# Quantifying Voter-controlled Privacy

Hugo Jonker

*in collaboration with Sjouke Mauw and Jun Pang*

hugo.jonker@uni.lu

SaToSS group, University of Luxembourg

# who am I

- PhD from TU/e, UL
  - TU/e: Formal Methods group (Baeten)
  - UL: Security and Trust of Software Systems group (Mauw)
  - Focusing on privacy in voting, DRM systems
- organised VoteID2009 in Luxembourg

privacy = tricky

# Dutch elections

Dutch ballot:

| 1. CDA | | | ... | 18. SGP | | |
|---|---|---|---|---|---|---|
| 1-1. | X | ☐ | ... | 18-1. | X' | ☐ |
| | ⋮ | | ⋮ | | ⋮ | |
| 1-13. | Y | ☐ | | 18-13. | Y' | ☐ |
| | ⋮ | | ⋮ | | | |
| 1-45. | Z | ☐ | ... | | | |

**Parties:** CDA, VVD, PvdA, SP, Groenlinks, Wilders, LPF, Christenunie, SGP, ...

# lesson

- Privacy is more than “for whom you voted”.

# Luxembourgian elections

Luxembourgian ballot:

| | | |
|---|---|---|
| 1. ADR | ... | 7. KPL |
| 1-1. J. Henckes ☐ ☐ | ... | 7-1. P. Back ☐ ☐ |
| ⋮ | ⋮ | ⋮ |
| 1-21. F. Zeutzius ☐ ☐ | ... | 7-21. M. Tani ☐ ☐ |

# Luxembourgian elections

Luxembourgian ballot:

| 1. ADR | ... | 7. KPL |
|---|---|---|
| 1-1. J. Henckes ☐ ☐ | ... | 7-1. P. Back ☐ ☐ |
| ⋮ | ⋮ | ⋮ |
| 1-21. F. Zeutzius ☐ ☐ | ... | 7-21. M. Tani ☐ ☐ |

- voter marks 21 boxes.

# Luxembourgian elections

Luxembourgian ballot:

| 1. ADR | ⋯ | 7. KPL |
|---|---|---|
| 1-1. J. Henckes ■ ■ | ⋯ | 7-1. P. Back □ □ |
| ⋮ | ⋮ | ⋮ |
| 1-21. F. Zeutzius □ □ | ⋯ | 7-21. M. Tani □ □ |

- voter marks 21 boxes.

# Luxembourgian elections

Luxembourgian ballot:

| 1. ADR | ⋯ | 7. KPL |
|---|---|---|
| 1-1. J. Henckes ■ ■ | ⋯ | 7-1. P. Back □ □ |
| ⋮ | ⋮ | ⋮ |
| 1-21. F. Zeutzius □ □ | ⋯ | 7-21. M. Tani □ □ |

- voter marks 21 boxes.
- pick 2. That leaves $\binom{292}{19} =$ 314,269,098,408,967,151,724,980,483,800 ways to fill in ballot.

# lesson

- Privacy is more than “for whom you voted”.
- Privacy depends on all knowledge you have.

# helpful voters

# lessons / approach

- Privacy is more than “for whom you voted”.
- Privacy depends on all knowledge you have.
- Subjects may seek to reduce/renounce privacy.

# lessons / approach

- Privacy is more than “for whom you voted”.
- Privacy depends on all knowledge you have.
- Subjects may seek to reduce/renounce privacy.

$\Longrightarrow$ quantify privacy

# lessons / approach

- Privacy is more than “for whom you voted”.
- Privacy depends on all knowledge you have.
- Subjects may seek to reduce/renounce privacy.

$\Longrightarrow$ quantify privacy

$\Longrightarrow$ base privacy on intruder knowledge

# lessons / approach

- Privacy is more than “for whom you voted”.
- Privacy depends on all knowledge you have.
- Subjects may seek to reduce/renounce privacy.

$\Longrightarrow$ quantify privacy

$\Longrightarrow$ base privacy on intruder knowledge

$\Longrightarrow$ take conspiring voters into account

# outline



- Model systems in process algebra
- Determine system behaviour
- Determine privacy of behaviour

# outline



- Model systems in process algebra
- Determine system behaviour
- Determine privacy of behaviour

*Note*
conspiring behaviour is trivially distinguishable from non-conspiring behaviour

# setting



- 1 voter, 1 vote.
- every vote has equal weight.
- election process is phased.
- voter's vote is independent of voting system.

# terms



- voters $v \in \mathcal{V}$, candidates $c \in \mathcal{C}$
- choice function $\gamma : \mathcal{V} \to \mathcal{C}$
- var $\in$ Vars, $k \in Keys$, $n \in Nonces$
- pairing, encryption

Terms: $\varphi ::= \text{var} \mid c \mid n \mid k \mid (\varphi_1, \varphi_2) \mid \{\varphi\}_k.$

matching:

$$\text{match}(\varphi_{cl}, \varphi_o, \sigma) \equiv \sigma(\varphi_o) = \varphi_{cl} \wedge \text{dom}(\sigma) = \text{fv}(\varphi_o).$$

# events



Phases, communication of terms:

$$\begin{aligned} Ev = \{ \quad & s(a,a',\varphi), r(a,a',\varphi), \\ & as(a,a',\varphi), ar(a',\varphi), \\ & us(a,a',\varphi), ur(a,a',\varphi), \\ & ph(i) \\ & \mid a,a' \in Agents, \varphi \in Terms, i \in \mathbb{N}\}. \end{aligned}$$

# processes



Process:

$$P \quad ::= \quad \delta \mid ev.P \mid P_1 + P_2 \mid$$
$$P_1 \lhd \varphi_1 = \varphi_2 \rhd P_2 \mid ev.X(\varphi_1, \ldots, \varphi_n).$$

where $X(\mathsf{var}_1, \ldots, \mathsf{var}_n) = P$.

# processes



Process:

$$P \quad ::= \quad \delta \mid ev.P \mid P_1 + P_2 \mid P_1 \lhd \varphi_1 = \varphi_2 \rhd P_2 \mid ev.X(\varphi_1, \ldots, \varphi_n).$$

where $X(\mathsf{var}_1, \ldots, \mathsf{var}_n) = P$.

State of agent is knowledge + process:

$$Agstate = \mathcal{P}(Terms) \times Processes.$$

# voting system



**Definition 1 (voting system)** *A voting system $\mathcal{VS} \in VotSys$ specifies the state of each agent:*

$$VotSys = Agents \to Agstate.$$

*Instantiation of a voting system $\mathcal{VS} \in VotSys$ with choice function $\gamma$ is denoted as $\mathcal{VS}^{\gamma}$.*

$$\mathcal{VS}^{\gamma}(a) = \begin{cases} \mathcal{VS}(a) & \text{if } a \notin \mathcal{V} \\ (\ \pi_1(\mathcal{VS}(a)), \sigma_a(\pi_2(\mathcal{VS}(a)))\ ) & \text{if } a \in \mathcal{V} \end{cases}$$

*where* $\sigma_a = \mathsf{vc} \mapsto \gamma(a)$.

# approach



- agent semantics $\Longrightarrow$ system semantics
- system semantics $\Longrightarrow$ LTS
- paths in LTS $\Longrightarrow$ traces
- set of traces specifies the behaviour of the system

$$State = \mathcal{P}(Terms) \times (Agents \rightarrow Agstate).$$

Agent state: $(K_I, knw_a, P_a)$.
Initial system state: $(K_I^0, \mathcal{VS}^\gamma)$.

# receiving terms



Readability of terms:

$$\mathrm{Rd}(knw_a, \varphi_{cl}, \varphi_o, \sigma) \equiv \quad \mathrm{match}(\varphi_{cl}, \varphi_o, \sigma) \wedge \forall_{\varphi' \sqsubseteq \varphi_o} : knw_a \cup \{\varphi_{cl}\} \vdash \sigma(\varphi').$$

# receiving terms



## Readability of terms:

$$\mathrm{Rd}(knw_a, \varphi_{cl}, \varphi_o, \sigma) \equiv \mathrm{match}(\varphi_{cl}, \varphi_o, \sigma) \wedge \forall_{\varphi' \sqsubseteq \varphi_o} : knw_a \cup \{\varphi_{cl}\} \vdash \sigma(\varphi').$$

## Deconstruction of terms:

$$\varphi \sqsubseteq \varphi$$

$$\varphi_1 \sqsubseteq (\varphi_1, \varphi_2) \qquad \varphi_2 \sqsubseteq (\varphi_1, \varphi_2)$$

$$\varphi \sqsubseteq \{\varphi\}_k \qquad k^{-1} \sqsubseteq \{\varphi\}_k$$

# agent semantics (1)

- send:

$$\frac{knw_a \vdash \varphi \qquad \mathrm{fv}(\varphi) = \emptyset}{(K_I, knw_a, s(a,x,\varphi).P) \xrightarrow{s(a,x,\varphi)} (K_I \cup \{\varphi\}, knw_a, P)}$$

# agent semantics (1)

- send:

$$\frac{knw_a \vdash \varphi \qquad \mathrm{fv}(\varphi) = \emptyset}{(K_I, knw_a, s(a,x,\varphi).P) \xrightarrow{s(a,x,\varphi)} (K_I \cup \{\varphi\}, knw_a, P)}$$

- receive:

$$\frac{K_I \vdash \varphi' \qquad \mathrm{fv}(\varphi') = \emptyset \qquad \mathrm{Rd}(knw_a, \varphi', \varphi, \sigma)}{(K_I, knw_a, r(x,a,\varphi).P) \xrightarrow{r(x,a,\varphi')} (K_I, knw_a \cup \{\varphi'\}, \sigma(P))}$$

# agent semantics (1)

- send:

$$\frac{knw_a \vdash \varphi \qquad \mathrm{fv}(\varphi) = \emptyset}{(K_I, knw_a, s(a,x,\varphi).P) \xrightarrow{s(a,x,\varphi)} (K_I \cup \{\varphi\}, knw_a, P)}$$

- receive:

$$\frac{K_I \vdash \varphi' \qquad \mathrm{fv}(\varphi') = \emptyset \qquad \mathrm{Rd}(knw_a, \varphi', \varphi, \sigma)}{(K_I, knw_a, r(x,a,\varphi).P) \xrightarrow{r(x,a,\varphi')} (K_I, knw_a \cup \{\varphi'\}, \sigma(P))}$$

- anonymous receive:

$$\frac{K_I \vdash \varphi' \qquad \mathrm{fv}(\varphi') = \emptyset \qquad \mathrm{Rd}(knw_a, \varphi', \varphi, \sigma)}{(K_I, knw_a, ar(a,\varphi).P) \xrightarrow{ar(a,\varphi')} (K_I, knw_a \cup \{\varphi'\}, \sigma(P))}$$

# agent semantics (1)

- send:

$$\frac{knw_a \vdash \varphi \qquad \mathrm{fv}(\varphi) = \emptyset}{(K_I, knw_a, s(a,x,\varphi).P) \xrightarrow{s(a,x,\varphi)} (K_I \cup \{\varphi\}, knw_a, P)}$$

- receive:

$$\frac{K_I \vdash \varphi' \qquad \mathrm{fv}(\varphi') = \emptyset \qquad \mathrm{Rd}(knw_a, \varphi', \varphi, \sigma)}{(K_I, knw_a, r(x,a,\varphi).P) \xrightarrow{r(x,a,\varphi')} (K_I, knw_a \cup \{\varphi'\}, \sigma(P))}$$

- anonymous receive:

$$\frac{K_I \vdash \varphi' \qquad \mathrm{fv}(\varphi') = \emptyset \qquad \mathrm{Rd}(knw_a, \varphi', \varphi, \sigma)}{(K_I, knw_a, ar(a,\varphi).P) \xrightarrow{ar(a,\varphi')} (K_I, knw_a \cup \{\varphi'\}, \sigma(P))}$$

- untappable receive:

$$\frac{\mathrm{fv}(\varphi') = \emptyset \qquad \mathrm{Rd}(knw_a, \varphi', \varphi, \sigma)}{(K_I, knw_a, ur(x,a,\varphi).P) \xrightarrow{ur(x,a,\varphi')} (K_I, knw_a \cup \{\varphi'\}, \sigma(P))}$$

# agent semantics (2)

- non-deterministic choice:

$$\frac{(K_I, knw_a, P_1) \xrightarrow{ev} (K'_I, knw'_a, P'_1)}{(K_I, knw_a, P_1 + P_2) \xrightarrow{ev} (K'_I, knw'_a, P'_1)}$$

# agent semantics (2)

- non-deterministic choice:

$$\frac{(K_I, knw_a, P_1) \xrightarrow{ev} (K'_I, knw'_a, P'_1)}{(K_I, knw_a, P_1 + P_2) \xrightarrow{ev} (K'_I, knw'_a, P'_1)}$$

- conditional choice:

$$\frac{(K_I, knw_a, P_2) \xrightarrow{ev} (K'_I, knw'_a, P'_2) \qquad \varphi_1 \neq \varphi_2 \qquad \mathrm{fv}(\varphi_1) = \mathrm{fv}(\varphi_2) = \emptyset}{(K_I, knw_a, P_1 \lhd \varphi_1 = \varphi_2 \rhd P_2) \xrightarrow{ev} (K'_I, knw'_a, P'_2)}$$

# agent semantics (2)

- non-deterministic choice:

$$\frac{(K_I, knw_a, P_1) \xrightarrow{ev} (K'_I, knw'_a, P'_1)}{(K_I, knw_a, P_1 + P_2) \xrightarrow{ev} (K'_I, knw'_a, P'_1)}$$

- conditional choice:

$$\frac{(K_I, knw_a, P_2) \xrightarrow{ev} (K'_I, knw'_a, P'_2) \qquad \varphi_1 \neq \varphi_2 \qquad \mathrm{fv}(\varphi_1) = \mathrm{fv}(\varphi_2) = \emptyset}{(K_I, knw_a, P_1 \lhd \varphi_1 = \varphi_2 \rhd P_2) \xrightarrow{ev} (K'_I, knw'_a, P'_2)}$$

- guarded recursion:

$$\frac{\begin{array}{c}(K_I, knw_a, \sigma(P)) \xrightarrow{ev} (K'_I, knw'_a, P') \\ X(\mathsf{var}_1, \ldots, \mathsf{var}_n) = P \qquad \sigma = \mathsf{var}_1 \mapsto \varphi_1 \circ \cdots \circ \mathsf{var}_n \mapsto \varphi_n\end{array}}{(K_I, knw_a, X(\varphi_1, \ldots, \varphi_n)) \xrightarrow{ev} (K'_I, knw'_a, P')}$$

# agent semantics (3)

- phase synchronisation:

$$i \in \mathbb{N}$$

$$Phase \subseteq \{a \,@\, (knw_a, P_a) \in S \mid \exists P'_a \colon (K_I, knw_a, P_a) \xrightarrow{ph(i)} (K_I, knw_a, P'_a)\}$$

$$Aut \subseteq \{a \in Agents \mid \exists knw_a, P_a \colon a \,@\, (knw_a, P_a) \in Phase\}$$

$$Phase' = \{a \,@\, (knw_a, P'_a) \mid \exists P_a \colon a \,@\, (knw_a, P_a) \in Phase \wedge (K_I, knw_a, P_a) \xrightarrow{ph(i)} (K_I, knw_a, P'_a)\}$$

---

$$(K_I, S) \xrightarrow{ph(i)} (K_I, Phase' \cup S \setminus Phase)$$

# system semantics

- non-synchronous events:

$$\frac{(K_I, knw_a, P) \xrightarrow{ev} (K'_I, knw'_a, P') \qquad ev \in Ev_{nosync} \qquad a \,@\, (knw_a, P) \in S}{(K_I, S) \xrightarrow{ev} (K'_I, \{a \,@\, (knw'_a, P')\} \cup S \setminus \{a \,@\, (knw_a, P)\})}$$

# system semantics

- non-synchronous events:

$$\frac{(K_I, knw_a, P) \xrightarrow{ev} (K_I', knw_a', P') \qquad ev \in Ev_{nosync} \qquad a\,@\,(knw_a, P) \in S}{(K_I, S) \xrightarrow{ev} (K_I', \{a\,@\,(knw_a', P')\} \cup S \setminus \{a\,@\,(knw_a, P)\})}$$

- untappable communication:

$$\frac{\begin{array}{c}(K_I, knw_a, P_a) \xrightarrow{us(a,b,\varphi)} (K_I, knw_a, P_a') \\ (K_I, knw_b, P_b) \xrightarrow{ur(a,b,\varphi)} (K_I, knw_b', P_b') \\ s_0 = \{a\,@\,(knw_a, P_a), b\,@\,(knw_b, P_b)\} \qquad s_0 \subseteq S\end{array}}{(K_I, S) \xrightarrow{uc(a,b,\varphi)} (K_I, \{a\,@\,(knw_a, P_a'), b\,@\,(knw_b', P_b')\} \cup S \setminus s_0)}$$

# traces



$$Tr(\mathcal{VS}^\gamma) = \{\alpha \in Labels^\star \mid \alpha = \alpha_0, \ldots, \alpha_{n-1} \wedge \exists s_0, \ldots, s_n \in State : s_0 = (K_I^0, \mathcal{VS}^\gamma) \wedge \forall 0 \leq i < n : s_i \xrightarrow{\alpha_i} s_{i+1}\}$$

Traces of $\mathcal{VS}$:

$$Tr(\mathcal{VS}) = \bigcup_{\gamma \in \mathcal{V} \to \mathcal{C}} Tr(\mathcal{VS}^\gamma)$$

# how to



Can the intruder tell for $t$, if $t \in Tr(\mathcal{VS}^{\gamma_1})$ or $t \in Tr(\mathcal{VS}^{\gamma_2})$?

Our approach: reinterpretation.

# reinterpretation



**Definition 2 (reinterpretation (GHPR05))** *Let $\rho$ be a permutation on the set of terms $Terms$ and let $K_I$ be a knowledge set. The map $\rho$ is a semi-reinterpretation under $K_I$ if it satisfies the following.*

$$\begin{aligned} \rho(p) &= p, \text{ for } p \in \mathcal{C} \cup Keys \\ \rho((\varphi_1, \varphi_2)) &= (\rho(\varphi_1), \rho(\varphi_2)) \\ \rho(\{\varphi\}_k) &= \{\rho(\varphi)\}_k, \text{ if } K_I \vdash \varphi, k \vee K_I \vdash \{\varphi\}_k, k^{-1} \end{aligned}$$

*Map $\rho$ is a reinterpretation under $K_I$ iff it is a semi-reinterpretation and its inverse $\rho^{-1}$ is a semi-reinterpretation under $\rho(K_I)$.*

# indistinguishability



**Definition 3 (trace indistinguishability)** *Traces $t, t'$ are indistinguishable, $t \sim t'$, iff $\exists \rho$ such that*

$$obstr(t') = \rho(obstr(t)) \ \wedge \ \overline{K_I^t} = \rho(\overline{K_I^{t'}}).$$

**Definition 4 (choice indistinguishability)** *For voting system $\mathcal{VS}$, choice functions $\gamma_1, \gamma_2$ are indistinguishable, $\gamma_1 \simeq_{\mathcal{VS}} \gamma_2$, iff*

$$\forall t \in Tr(\mathcal{VS}^{\gamma_1}) : \exists t' \in Tr(\mathcal{VS}^{\gamma_2}) : t \sim t' \quad \wedge$$
$$\forall t \in Tr(\mathcal{VS}^{\gamma_2}) : \exists t' \in Tr(\mathcal{VS}^{\gamma_1}) : t \sim t'$$

**Definition 5 (choice group)** *The choice group for a voting system $\mathcal{VS}$ and a choice function $\gamma$ is given by*

$$cg(\mathcal{VS}, \gamma) = \{\gamma' \mid \gamma \simeq_{\mathcal{VS}} \gamma'\}.$$

*The choice group for a particular voter $v$:*

$$cg_v(\mathcal{VS}, \gamma) = \{\gamma'(v) \mid \gamma' \in cg(\mathcal{VS}, \gamma)\ \}.$$

# maintaining privacy



- secret initial knowledge
- encryption, $\{m\}_k, \{m\}_{pk(A)}$
- *homomorphic encyption,* $\{\!|m|\!\}_k$
- *blind signatures,* $[\![m]\!]_k$

# maintaining privacy

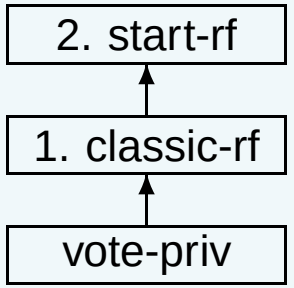


- secret initial knowledge
- encryption, $\{m\}_k, \{m\}_{pk(A)}$
- *homomorphic encyption,* $\{\!|m|\!\}_k$
- *blind signatures,* $[\![m]\!]_k$

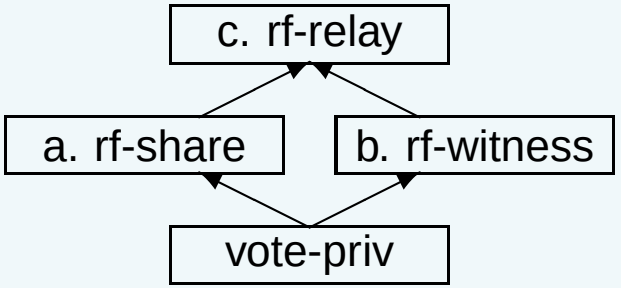
- privacy-enhancing communication
  a. public channel
  b. anonymous channel
  c. untappable channel
     - authority $\rightarrow$ voter
     - voter$\rightarrow$ authority
     - voter$\leftrightarrow$ authority

# conspiracy





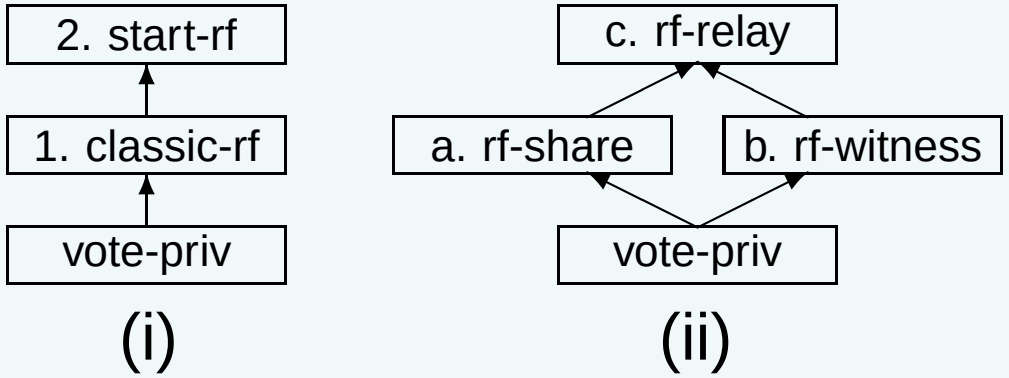

(i)




(ii)

# conspiracy

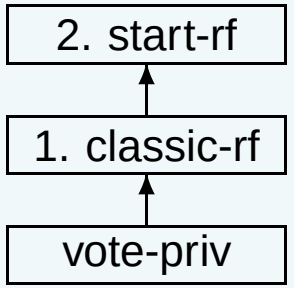




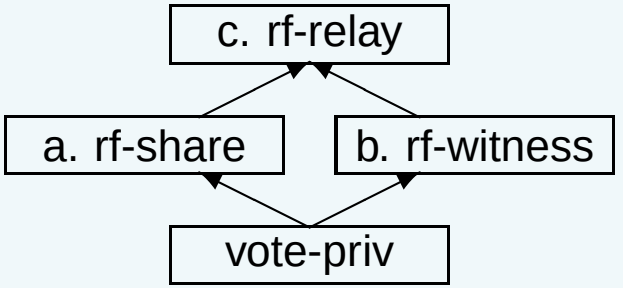

- transform processes using $\Theta_i$, where $i \in \{1, 2, a, b, c\}$.

# conspiracy






(i)




(ii)

- transform processes using $\Theta_i$, where $i \in \{1, 2, a, b, c\}$.
- transform events using $\theta_i$

# event transformation



- $\theta_a(v, ev) =$

$$\begin{cases} ur(ag, v, \varphi) \ . \ is(v, \varphi) & \text{if } ev = ur(ag, v, \varphi) \\ ev & \text{otherwise} \end{cases}$$

# event transformation



- $\theta_a(v, ev) =$

$$\begin{cases} ur(ag, v, \varphi) \ . \ is(v, \varphi) & \text{if } ev = ur(ag, v, \varphi) \\ ev & \text{otherwise} \end{cases}$$

- $\theta_b(v, ev) =$

$$\begin{cases} is(v, \mathrm{vars}(v, \varphi)) \ . \ ir(v, \mathrm{vars}(v, \varphi')) \ . \ us(v, ag, \varphi') \\ \qquad \text{if } ev = us(v, ag, \varphi), \text{ for } \varphi' = \mathrm{freshvars}(v, \varphi) \\ ev \qquad \text{otherwise} \end{cases}$$

# event transformation



- $\theta_a(v, ev) =$

$$\begin{cases} ur(ag, v, \varphi) \ . \ is(v, \varphi) & \text{if } ev = ur(ag, v, \varphi) \\ ev & \text{otherwise} \end{cases}$$

- $\theta_b(v, ev) =$

$$\begin{cases} is(v, \mathrm{vars}(v, \varphi)) \ . \ ir(v, \mathrm{vars}(v, \varphi')) \ . \ us(v, ag, \varphi') \\ \qquad \text{if } ev = us(v, ag, \varphi), \text{ for } \varphi' = \mathrm{freshvars}(v, \varphi) \\ ev \qquad \text{otherwise} \end{cases}$$

- $\theta_c(v, ev) = \theta_b(v, \theta_a(v, ev))$

# process transformation



$$\Theta_2(v, P) = is(knw_v).P$$

$$\Theta_i(v, P) = \begin{cases} \delta & \text{if} \quad i \neq 1 \wedge P = \delta \\ is(v, knw_v).\delta & \text{if} \quad i = 1 \wedge P = \delta \\ \\ \theta_i(v, ev).\Theta_i(v, P) & \text{if} \quad P = ev.P \\ \Theta_i(v, P_1) + \Theta_i(v, P_2) & \text{if} \quad P = P_1 + P_2 \\ \Theta_i(v, P_1) \lhd \varphi_1 = \varphi_2 \rhd \Theta_i(v, P_2) & \text{if} \quad P = P_1 \lhd \varphi_1 = \varphi_2 \rhd P_2, \\ & \quad \text{for } \varphi_1, \varphi_2 \in Terms \\ \\ \theta_i(v, ev).Y(\varphi_1, \ldots, \varphi_n), & \text{for fresh } Y(\mathsf{var}_1, \ldots, \mathsf{var}_n) = \Theta_i(v, P') \\ & \text{if } P = X(\varphi_1, \ldots, \varphi_n) \wedge X(\mathsf{var}_1, \ldots, \mathsf{var}_n) = P' \end{cases}$$

# conspiracy-resistance



classical notion:

$$\forall v, \gamma \colon \left| cg_v^1(\mathcal{VS}, \gamma) \right| > 1.$$

improved definition:

**Definition 6 (conspiracy-resistance)** *We call voting system $\mathcal{VS}$ conspiracy-resistant for conspiring behaviour $i \in \{1, 2, a, b, c\}$ iff*

$$\forall v \in \mathcal{V}, \gamma \in \mathcal{V} \to \mathcal{C} \colon cg_v^i(\mathcal{VS}, \gamma) = cg_v(\mathcal{VS}, \gamma).$$

# concluding



Conclusions:

- we can quantify privacy in voting
- enables detection of new attacks
- lifted notion of anonymity group to voting

Future work:

- consider conspiring authorities
- defense strategies
- automated application
- extend with probabilism (election result)

# final slide



Thank you for your attention.

Questions?