dolphin
☐	Elephant

- $s-$best. Küsters *et al.*, EuroS&P'20 (Ordinos).
- Hare-Niemeyer. Hertel *et al.*, E-VoteID'20 (Ordinos).

Tally-hiding in the literature

Single choice voting: Voters select one candidate or list.

☐	Ape
☐	Beaver
☑	Capybara
☐	Dolphin
☐	Elephant

- $s-$best. Küsters *et al.*, EuroS&P'20 (Ordinos).
- Hare-Niemeyer. Hertel *et al.*, E-VoteID'20 (Ordinos).

Shortcomings in case of a tie! (Our finding)

Tally-hiding in the literature

Majority Judgment: Voters grade each candidate.
(Used in a primary election in France.)

Ape	Reject
Beaver	Good
Capybara	Excellent
Dolphin	Good
Elephant	Bad

- Canard *et al.*, ESORICS'18.

Tally-hiding in the literature

Majority Judgment: Voters grade each candidate.
(Used in a primary election in France.)

Ape	Reject
Beaver	Good
Capybara	Excellent
Dolphin	Good
Elephant	Bad

- ~~Canard *et al.*, ESORICS'18.~~ (Majority gauge)

  Fails to tally in some not-so-rare cases.
  Not fully tally-hiding!

  Our finding

Tally-hiding in the literature

Ranked Voting: Voters rank candidates (several methods).

1	Capybara
2	Beaver
3	Dolphin
4	Elephant
5	Ape

Tally-hiding in the literature

Ranked Voting: Voters rank candidates (several methods).

The Condorcet methods are very popular (used in Debian).

1	Capybara
2	Beaver
3	Dolphin
4	Elephant
5	Ape

- Condorcet. Haines, Pattinson and Tiwari, VSTTE'19.

Tally-hiding in the literature

Ranked Voting: Voters rank candidates (several methods).

The Condorcet methods are very popular (used in Debian).

1	Capybara
2	Beaver
3	Dolphin
4	Elephant
5	Ape

- Condorcet. ~~Haines, Pattinson and Tiwari, VSTTE'19.~~
  Privacy breach when candidates are ranked equal. (Our finding)

Tally-hiding in the literature

Ranked Voting: Voters rank candidates (several methods).

The Condorcet methods are very popular (used in Debian).

1	Capybara
2	Beaver
3	Dolphin
4	Elephant
5	Ape

- Condorcet. ~~Haines, Pattinson and Tiwari, VSTTE'19.~~

  Privacy breach when candidates are ranked equal. (Our finding)
- Condorcet. Hertel *et al.*, E-VoteID'20 (Ordinos).

  Does **not** allow equal ranking.

Tally-hiding in the literature

Ranked Voting: Voters rank candidates (several methods).

STV and IRV are used for high-stake elections in several countries.

1	Capybara
2	Beaver
3	Dolphin
4	Elephant
5	Ape

- Condorcet. ~~Haines, Pattinson and Tiwari, VSTTE'19.~~

  Privacy breach when candidates are ranked equal. (Our finding)
- Condorcet. Hertel *et al.*, E-VoteID'20 (Ordinos).

  Does **not** allow equal ranking.
- Borda. Hertel *et al.*, E-VoteID'20 (Ordinos).
- Single Transferable Vote. Benaloh *et al.*, IEEE TIFS'10.
- Instant Run-off Voting. Culnane *et al.*, FC'19.
- Instant Run-off Voting. Hertel *et al.*, E-VoteID'20 (Ordinos).

Tally-hiding in the literature

Ranked Voting: Voters rank candidates (several methods).

STV and IRV are used for high-stake elections in several countries.

1	Capybara
2	Beaver
3	Dolphin
4	Elephant
5	Ape

- Condorcet. ~~Haines, Pattinson and Tiwari, VSTTE'19.~~

  Privacy breach when candidates are ranked equal. (Our finding)
- Condorcet. Hertel *et al.*, E-VoteID'20 (Ordinos).

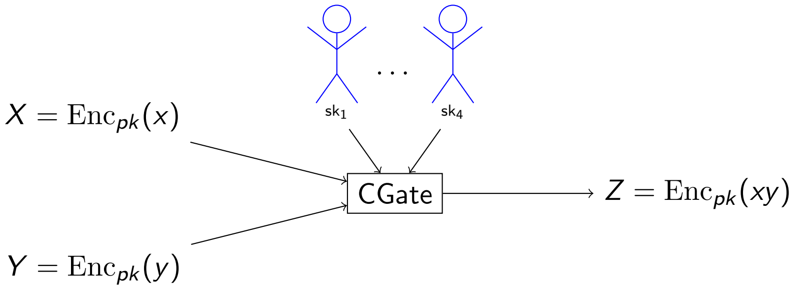
  Does **not** allow equal ranking.
- Borda. Hertel *et al.*, E-VoteID'20 (Ordinos).
- Single Transferable Vote. Benaloh *et al.*, IEEE TIFS'10.*
- Instant Run-off Voting. Culnane *et al.*, FC'19.*
- Instant Run-off Voting. ~~Hertel *et al.*, E-VoteID'20 (Ordinos).~~

  Super-exponential complexity.

* Not fully tally-hiding

Our approach - Logical operations on encrypted bits

We use the CGate primitive to build our MPC protocols:



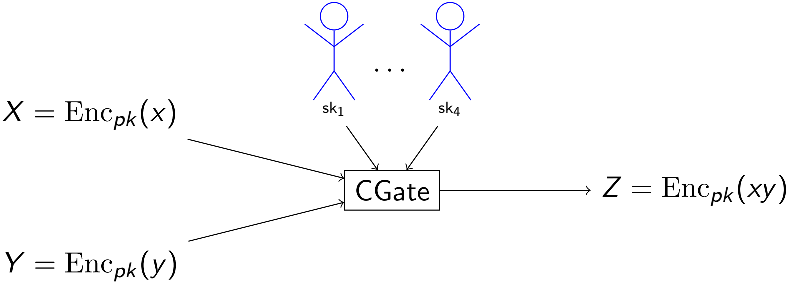
It allows logical operations on encrypted bits

Not gate: $\mathrm{Not}(B) = \mathrm{Enc}(1)/B \equiv \mathrm{Enc}(1-b)$

(Primitive adapted from Shoenmakers and Tuyls, Asiacrypt'04.)

Our approach - Logical operations on encrypted bits

We use the CGate primitive to build our MPC protocols:



It allows logical operations on encrypted bits

Not gate: $\mathrm{Not}(B) = \mathrm{Enc}(1)/B \equiv \mathrm{Enc}(1-b)$

(Primitive adapted from Shoenmakers and Tuyls, Asiacrypt'04.)

We can use the ElGamal encryption scheme.

Paillier vs ElGamal

Encryption	Paillier	ElGamal
Property	additively homomorphic $\mathrm{Enc}_{\mathsf{pk}}(m_1)\mathrm{Enc}_{\mathsf{pk}}(m_2) = \mathrm{Enc}_{\mathsf{pk}}(m_1 + m_2)$	homomorphic $\mathrm{Enc}_{\mathsf{pk}}(m_1)\mathrm{Enc}_{\mathsf{pk}}(m_2) = \mathrm{Enc}_{\mathsf{pk}}(m_1 m_2)$
Based on	Decisional Composite Residuosity Assumption	Decisional Diffie-Hellman
Key size	3072 bits	256 bits (elliptic curve)
Operation	3072−bits exponentiation modulo a 6144−bits integer	256−bits exponentiation
Libraries	??	Libsodium, OpenSSL, Crypto++, $\cdots$

A toolbox for generic MPC in ElGamal

Operation	Comment
Basic	$+, -, x$
Fixed-point division	$/$
Comparisons	$\leq, <, =$

A toolbox for generic MPC in ElGamal

Operation	Comment
Basic	$+, -, x$
Fixed-point division	$/$
Comparisons	$\leq, <, =$
Conditionals	(for branch-freeness)
$\vdots$	$\vdots$

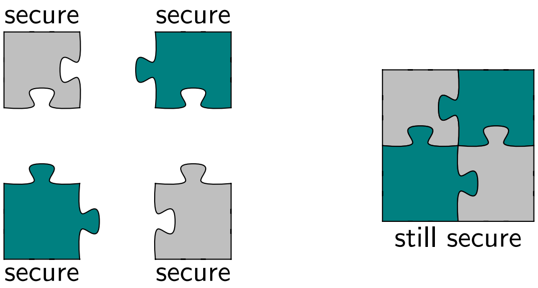
Support complex operations: Sorting, Finding the s greatest values...

One difficulty was to give several optimizations and trade-offs.

The main protocols of the toolbox

Functionality	Option	Protocol	# exp.	Synch. locks	Transcript size
Not	–	–	0	0	0
CSZ	original	[ST04]	$19n_T$	n_T	$18n_T$
	SUC-secure	CSZ	$33n_T$	n_T	$34n_T$
And, Or, Xor	–	And, Or, Xor	CSZ	CSZ	CSZ
If, Cond. swap	–	If, Swap	CSZ	CSZ	CSZ
Select, Cond. shift	–	Select, CLS, CRS	NCSZ	CSZ	NCSZ
Addition,	linear	Add, Sub	2ℓCSZ	2ℓCSZ	2ℓCSZ
subtraction	sublinear	UFCAdd	$\frac{3}{2}\ell\log\ell$CSZ	$2\log\ell$CSZ	$\frac{3}{2}\ell\log\ell$CSZ
Opposite	–	Neg	ℓCSZ	ℓCSZ	ℓCSZ
Aggregation	–	Aggreg	$3N$CSZ	$\frac{1}{2}\log(N)^2$CSZ	$3N$CSZ
	UFC	(use UFCAdd)	$5.54N$CSZ	$2N\log N\log\log N$CSZ	$5.54N$CSZ
Multiplication	–	Mult	$3\ell^2$CSZ	$2\ell^2$CSZ	$3\ell^2$CSZ
Division	–	Div	$3r\ell$CSZ	$2r\ell$CSZ	$3r\ell$CSZ
Equality	–	Eq	2ℓCSZ	$\log\ell$CSZ	2ℓCSZ
Comparison	linear	Lt	2ℓCSZ	2ℓCSZ	2ℓCSZ
	lin. + eq	LtEq	3ℓCSZ	2ℓCSZ	3ℓCSZ
	sublinear	CLt	4ℓCSZ	$2\log\ell$CSZ	4ℓCSZ
	sublin. + eq	CLt	5ℓCSZ	$2\log\ell$CSZ	5ℓCSZ
Min, max	linear	Min, Max	$(3\ell+\log N)N$CSZ	$2\ell\log N$CSZ	$(3\ell+\log N)N$
	sublinear	(CLt instead of Lt)	$(5\ell+\log N)N$CSZ	$2\log\ell\log N$CSZ	$(5\ell+\log N)N$CSZ
s largest	comp.	sInsert	$(N-\frac{s}{2})s(3\ell+\log N)$CSZ	$2\ell s(N-\frac{s}{2})$CSZ	$(3\ell+\log N)N$CSZ
	trade-off	sSelect	$Ns(3\ell+\log N)$CSZ	$2s\ell\log N$CSZ	$Ns(3\ell+\log N)$CSZ
	comm.	(use sublin. Max)	$Ns(5\ell+\log N)$CSZ	$2s\log\ell\log N$CSZ	$Ns(5\ell+\log N)$CSZ
Sorting	oblivious	OddEvenMergeSort	$\frac{3}{4}N\log(N)^2\ell$CSZ	$\ell\log(N)^2$CSZ	$\frac{3}{4}N\log(N)^2\ell$CSZ
	with CLt		$\frac{5}{4}N\log(N)^2\ell$CSZ	$\log\ell\log(N)^2$CSZ	$\frac{5}{4}N\log(N)^2\ell$CSZ

The key to modularity: Universal composability



We use the SUC framework of Canetti, Cohen and Lindell, Crypto'15.

Application of the toolbox to tally-hiding

We applied the toolbox to various counting methods:
Single choice voting, Majority Judgment, Condorcet, STV.

- Fixed the existing shortcomings.
- Complete leakage-free solutions.
- Based on ElGamal, as efficient as in the Paillier setting.

Application of the toolbox to tally-hiding

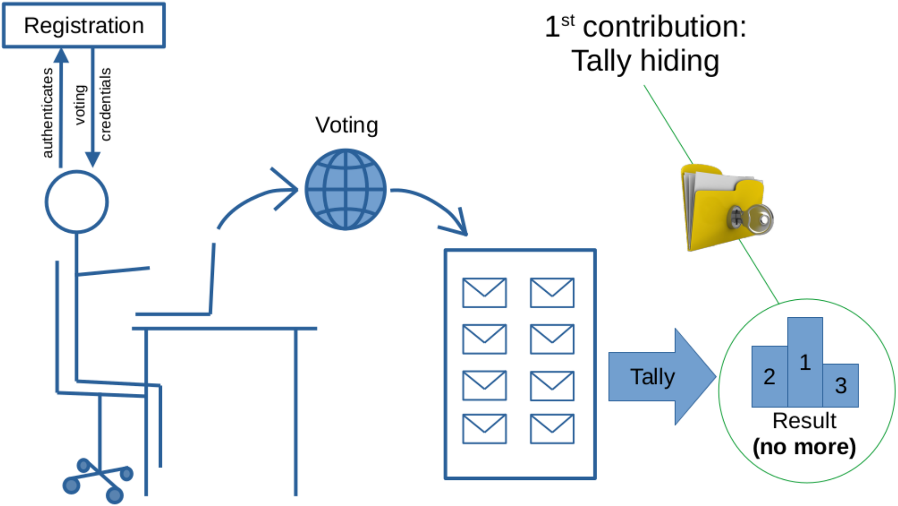
We applied the toolbox to various counting methods:
Single choice voting, Majority Judgment, Condorcet, STV.

- Fixed the existing shortcomings.
- Complete leakage-free solutions.
- Based on ElGamal, as efficient as in the Paillier setting.

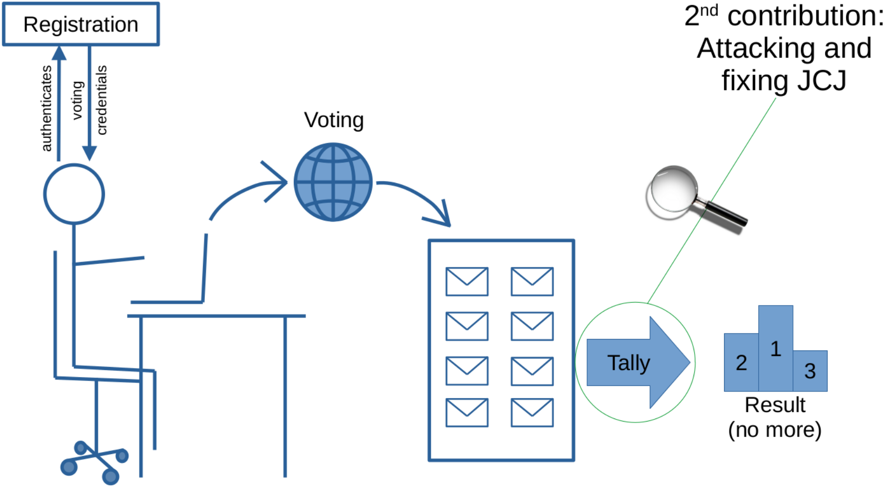
Another counting function? We can do it!

- Adaptation to the D'Hondt method, which is used in Belgium.

Our contributions



Our contributions



Outline

Registrars

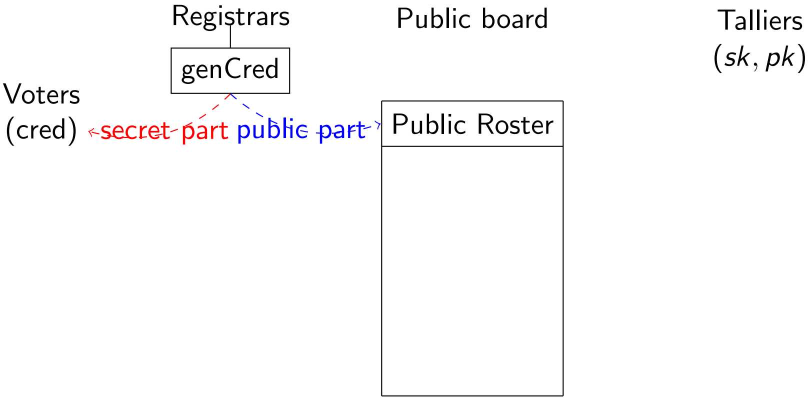
Public board

Talliers
(sk, pk)

Voters

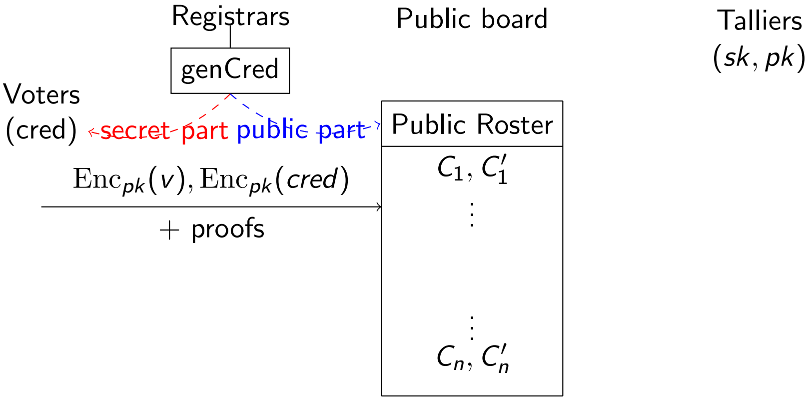
Scheme by Juels, Catalano and Jakobsson, in Coercion-Resistant Electronic Elections, WPES 2005.

Coercion-resistance in the literature: The JCJ scheme



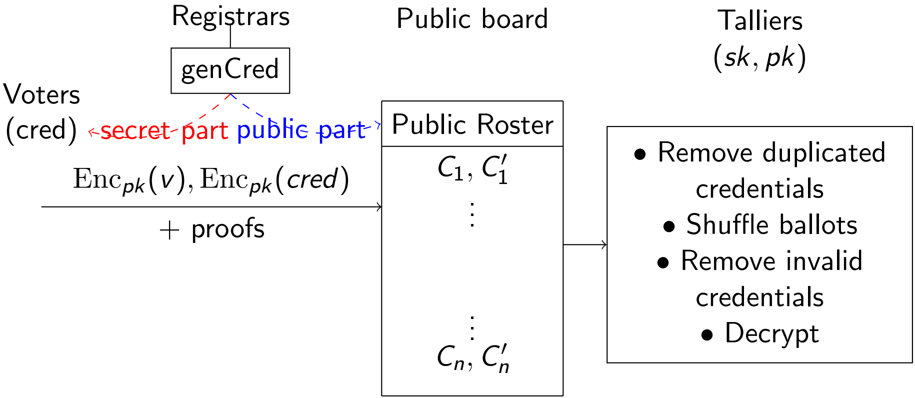
Scheme by Juels, Catalano and Jakobsson, in Coercion-Resistant Electronic Elections, WPES 2005.

Coercion-resistance in the literature: The JCJ scheme



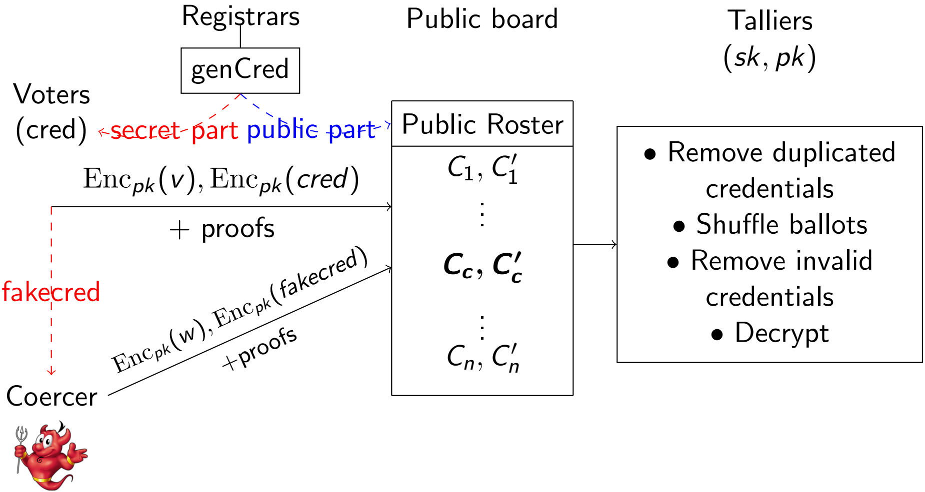
Scheme by Juels, Catalano and Jakobsson, in Coercion-Resistant Electronic Elections, WPES 2005.

Coercion-resistance in the literature: The JCJ scheme



Scheme by Juels, Catalano and Jakobsson, in Coercion-Resistant Electronic Elections, WPES 2005.

Coercion-resistance in the literature: The JCJ scheme



Scheme by Juels, Catalano and Jakobsson, in Coercion-Resistant Electronic Elections, WPES 2005.

Analysis of JCJ - Where does the security come from?

Intuitively:

- Indistinguishability of fake and real credentials.
- Untraceability of the shuffle.

Formally:

- We need a definition of coercion-resistance.

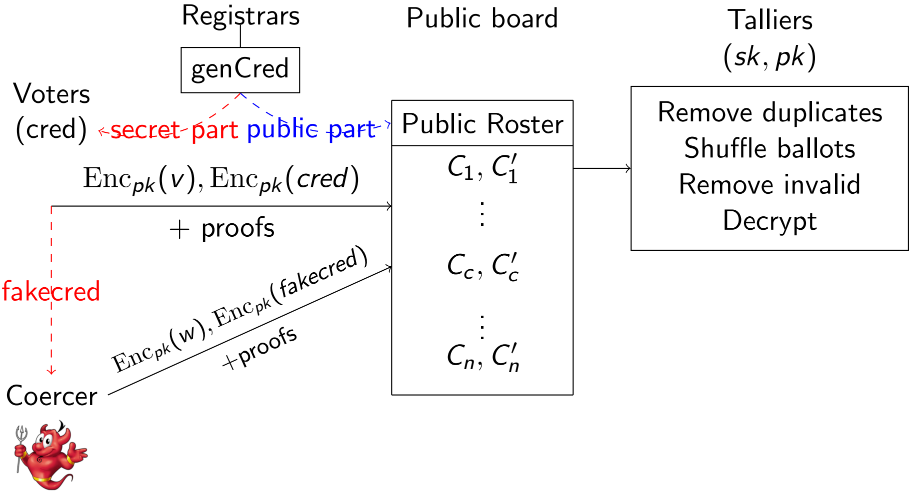
Analysis of JCJ - Defining coercion-resistance

Real Game	Ideal game
Takes part in the protocol	Learns the size of the board
Learns any public information	Learns the result

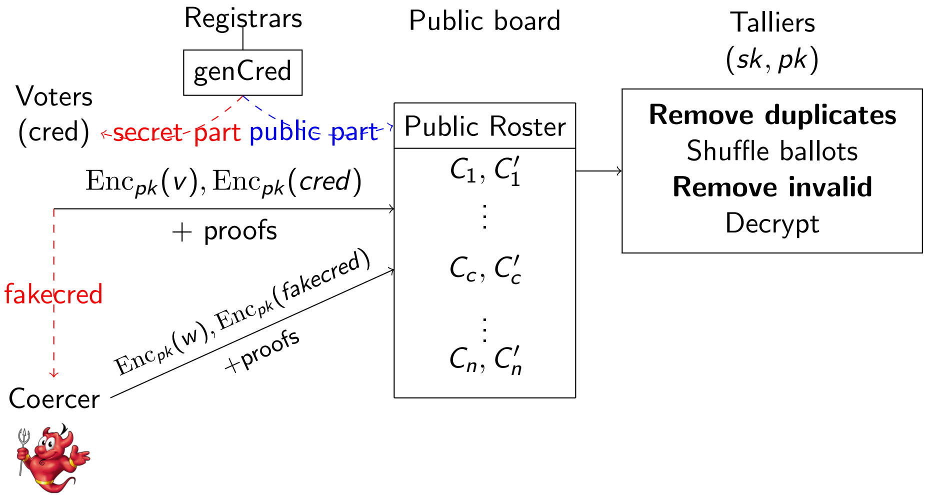
Definition (Informal)

A scheme is **coercion-resistant** if the adversary cannot guess the voter's behavior with a better probability in the real game than in the ideal game.

Coercion-resistance in the literature: The JCJ scheme



Coercion-resistance in the literature: The JCJ scheme



Analysis of JCJ - Is it really coercion-resistant?

Real Game	Ideal game
Takes part in the protocol	Learns the size of the board
Learns any public information	Learns the result
Learn the # duplicates	$\#\mathit{duplicates} + \#\mathit{invalid}$
Learn the # invalid ballots	$= \mathit{size}(\mathit{board}) - \mathit{size}(\mathit{result})$

Analysis of JCJ - Is it really coercion-resistant?

Real Game	Ideal game
Takes part in the protocol	Learns the size of the board
Learns any public information	Learns the result
Learn the # duplicates	$\#duplicates + \#invalid$
Learn the # invalid ballots	$= size(board) - size(result)$

Conclusion:

- The adversary has more information in the real game.
- JCJ is **not** coercion-resistant!

The impact of the leakage on coercion-resistance

We used a framework from Küsters, Truderung and Vogt, CSF 2010.

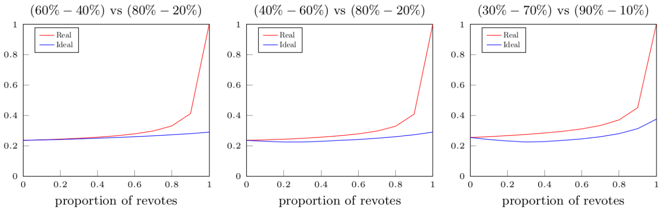


Figure 43: Coercion levels as a function of the probability of revoting; with 20 voters, 2 candidates and 30% abstention, with three different distributions between the candidates.

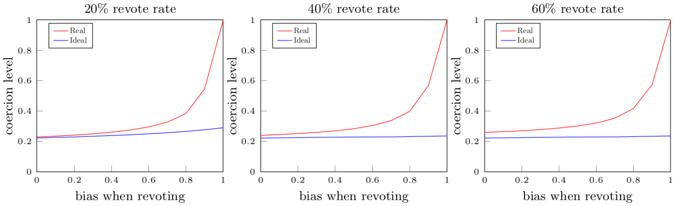
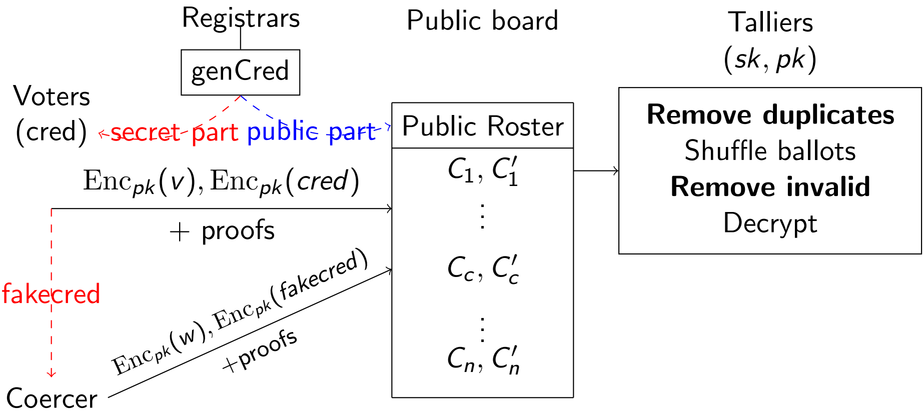
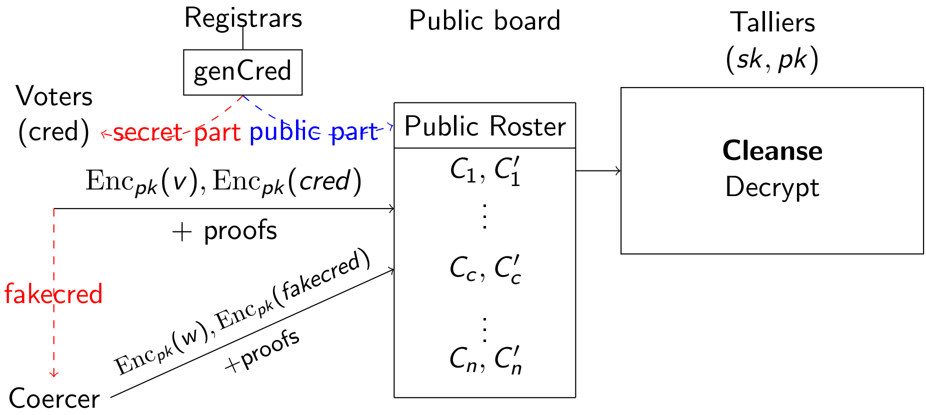


Figure 44: Coercion levels as a function of the bias when revoting; with 20 voters, 2 candidates and 30% abstention.

An application of the toolbox: cleansing-hiding

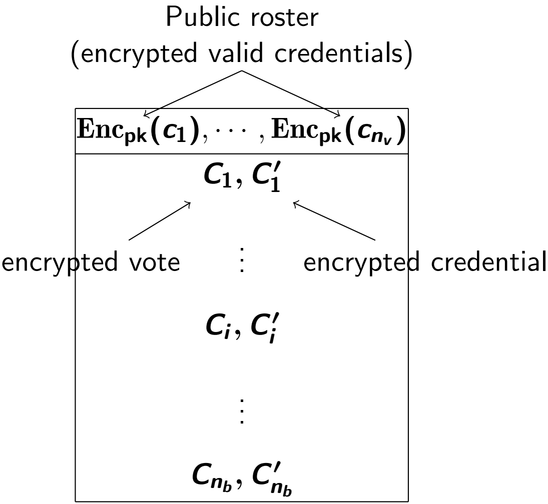


An application of the toolbox: cleansing-hiding



CHide, a cleansing-hiding fix of JCJ.

CHide: a quasi-linear solution based on sorting



CHide: a quasi-linear solution based on sorting

$\mathbf{Enc_{pk}(c_1)}, \cdots, \mathbf{Enc_{pk}(c_{n_v})}$
$\boldsymbol{C_1, C'_1}$
$\vdots$
$\boldsymbol{C_i, C'_i}$
$\vdots$
$\boldsymbol{C_{n_b}, C'_{n_b}}$

$$\mathrm{Enc}_{\mathsf{pk}}(0), \mathbf{Enc_{pk}(c_1)}, \mathrm{Enc}_{\mathsf{pk}}(n_b)$$
$$\vdots$$
$$\mathrm{Enc}_{\mathsf{pk}}(0), \mathbf{Enc_{pk}(c_{n_v})}, \mathrm{Enc}_{\mathsf{pk}}(n_b)$$

$$\boldsymbol{C_1, C'_1}, \mathrm{Enc}_{\mathsf{pk}}(0)$$
$$\vdots$$
$$\boldsymbol{C_{n_b}, C'_{n_b}}, \mathrm{Enc}_{\mathsf{pk}}(n_b - 1)$$

CHide: a quasi-linear solution based on sorting

$\mathrm{Enc}_{\mathsf{pk}}(c_1), \cdots, \mathrm{Enc}_{\mathsf{pk}}(c_{n_v})$
$C_1, \mathrm{Enc}_{\mathsf{pk}}(c_1)$ $C_2, \mathrm{Enc}_{\mathsf{pk}}(c_2)$ $C_3, \mathrm{Enc}_{\mathsf{pk}}(c_1)$ $C_4, \mathrm{Enc}_{\mathsf{pk}}(\tilde{c})$ $\vdots$ C_{n_b}, C'_{n_b}

$$\longrightarrow \begin{array}{c} \mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b) \\ \vdots \\ \mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b) \end{array}$$

$$\longrightarrow \begin{array}{c} C_1, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0) \\ C_2, \mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1) \\ C_3, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2) \\ C_4, \mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3) \\ \vdots \end{array}$$

CHide: a quasi-linear solution based on sorting

$\mathrm{Enc}_{\mathsf{pk}}(c_1), \cdots, \mathrm{Enc}_{\mathsf{pk}}(c_{n_v})$
$C_1, \mathrm{Enc}_{\mathsf{pk}}(c_1)$ $C_2, \mathrm{Enc}_{\mathsf{pk}}(c_2)$ $C_3, \mathrm{Enc}_{\mathsf{pk}}(c_1)$ $C_4, \mathrm{Enc}_{\mathsf{pk}}(\tilde{c})$ $\vdots$ C_{n_b}, C'_{n_b}

$$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)$$
$$\vdots$$
$$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b)$$

$$C_1, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)$$
$$C_2, \mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)$$
$$C_3, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)$$
$$C_4, \mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)$$
$$\vdots$$

value key

CHide: a quasi-linear solution based on sorting

$$\begin{array}{c} \underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)} \\ \vdots \\ \underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b)} \\ \\ \underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)} \\ \underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)} \\ \underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)} \\ \underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)} \\ \vdots \end{array} \left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\\1\\1\\1\\1\end{array}}\right\} \begin{array}{c} \text{sort} \longrightarrow \\ \text{by key} \\ \text{in MPC} \end{array} \begin{array}{c} \vdots \\ \underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)} \\ \underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(n_b)} \\ \vdots \\ \underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)} \\ \underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)} \\ \underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)} \\ \vdots \\ \underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)} \\ \vdots \end{array}$$

value key

CHide: a quasi-linear solution based on sorting

$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$
$$\vdots$$
$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$

$$\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)}$$
$$\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)}$$
$$\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)}$$
$$\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)}$$
$$\vdots$$

sort $\longrightarrow$
by key
in MPC

$$\vdots$$
$$\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\boldsymbol{c_2}), \mathrm{Enc}_{\mathsf{pk}}(1)}$$
$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\boldsymbol{c_2}), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$
$$\vdots$$
$$\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)}$$
$$\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)}$$
$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$
$$\vdots$$
$$\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)}$$
$$\vdots$$

value key

equal? $\rightarrow$ same credential

equal to n_b? $\rightarrow$ valid credential

CHide: a quasi-linear solution based on sorting

$$\left.\begin{array}{c}
\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)} \\
\vdots \\
\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b)} \\
\\
\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)} \\
\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)} \\
\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)} \\
\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)} \\
\vdots
\end{array}\right\}$$

sort ⟶
by key
in MPC

$$\begin{array}{c}
\vdots \\
\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)} \\
\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\boldsymbol{c_2}), \mathrm{Enc}_{\mathsf{pk}}(n_b)} \\
\vdots \\
\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)} \\
\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)} \\
\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)} \\
\vdots \\
\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)} \\
\vdots
\end{array}$$

value key

equal? → same credential

equal to n_b? → valid credential

CHide: a quasi-linear solution based on sorting

$$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)$$
$$\vdots$$
$$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b)$$

$$C_1, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)$$
$$C_2, \mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)$$
$$C_3, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)$$
$$C_4, \mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)$$
$$\vdots$$

value key

sort $\longrightarrow$
by key
in MPC

$$\vdots$$
$$C_2, \mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)$$
$$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(n_b)$$
$$\vdots$$
$$C_1, \mathrm{Enc}_{\mathsf{pk}}(\boldsymbol{c_1}), \mathrm{Enc}_{\mathsf{pk}}(0)$$
$$C_3, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)$$
$$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)$$
$$\vdots$$
$$C_4, \mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)$$
$$\vdots$$

equal? $\rightarrow$ same credential
equal to n_b? $\rightarrow$ valid credential

CHide: a quasi-linear solution based on sorting

$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$
$$\vdots$$
$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$

$$\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)}$$
$$\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)}$$
$$\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)}$$
$$\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)}$$
$$\vdots$$

sort $\longrightarrow$
by key
in MPC

$$\vdots$$
$$\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)}$$
$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$
$$\vdots$$
$$\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\boldsymbol{c_1}), \mathrm{Enc}_{\mathsf{pk}}(0)}$$
$$\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\boldsymbol{c_1}), \mathrm{Enc}_{\mathsf{pk}}(2)}$$
$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$
$$\vdots$$
$$\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)}$$
$$\vdots$$

value key

equal? $\rightarrow$ same credential
equal to n_b? $\rightarrow$ valid credential

CHide: a quasi-linear solution based on sorting

$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$

$\vdots$

$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$

$\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)}$
$\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)}$
$\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)}$
$\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)}$

$\vdots$

sort $\longrightarrow$
by key
in MPC

$\vdots$

$\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)}$
$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$

$\vdots$

$\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)}$
$\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\boldsymbol{c_1}), \mathrm{Enc}_{\mathsf{pk}}(2)}$
$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\boldsymbol{c_1}), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$

$\vdots$

$\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)}$

$\vdots$

value key

equal? $\rightarrow$ same credential
equal to n_b? $\rightarrow$ valid credential

CHide: a quasi-linear solution based on sorting

$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$
$$\vdots$$
$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$

$$\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)}$$
$$\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)}$$
$$\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)}$$
$$\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)}$$
$$\vdots$$

sort $\longrightarrow$
by key
in MPC

$$\vdots$$
$$\underline{C_2}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)}$$
$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$
$$\vdots$$
$$\underline{C_1}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)}$$
$$\underline{C_3}, \underline{\mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)}$$
$$\underline{\mathrm{Enc}_{\mathsf{pk}}(0)}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\boldsymbol{c_1}), \mathrm{Enc}_{\mathsf{pk}}(n_b)}$$
$$\vdots$$
$$\underline{C_4}, \underline{\mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)}$$
$$\vdots$$

value key

equal? $\rightarrow$ same credential
equal to n_b? $\rightarrow$ valid credential

CHide: a quasi-linear solution based on sorting

$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)$

$\vdots$

$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_{n_v}), \mathrm{Enc}_{\mathsf{pk}}(n_b)$

$C_1, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)$
$C_2, \mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)$
$C_3, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)$
$C_4, \mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)$

$\vdots$

sort $\longrightarrow$
by key
in MPC

$\vdots$

$C_2, \mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(1)$
$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_2), \mathrm{Enc}_{\mathsf{pk}}(n_b)$

$\vdots$

$C_1, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(0)$
$C_3, \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(2)$
$\mathrm{Enc}_{\mathsf{pk}}(0), \mathrm{Enc}_{\mathsf{pk}}(c_1), \mathrm{Enc}_{\mathsf{pk}}(n_b)$

$\vdots$

$C_4, \mathrm{Enc}_{\mathsf{pk}}(\tilde{c}), \mathrm{Enc}_{\mathsf{pk}}(3)$

$\vdots$

value key

equal? $\rightarrow$ same credential
equal to n_b? $\rightarrow$ valid credential

Sorting with the toolbox: OddEvenMergeSort

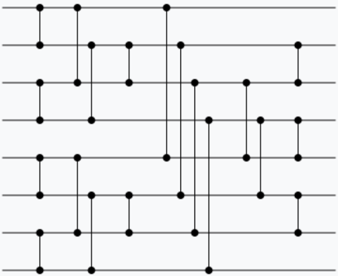
Batcher's algorithm (1968):

OddEvenMS($a_1, \cdots, a_n$) (iterative)

$t \leftarrow \lceil \log n \rceil$; $p \leftarrow 2^{t-1}$;
while $p > 0$ **do**
    $q \leftarrow 2^{t-1}$; $r \leftarrow 0$; $d \leftarrow p$;
    while $d > 0$ **do**
        for $i = 0$ *to* $n - d - 1$ **do**
            if $\mathrm{BitwiseAnd}(i, p) = r$ **then**
                $\mathrm{CompSwap}(i, i + d)$;

From The art of computer programming, vol. 3 (D. Knuth)

- Data oblivious
- Efficient
- Parallelizable



Visualization of the odd–even mergesort network with eight inputs

Class	Sorting algorithm
Data structure	Array
Worst-case performance	$O(\log^2(n))$ parallel time
Best-case performance	$O(\log^2(n))$ parallel time
Average performance	$O(\log^2(n))$ parallel time
Worst-case space complexity	$O(n \log^2(n))$ non-parallel time

From Wikipedia

Defining coercion-resistance - The JCJ definition

(Simplified version)

Real game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
$\mathrm{PB} \leftarrow (\mathrm{EncBallot}(\nu))_{\nu \in B}$;
$\tilde{c} \leftarrow c_V$;
if $b = 0$ *(evasion strategy)* **then**
  $\tilde{c} \leftarrow \mathrm{Fakecred}(c_V)$;
  $\mathrm{PB} \leftarrow \mathrm{PB} || \mathrm{EncBallot}(\nu_{\mathrm{intent}})$;
$\boldsymbol{M} \longleftarrow \mathbb{A}(\mathrm{PB}, \tilde{c})$;
$\mathrm{PB} \leftarrow \mathrm{PB} || \{\mathrm{b} \in \boldsymbol{M} \mid \mathrm{b\ is\ valid}\}$;

$r, \Pi \leftarrow \mathrm{tally}(\mathrm{PB})$;
$\mathrm{guess} \leftarrow \mathbb{A}(r, \Pi)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

Ideal game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
if $b = 0$ **then**
  $B \leftarrow B || \nu_{\mathrm{intent}}$;
$(\nu_i)_{i \in \mathrm{corrupted}}, \nu_{\mathrm{instruct}} \leftarrow \mathbb{A}(|B|)$;
$B \leftarrow B || (\nu_i)_{i \in \mathrm{corrupted}}$;
if $b = 1$ **then** $B \leftarrow B || \nu_{\mathrm{instruct}}$;
$r \leftarrow \mathrm{result}(B)$;
$\mathrm{guess} \leftarrow \mathbb{A}(r)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

Defining coercion-resistance - The JCJ definition

(Simplified version)

Real game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
$\mathrm{PB} \leftarrow (\mathrm{EncBallot}(\nu))_{\nu \in B}$;
$\tilde{c} \leftarrow c_V$;
if $b = 0$ *(evasion strategy)* **then**
  $\tilde{c} \leftarrow \mathrm{Fakecred}(c_V)$;
  $\mathrm{PB} \leftarrow \mathrm{PB}||\mathrm{EncBallot}(\nu_{\mathrm{intent}})$;
$\boldsymbol{M} \longleftarrow \mathbb{A}(\mathrm{PB},\tilde{c})$;
$\mathrm{PB} \leftarrow \mathrm{PB}||\{\mathrm{b} \in \boldsymbol{M} \mid \mathrm{b\ is\ valid}\}$;
$r, \Pi \leftarrow \mathrm{tally}(\mathrm{PB})$;
$\mathrm{guess} \leftarrow \mathbb{A}(r, \Pi)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

Ideal game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
if $b = 0$ **then**
  $B \leftarrow B||\nu_{\mathrm{intent}}$;
$(\nu_i)_{i \in \mathrm{corrupted}}, \nu_{\mathrm{instruct}} \leftarrow \mathbb{A}(|B|)$;
$B \leftarrow B||(\nu_i)_{i \in \mathrm{corrupted}}$;
if $b = 1$ **then** $B \leftarrow B||\nu_{\mathrm{instruct}}$;
$r \leftarrow \mathrm{result}(B)$;
$\mathrm{guess} \leftarrow \mathbb{A}(r)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

The adversary ($\mathbb{A}$) must guess the behavior of the coerced voter (V).

Defining coercion-resistance - The JCJ definition

(Simplified version)

Real game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
$\mathrm{PB} \leftarrow (\mathrm{EncBallot}(\nu))_{\nu \in B}$;
$\tilde{c} \leftarrow c_V$;
if $b = 0$ *(evasion strategy)* **then**
  $\tilde{c} \leftarrow \mathrm{Fakecred}(c_V)$;
  $\mathrm{PB} \leftarrow \mathrm{PB}||\mathrm{EncBallot}(\nu_{\mathrm{intent}})$;
$\boldsymbol{M} \longleftarrow \mathbb{A}(\mathrm{PB},\tilde{c})$;
$\mathrm{PB} \leftarrow \mathrm{PB}||\{\mathrm{b} \in \boldsymbol{M} \mid \mathrm{b\ is\ valid}\}$;
$r, \Pi \leftarrow \mathrm{tally}(\mathrm{PB})$;
$\mathrm{guess} \leftarrow \mathbb{A}(r, \Pi)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

Ideal game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
if $b = 0$ **then**
  $B \leftarrow B||\nu_{\mathrm{intent}}$;
$(\nu_i)_{i \in \mathrm{corrupted}}, \nu_{\mathrm{instruct}} \leftarrow \mathbb{A}(|B|)$;
$B \leftarrow B||(\nu_i)_{i \in \mathrm{corrupted}}$;
if $b = 1$ **then** $B \leftarrow B||\nu_{\mathrm{instruct}}$;
$r \leftarrow \mathrm{result}(B)$;
$\mathrm{guess} \leftarrow \mathbb{A}(r)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

The adversary ($\mathbb{A}$) must guess the behavior of the coerced voter (V).

Defining coercion-resistance - The JCJ definition

(Simplified version)

Real game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
$\mathrm{PB} \leftarrow (\mathrm{EncBallot}(\nu))_{\nu \in B}$;
$\tilde{c} \leftarrow c_V$;
if $b = 0$ *(evasion strategy)* **then**
  $\tilde{c} \leftarrow \mathrm{Fakecred}(c_V)$;
  $\mathrm{PB} \leftarrow \mathrm{PB}||\mathrm{EncBallot}(\nu_{\mathrm{intent}})$;
$M \longleftarrow \mathbb{A}(\mathrm{PB}, \tilde{c})$;
$\mathrm{PB} \leftarrow \mathrm{PB}||\{\mathrm{b} \in M \mid \mathrm{b\ is\ valid}\}$;
$r, \Pi \leftarrow \mathrm{tally}(\mathrm{PB})$;
$\mathrm{guess} \leftarrow \mathbb{A}(r, \Pi)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

Ideal game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
if $b = 0$ **then**
  $B \leftarrow B||\nu_{\mathrm{intent}}$;
$(\nu_i)_{i \in \mathrm{corrupted}}, \nu_{\mathrm{instruct}} \leftarrow \mathbb{A}(|B|)$;
$B \leftarrow B||(\nu_i)_{i \in \mathrm{corrupted}}$;
if $b = 1$ **then** $B \leftarrow B||\nu_{\mathrm{instruct}}$;
$r \leftarrow \mathrm{result}(B)$;
$\mathrm{guess} \leftarrow \mathbb{A}(r)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

The adversary ($\mathbb{A}$) must guess the behavior of the coerced voter (V).

Defining coercion-resistance - The JCJ definition

(Simplified version)

Real game	Ideal game
$b \xleftarrow{\$} \{0,1\}$;	$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;	$B \leftarrow \mathrm{HonestVotes}$;
$\mathrm{PB} \leftarrow (\mathrm{EncBallot}(\nu))_{\nu \in B}$;	
$\tilde{c} \leftarrow c_V$;	
if $b = 0$ *(evasion strategy)* **then**	**if** $b = 0$ **then**
$\quad \tilde{c} \leftarrow \mathrm{Fakecred}(c_V)$;	$\quad B \leftarrow B \| \nu_{\mathrm{intent}}$;
$\quad \mathrm{PB} \leftarrow \mathrm{PB} \| \mathrm{EncBallot}(\nu_{\mathrm{intent}})$;	
$M \longleftarrow \mathbb{A}(\mathrm{PB}, \tilde{c})$;	$(\nu_i)_{i \in \mathrm{corrupted}}, \nu_{\mathrm{instruct}} \leftarrow \mathbb{A}(\|B\|)$;
$\mathrm{PB} \leftarrow \mathrm{PB} \| \{\mathrm{b} \in M \mid \mathrm{b\ is\ valid}\}$;	$B \leftarrow B \| (\nu_i)_{i \in \mathrm{corrupted}}$;
	if $b = 1$ **then** $B \leftarrow B \| \nu_{\mathrm{instruct}}$;
$r, \Pi \leftarrow \mathrm{tally}(\mathrm{PB})$;	$r \leftarrow \mathrm{result}(B)$;
$\mathrm{guess} \leftarrow \mathbb{A}(r, \Pi)$;	$\mathrm{guess} \leftarrow \mathbb{A}(r)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;	**if** $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;	**else** $\mathbb{A}$ loses;

The coerced voter's behavior influences the result.

Defining coercion-resistance - The JCJ definition

(Simplified version)

Real game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
$\mathrm{PB} \leftarrow (\mathrm{EncBallot}(\nu))_{\nu \in B}$;
$\tilde{c} \leftarrow c_V$;
if $b = 0$ *(evasion strategy)* **then**
  $\tilde{c} \leftarrow \mathrm{Fakecred}(c_V)$;
  $\mathrm{PB} \leftarrow \mathrm{PB}||\mathrm{EncBallot}(\nu_{\mathrm{intent}})$;
$M \longleftarrow \mathbb{A}(\mathrm{PB},\tilde{c})$;
$\mathrm{PB} \leftarrow \mathrm{PB}||\{\mathrm{b} \in M \mid \mathrm{b\ is\ valid}\}$;
$r, \Pi \leftarrow \mathrm{tally}(\mathrm{PB})$;
$\mathrm{guess} \leftarrow \mathbb{A}(r, \Pi)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

Ideal game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
if $b = 0$ **then**
  $B \leftarrow B||\nu_{\mathrm{intent}}$;
$(\nu_i)_{i \in \mathrm{corrupted}}, \nu_{\mathrm{instruct}} \leftarrow \mathbb{A}(|B|)$;
$B \leftarrow B||(\nu_i)_{i \in \mathrm{corrupted}}$;
if $b = 1$ **then** $B \leftarrow B||\nu_{\mathrm{instruct}}$;
$r \leftarrow \mathrm{result}(B)$;
$\mathrm{guess} \leftarrow \mathbb{A}(r)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

The result also depends on the honest voters.

Defining coercion-resistance - The JCJ definition

(Simplified version)

Real game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
$\mathrm{PB} \leftarrow (\mathrm{EncBallot}(\nu))_{\nu \in B}$;
$\tilde{c} \leftarrow c_V$;
if $b = 0$ *(evasion strategy)* **then**
  $\tilde{c} \leftarrow \mathrm{Fakecred}(c_V)$;
  $\mathrm{PB} \leftarrow \mathrm{PB} || \mathrm{EncBallot}(\nu_{\mathrm{intent}})$;
$\boldsymbol{M} \longleftarrow \mathbb{A}(\mathrm{PB}, \tilde{c})$;
$\mathrm{PB} \leftarrow \mathrm{PB} || \{\mathrm{b} \in \boldsymbol{M} \mid \mathrm{b\ is\ valid}\}$;
$r, \Pi \leftarrow \mathrm{tally}(\mathrm{PB})$;
$\mathrm{guess} \leftarrow \mathbb{A}(r, \Pi)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

Ideal game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \mathrm{HonestVotes}$;
if $b = 0$ **then**
  $B \leftarrow B || \nu_{\mathrm{intent}}$;
$(\nu_i)_{i \in \mathrm{corrupted}}, \nu_{\mathrm{instruct}} \leftarrow \mathbb{A}(|B|)$;
$B \leftarrow B || (\nu_i)_{i \in \mathrm{corrupted}}$;
if $b = 1$ **then** $B \leftarrow B || \nu_{\mathrm{instruct}}$;
$r \leftarrow \mathrm{result}(B)$;
$\mathrm{guess} \leftarrow \mathbb{A}(r)$;
if $\mathrm{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

The result also depends on the adversary's votes.

Defining coercion-resistance - The JCJ definition

(Simplified version)

Real game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \text{HonestVotes}$;
$\text{PB} \leftarrow (\text{EncBallot}(\nu))_{\nu \in B}$;
$\tilde{c} \leftarrow c_V$;
if $b = 0$ *(evasion strategy)* **then**
  $\tilde{c} \leftarrow \text{Fakecred}(c_V)$;
  $\text{PB} \leftarrow \text{PB} || \text{EncBallot}(\nu_{\text{intent}})$;
$\boldsymbol{M} \longleftarrow \mathbb{A}(\text{PB},\tilde{c})$;
$\text{PB} \leftarrow \text{PB} || \{\text{b} \in \boldsymbol{M} \mid \text{b is valid}\}$;

$r, \Pi \leftarrow \text{tally}(\text{PB})$;
$\text{guess} \leftarrow \mathbb{A}(r, \Pi)$;
if $\text{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

Ideal game

$b \xleftarrow{\$} \{0,1\}$;
$B \leftarrow \text{HonestVotes}$;
if $b = 0$ **then**
  $B \leftarrow B || \nu_{\text{intent}}$;
$(\nu_i)_{i \in \text{corrupted}}, \nu_{\text{instruct}} \leftarrow \mathbb{A}(|B|)$;
$B \leftarrow B || (\nu_i)_{i \in \text{corrupted}}$;
if $b = 1$ **then** $B \leftarrow B || \nu_{\text{instruct}}$;
$r \leftarrow \text{result}(B)$;
$\text{guess} \leftarrow \mathbb{A}(r)$;
if $\text{guess} = b$ **then** $\mathbb{A}$ wins;
else $\mathbb{A}$ loses;

But how are the honest votes chosen?

How are the honest votes chosen?

- Only **one** distribution $\mathrm{HonestVotes}$ is considered.

How are the honest votes chosen?

- Only **one** distribution $\mathrm{HonestVotes}$ is considered.
- $\mathrm{HonestVotes}$ does not allow revoting!

How are the honest votes chosen?

- Only **one** distribution $\mathrm{HonestVotes}$ is considered.
- $\mathrm{HonestVotes}$ does not allow revoting!

Problem: no revotes $\implies$ $\#\mathit{duplicates} = 0$.

$$\#\mathit{duplicates} + \#\mathit{invalid} = \mathit{size}(\mathit{board}) - \mathit{size}(\mathit{result}).$$

The JCJ definition is flawed! (It does not capture the leakage in JCJ.)

Defining coercion-resistance - The (flawed) JCJ definition

How are the honest votes chosen?

- Only **one** distribution $\mathrm{HonestVotes}$ is considered.
- $\mathrm{HonestVotes}$ does not allow revoting!

Problem: no revotes $\implies$ $\#\mathit{duplicates} = 0$.

$$\#\mathit{duplicates} + \#\mathit{invalid} = \mathit{size}(\mathit{board}) - \mathit{size}(\mathit{result}).$$

The JCJ definition is flawed! (It does not capture the leakage in JCJ.)

Solution:

- Consider a larger **family** of honest behaviors (that allow revoting).
- Coercion-resistance must be achieved for **all** distribution $\mathcal{B}$.

Our definition of coercion-resistance

Algorithm 101: $\mathsf{Real}^{\mathsf{CR}}$

Requires: $\mathbb{A}, \lambda, n_T, C_t, n_V, n_A, n_C, \mathcal{B}$

$\mathsf{pk}, (s_i, h_i)_{i=1}^{n_T}, \Pi^S \longleftarrow \mathsf{Setup}(\lambda, n_T, t)$;
$(c_i, \pi_i), \Pi^R \longleftarrow \mathsf{Register}(\mathsf{pk}, n_V)$;
$\mathsf{PB} \longleftarrow \Pi^S || \Pi^R$;
$A \longleftarrow \mathbb{A}(\mathsf{PB}, \{s_i \mid i \in C_t\})$ (* corrupt voters *);
$(j, \alpha) \longleftarrow \mathbb{A}(\{c_i; i \in A\})$;
(* coerces j who has the intention α *)
if $|A| \neq n_A \vee j \notin [1, n_V] \backslash A \vee \alpha \notin [1, n_C] \bigcup \{\phi\}$ **then return** 0 ;
$B \longleftarrow \mathcal{B}([1, n_V] \backslash A, n_C)$;
(* samples a sequence of pairs (i, ν_i) with $i \in ([1, n_V] \backslash A) \bigcup \{n \mid n < 0\}$ *)
for $(i, *) \in B,\ i \notin [1, n_V]$ **do**
    $c_i \longleftarrow \mathsf{Fakecred}()$;
    (* this captures dummy ballots *)
$b \xleftarrow{\$} \{0, 1\}$;
$\tilde{c} \longleftarrow c_j$;
if $b = 1$ **then** Remove all $(j, *) \in B$;
else
    Remove all $(j, *) \in B$ but the last, which is replaced by (j, α) if $\alpha \neq \phi$ and removed otherwise;
    $\tilde{c} \longleftarrow \mathsf{Fakecred}(c_j)$;
$\mathbb{A}(\tilde{c})$ (* $\mathbb{A}$ learns $\tilde{c}$ *);
for $(i, \nu_i) \in B$ *(in this order)* **do**
    $\mathbb{A}^{\mathcal{O}_{\mathsf{cast}}}(\mathsf{PB})$ (* casts valid ballots *);
    $\mathsf{PB} \leftarrow \mathsf{PB} \bigcup \{\mathsf{Vote}_{\mathsf{pk}}(c_i, \nu_i)\}$;
$\mathbb{A}^{\mathcal{O}_{\mathsf{cast}}}(\mathsf{PB}, \texttt{"end for"})$;
$X, \Pi \leftarrow \mathsf{Tally}^{\mathbb{A}}(\mathsf{PB}, \mathsf{pk}, \{s_i\})$;
$b' \longleftarrow \mathbb{A}()$;
if $b' = b$ **then return** 1 **else return** 0;

Algorithm 102: $\mathsf{Ideal}^{\mathsf{CR}}$

Requires: $\mathbb{A}, \lambda, n_V, n_A, n_C, \mathcal{B}$

;
;
;
$A \longleftarrow \mathbb{A}(\lambda)$ (* corrupt voters *);
$(j, \alpha) \longleftarrow \mathbb{A}()$;
(* coerces j who has the intention α *);
if $|A| \neq n_A \vee j \notin [1, n_V] \backslash A \vee \alpha \notin [1, n_C] \bigcup \{\phi\}$ **then return** 0 ;
$B \longleftarrow \mathcal{B}([1, n_V] \backslash A, n_C)$;
(* samples a sequence of pairs (i, ν_i) with $i \in ([1, n_V] \backslash A) \bigcup \{n \mid n < 0\}$ *);
;
;
;
$b \xleftarrow{\$} \{0, 1\}$;
;
if $b = 1$ **then** Remove all $(j, *) \in B$;
else
    Remove all $(j, *) \in B$ but the last, which is replaced by (j, α) if $\alpha \neq \phi$ and removed otherwise;
;
;
$(\nu_i)_{i \in A}, \beta \longleftarrow \mathbb{A}(|B|)$;
if $(b = 1) \wedge (\beta \in [1, n_C])$ **then**
    $B \longleftarrow B \bigcup \{(j, \beta)\}$;
$B \longleftarrow B \bigcup \{(i, \nu_i) \mid i \in A, \nu_i \in [1, n_C]\}$;
$X \longleftarrow \mathsf{count}(\mathsf{cleanse}(B))$;
$b' \longleftarrow \mathbb{A}(X)$;
if $b' = b$ **then return** 1 **else return** 0;

Our definition of coercion-resistance

Algorithm 101: $\mathsf{Real}^{\mathsf{CR}}$

Requires: $\mathbb{A}, \lambda, n_T, C_t, n_V, n_A, n_C, \mathcal{B}$

```
pk, (s_i, h_i)_{i=1}^{n_T}, Π^S ⟵ Setup(λ, n_T, t);
(c_i, π_i), Π^R ⟵ Register(pk, n_V);
PB ⟵ Π^S || Π^R;
A ⟵ 𝔸(PB, {s_i | i ∈ C_t}) (* corrupt voters *);
(j, α) ⟵ 𝔸({c_i; i ∈ A});
(* coerces j who has the intention α *)
if |A| ≠ n_A ∨ j ∉ [1, n_V]\A ∨ α ∉ [1, n_C] ⋃ {φ} then return 0 ;
B ⟵ 𝓑([1, n_V]\A, n_C);
(* samples a sequence of pairs (i, ν_i) with i ∈ ([1, n_V]\A) ⋃ {n | n < 0} *)
for (i, *) ∈ B, i ∉ [1, n_V] do
    c_i ⟵ Fakecred();
...
    Remove all (j, *) ∈ B but the last, which is replaced by (j, α) if α ≠ φ and removed otherwise;
    c̃ ⟵ Fakecred(c_j);
𝔸(c̃) (* 𝔸 learns c̃ *);
for (i, ν_i) ∈ B (in this order) do
    𝔸^{O_cast}(PB) (* casts valid ballots *);
    PB ← PB ⋃ {Vote_pk(c_i, ν_i)};
𝔸^{O_cast}(PB, "end for");
X, Π ← Tally^𝔸(PB, pk, {s_i});
b' ⟵ 𝔸();
if b' = b then return 1 else return 0;
```

Algorithm 102: $\mathsf{Ideal}^{\mathsf{CR}}$

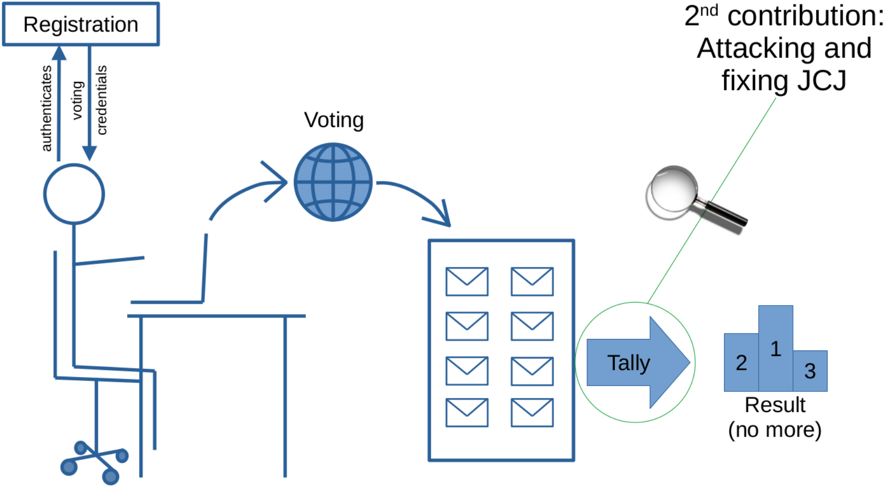
Requires: $\mathbb{A}, \lambda, n_V, n_A, n_C, \mathcal{B}$

```
;
;
;
A ⟵ 𝔸(λ) (* corrupt voters *);
(j, α) ⟵ 𝔸();
(* coerces j who has the intention α *);
if |A| ≠ n_A ∨ j ∉ [1, n_V]\A ∨ α ∉ [1, n_C] ⋃ {φ} then return 0 ;
B ⟵ 𝓑([1, n_V]\A, n_C);
(* samples a sequence of pairs (i, ν_i) with i ∈ ([1, n_V]\A) ⋃ {n | n < 0} *);
;
;
...
    which is replaced by (j, α) if α ≠ φ and removed otherwise;
;
;
(ν_i)_{i∈A}, β ⟵ 𝔸(|B|);
if (b = 1) ∧ (β ∈ [1, n_C]) then
    B ⟵ B ⋃ {(j, β)};
B ⟵ B ⋃ {(i, ν_i) | i ∈ A, ν_i ∈ [1, n_C]};
X ⟵ count(cleanse(B));
b' ⟵ 𝔸(X);
if b' = b then return 1 else return 0;
```

Theorem

CHide is coercion-resistant. JCJ is not.

Our contributions



Conclusion

Tally hiding	Coercion-resistance	Vote-buying
• A toolbox for MPC in ElGamal	• Uncover a leakage in the JCJ scheme	• A receipt-free voting protocol
• Applied to various counting methods	• Fixed with CHide	• A new definition of receipt-freeness
• Security proofs in the UC framework	• A new definition of coercion-resistance	• Security proofs
	• Proof that CHide is coercion-resistant	

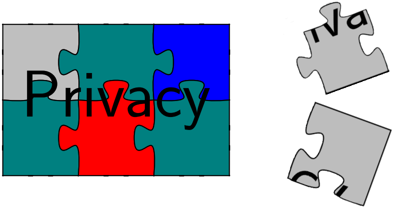
Conclusion

Tally hiding	Coercion-resistance	Vote-buying
• A toolbox for MPC in ElGamal	• Uncover a leakage in the JCJ scheme	• A receipt-free voting protocol
• Applied to various counting methods	• Fixed with CHide	• A new definition of receipt-freeness
• Security proofs in the UC framework	• A new definition of coercion-resistance	• Security proofs
	• Proof that CHide is coercion-resistant	

Design of protocols Security proofs Security definitions

Future works

- **More modularity for security definitions**



Future works

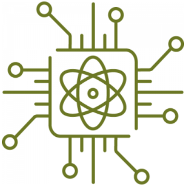
- More modularity for security definitions
- **Accountability and dispute resolution**

Future works

- More modularity for security definitions
- Accountability and dispute resolution
- **Registration and eligibility**

Future works

- More modularity for security definitions
- Accountability and dispute resolution
- Registration and eligibility
- **Post-quantum electronic voting**



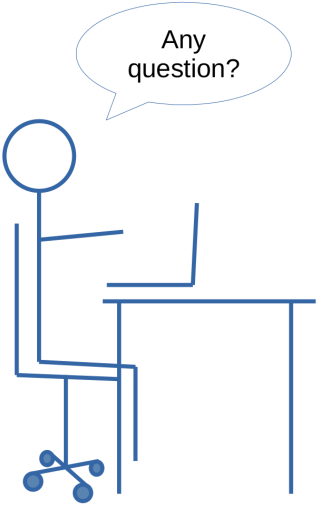
Thank you!

Publications

- A toolbox for verifiable tally-hiding e-voting systems. ESORICS 2022.
- How to fake zero-knowledge proofs, again. E-Vote-ID 2020 (short paper).

Work in progress

- Is the JCJ voting system really coercion-resistant?
- End-to-End Verifiable Receipt-Free E-Voting. Henri Devillez, Thomas Peters, Olivier Pereira, Quentin Yang.



Odd-Even Merge Sort

back

Recursive algorithm, assuming the length is a power of 2.

70 28 49 49 85 66 46 36 20 0 92 60 80 34 29 79

Odd-Even Merge Sort

back

Recursive algorithm, assuming the length is a power of 2.

70 28 49 49 85 66 46 36 | 20 0 92 60 80 34 29 79

Odd-Even Merge Sort

back

Recursive algorithm, assuming the length is a power of 2.

70 28 49 49 85 66 46 36 | 20 0 92 60 80 34 29 79

$\downarrow$ recursive call $\qquad$ $\downarrow$ recursive call

28 36 46 49 49 66 70 85 | 0 20 29 34 60 79 80 92

Odd-Even Merge Sort

back

Recursive algorithm, assuming the length is a power of 2.

70 28 49 49 85 66 46 36 | 20 0 92 60 80 34 29 79

$\downarrow$ recursive call　　$\downarrow$ recursive call

28 36 46 49 49 66 70 85 0 20 29 34 60 79 80 92

Odd-Even Merge Sort

back

Recursive algorithm, assuming the length is a power of 2.

70 28 49 49 85 66 46 36 | 20 0 92 60 80 34 29 79

recursive call   recursive call

28 36 46 49 49 66 70 85 0 20 29 34 60 79 80 92

merge odd   merge even

0 20 28 34 29 36 46 49 49 66 60 79 70 85 80 92

Odd-Even Merge Sort

back

Recursive algorithm, assuming the length is a power of 2.

70 28 49 49 85 66 46 36 | 20 0 92 60 80 34 29 79

recursive call　　recursive call

28 36 46 49 49 66 70 85 0 20 29 34 60 79 80 92

merge odd　　merge even

0 20 28 34 29 36 46 49 49 66 60 79 70 85 80 92

pairwise comparisons

0 20 28 29 34 36 46 49 49 60 66 70 79 80 85 92

Odd-Even Merge

back

Recursive algorithm, assuming the length is a power of 2.

28 36 46 49 49 66 70 85 0 20 29 34 60 79 80 92

back

Recursive algorithm, assuming the length is a power of 2.

28 36 46 49 49 66 70 85 0 20 29 34 60 79 80 92

Odd-Even Merge

back

Recursive algorithm, assuming the length is a power of 2.

28 36 46 49 49 66 70 85 0 20 29 34 60 79 80 92

↓ merge odd ↓ merge even

0 36 29 49 28 66 46 85 49 20 70 34 60 79 80 92

Odd-Even Merge

back

Recursive algorithm, assuming the length is a power of 2.

28 36 46 49 49 66 70 85 0 20 29 34 60 79 80 92

merge odd   merge even

0 36 29 49 28 66 46 85 49 20 70 34 60 79 80 92

pairwise comparisons

0 36 28 49 29 66 46 85 49 20 60 34 70 79 80 92

Computing the logical AND with cards.

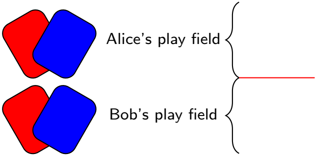
Material:

- Three identical red cards (with a hidden side)
- Two identical blue cards (with a hidden side)

Setting:

- Two participants with each a secret input bit
- One red card is put on the table (face down)

back

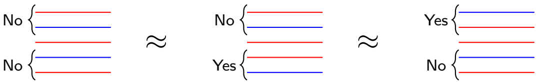


Computing the logical AND with cards.

Step 1. Commit: put your cards (face down) so that the red cards touch (yes) / do not touch (no) each other.

Step 2. Mask: each participant cut the deck any number of times.

Step 3. Reveal: check if there are three “consecutive” red cards.



Indistinguishable up to a circular permutation!

back

The Paillier encryption scheme

back

Public key: $n = pq$, a strong RSA modulo.

Secret key: s, congruent to 1 moodulo n and to 0 modulo $\varphi(n)$.

Encryption: To encrypt $m \in \mathbb{Z}_n$, pick $r \xleftarrow{\$} \mathbb{Z}_{n^2}^\times$ and compute $C = (1+n)^m r^n \bmod n^2$.

Decryption: To decrypt C, compute $D = C^s \bmod n^2$ and $m = (D-1)/n$.

DCRA: The nth residues modulo n^2 are indistinguishable from random.