Quantifying voter-controlled privacy

Hugo Jonker

in collaboration with Sjouke Mauw and Jun Pang

h.jonker@surrey.ac.uk, hugo.jonker@uni.lu

Trustworth Voting project, University of Surrey

overview

- Express voting systems formally (syntax, semantics)
- Parametrise over voters' choice γ
- Determine trace set
- Compare trace set for γ_1 with trace set for γ_2

acquiring privacy

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

acquiring privacy

- secret initial knowledge
- encyption, $\{m\}_k, \{m\}_{pk(A)}$
- *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

setting

- 1 voter, 1 vote.
- every vote has equal weight.
- election process is phased.
- how voters vote is given (γ).

terms

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

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

terms

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

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

- syntactical equivalence: $\varphi_1 = \varphi_2$
- $\mathsf{vc} \in \mathsf{Vars}$ parametrises choice

- variable substitution: $\sigma = \mathsf{var} \mapsto \varphi_1$
  application to φ_2: $\sigma(\varphi_2)$

  No bound variables!

term matching

First idea:

$$\begin{aligned}
&\mathrm{match}(\varphi_{cl}, \varphi_o) \equiv \\
&\quad \varphi_o = \varphi_{cl} \ \vee \ \varphi_o \in \mathsf{Vars} \vee \\
&\quad \langle\ \exists \varphi'_{cl}, \varphi'_o, k\colon (\varphi_{cl} = \{\varphi'_{cl}\}_k \wedge \varphi_o = \{\varphi'_o\}_k) \ \wedge \ \mathrm{match}(\varphi'_{cl}, \varphi'_o)\ \rangle \\
&\quad \vee \langle \exists \varphi'_{cl}, \varphi''_{cl}, \varphi'_o, \varphi''_o\colon \varphi_{cl} = (\varphi'_{cl}, \varphi''_{cl}) \wedge \varphi_o = (\varphi'_o, \varphi''_o) \wedge \\
&\qquad \mathrm{match}(\varphi'_{cl}, \varphi'_o) \wedge \mathrm{match}(\varphi''_{cl}, \varphi''_o) \rangle
\end{aligned}$$

term matching

First idea:

$$\text{match}(\varphi_{cl}, \varphi_o) \equiv$$
$$\varphi_o = \varphi_{cl} \ \vee\ \varphi_o \in \mathsf{Vars} \vee$$
$$\langle\ \exists \varphi'_{cl}, \varphi'_o, k \colon (\varphi_{cl} = \{\varphi'_{cl}\}_k \wedge \varphi_o = \{\varphi'_o\}_k)\ \wedge\ \text{match}(\varphi'_{cl}, \varphi'_o)\ \rangle$$
$$\vee\ \langle \exists \varphi'_{cl}, \varphi''_{cl}, \varphi'_o, \varphi''_o \colon \varphi_{cl} = (\varphi'_{cl}, \varphi''_{cl}) \wedge \varphi_o = (\varphi'_o, \varphi''_o) \wedge$$
$$\text{match}(\varphi'_{cl}, \varphi'_o) \wedge \text{match}(\varphi''_{cl}, \varphi''_o)\rangle$$

However:

$$\text{match}(\quad (\{\varphi_1, k1\}_k, k), \quad (\{\varphi_1, k1\}_{\mathsf{var}}, \mathsf{var}) \quad)?$$

term matching

First idea:

$$\mathrm{match}(\varphi_{cl}, \varphi_o) \equiv$$
$$\varphi_o = \varphi_{cl} \ \vee \ \varphi_o \in \mathsf{Vars} \vee$$
$$\langle\, \exists \varphi'_{cl}, \varphi'_o, k\colon (\varphi_{cl} = \{\varphi'_{cl}\}_k \wedge \varphi_o = \{\varphi'_o\}_k) \ \wedge \ \mathrm{match}(\varphi'_{cl}, \varphi'_o) \,\rangle$$
$$\vee \ \langle \exists \varphi'_{cl}, \varphi''_{cl}, \varphi'_o, \varphi''_o\colon \varphi_{cl} = (\varphi'_{cl}, \varphi''_{cl}) \wedge \varphi_o = (\varphi'_o, \varphi''_o) \wedge$$
$$\mathrm{match}(\varphi'_{cl}, \varphi'_o) \wedge \mathrm{match}(\varphi''_{cl}, \varphi''_o)\rangle$$

However:

$$\mathrm{match}(\quad (\{\varphi_1, k1\}_k, k), \quad (\{\varphi_1, k1\}_{\mathsf{var}}, \mathsf{var}) \quad)?$$

Solution:

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

term derivation

$$K \cup \{\varphi\} \vdash \varphi$$

$$K \vdash \varphi_1,\ K \vdash \varphi_2 \implies K \vdash (\varphi_1, \varphi_2)$$

$$K \vdash (\varphi_1, \varphi_2) \implies K \vdash \varphi_1,\ K \vdash \varphi_2$$

$$K \vdash \varphi_1,\ K \vdash k \implies K \vdash \{\varphi_1\}_k$$

$$K \vdash \{\varphi_1\}_k,\ K \vdash k^{-1} \implies K \vdash \varphi_1$$

closure: $\overline{K} = \{\varphi \mid K \vdash \varphi\}$

events & processes

Phases, communication of terms:

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

Event order:

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

agents & voting system

State of agent is knowledge + process:

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

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

$$VotSys = Agents \rightarrow Agstate.$$

Instantiation of a voting system $\mathcal{VS}$ with choice function γ 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

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

receiving terms

Deconstruction of terms:

$$\varphi \sqsubseteq \varphi$$

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

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

receiving terms

Deconstruction of terms:

$$\begin{array}{lllllll} \varphi & \sqsubseteq & \varphi & & & & \\ \varphi_1 & \sqsubseteq & (\varphi_1, \varphi_2) & \quad & \varphi_2 & \sqsubseteq & (\varphi_1, \varphi_2) \\ \varphi & \sqsubseteq & \{\varphi\}_k & \quad & k^{-1} & \sqsubseteq & \{\varphi\}_k \end{array}$$

Readability of terms:

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

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:

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

- receive:

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

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:

$$\frac{\begin{array}{c} i \in \mathbb{N} \\ \mathit{Phase} \subseteq \{a\,@\,(knw_a, P_a) \in S \mid \exists P'_a : (K_I, knw_a, P_a) \xrightarrow{ph(i)} (K_I, knw_a, P'_a)\} \\ \mathit{Aut} \subseteq \{a \in \mathit{Agents} \mid \exists knw_a, P_a : a\,@\,(knw_a, P_a) \in \mathit{Phase}\} \\ \mathit{Phase}' = \{a\,@\,(knw_a, P'_a) \mid \exists P_a : a\,@\,(knw_a, P_a) \in \mathit{Phase} \wedge \\ \qquad (K_I, knw_a, P_a) \xrightarrow{ph(i)} (K_I, knw_a, P'_a)\} \end{array}}{(K_I, S) \xrightarrow{ph(i)} (K_I, \mathit{Phase}' \cup S \setminus \mathit{Phase})}$$

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

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

Traces of $\mathcal{VS}$:

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

modelling privacy

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

New idea:
When can the intruder distinguish $Tr(\mathcal{VS}^{\gamma_1})$ from $Tr(\mathcal{VS}^{\gamma_2})$?

When he can <u>reinterpret</u> t as t'.

reinterpretation

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

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

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

indistinguishability

Definition 3 (trace indistinguishability) *Traces t, t' are indistinguishable for the intruder, notation $t \sim t'$ iff there exists a reinterpretation ρ such that*

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

Definition 4 (choice indistinguishability) *Given voting system $\mathcal{VS}$, choice functions γ_1, γ_2 are indistinguishable to the intruder, notation $\gamma_1 \simeq_{\mathcal{VS}} \gamma_2$ iff*

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

measuring privacy

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

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

The choice group for a particular voter v, i.e. the set of candidates indistinguishable from v's chosen candidate, is given by

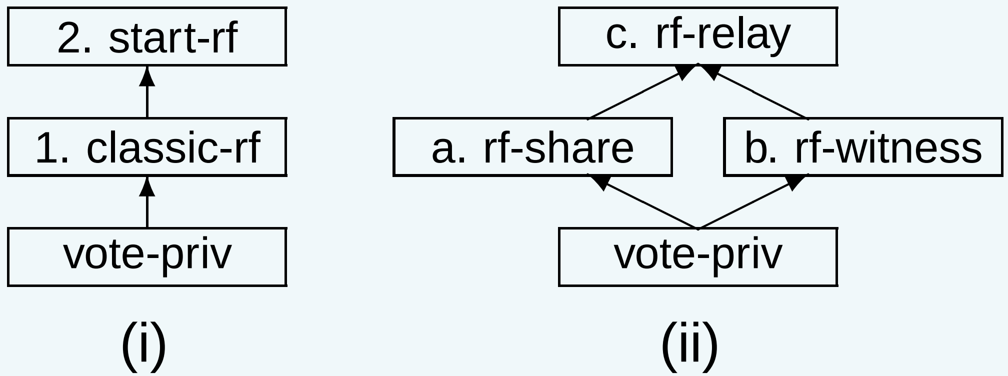
$$cg_v(\mathcal{VS}, \gamma) = \{\gamma'(v) \mid \gamma' \in cg(\mathcal{VS}, \gamma)\ \}.$$

recall

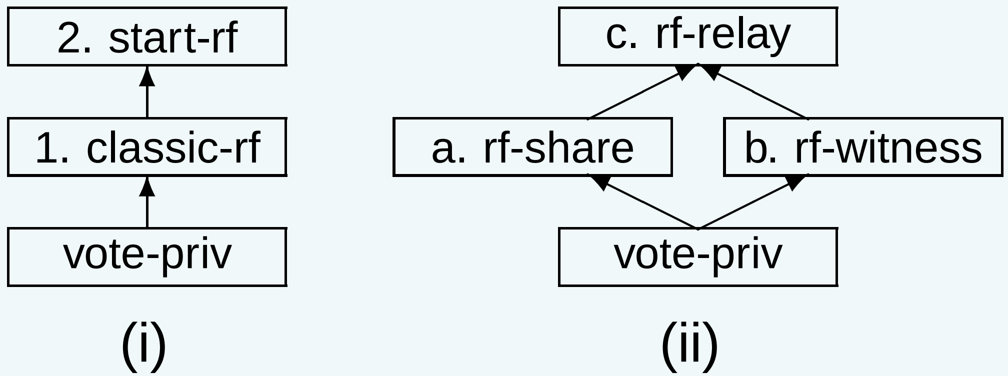
Privacy techniques:

- secret initial knowledge
- specific communication channels
  - untappable channel

conspiracy

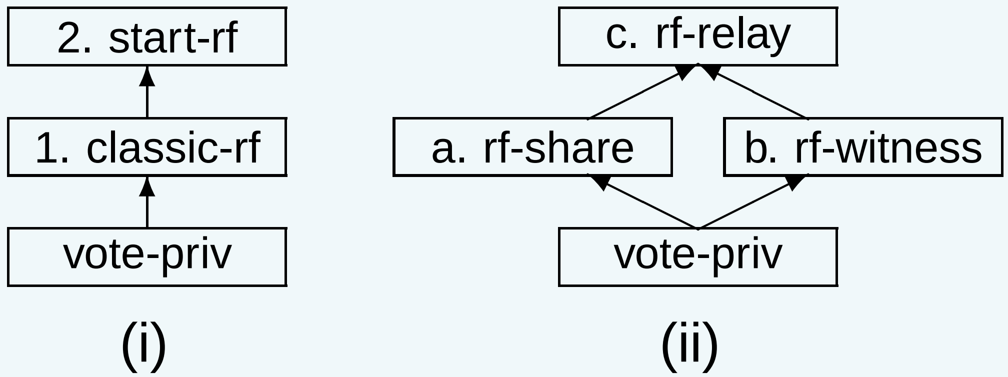

2. start-rf
1. classic-rf
vote-priv
(i)
c. rf-relay
a. rf-share
b. rf-witness
vote-priv
(ii)

conspiracy



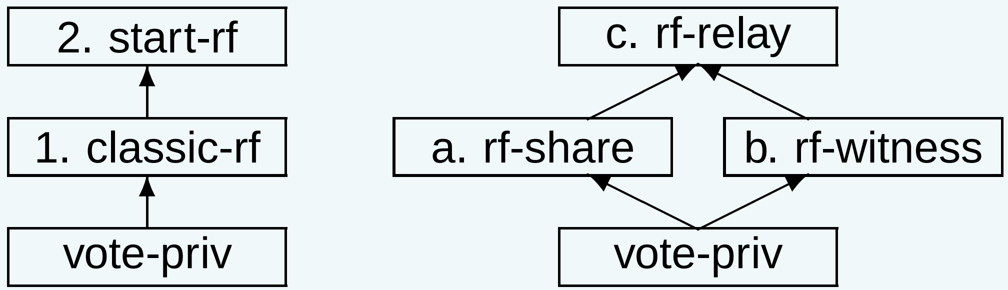
- transform processes using Θ_i, where $i \in \{1, 2, a, b, c\}$.

conspiracy



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

conspiracy



(i) (ii)

- transform processes using Θ_i, where $i \in \{1, 2, a, b, c\}$.
- transform events using θ_i
- coercion resistance i: $\forall v, \gamma\colon\ cg^i_v(\mathcal{VS}, \gamma) = cg_v(\mathcal{VS}, \gamma)$

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

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

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

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

$$\Theta_i(v, P) =$$

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

conspiracy-resistance

classical notion:

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

Our 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
- possibility to detect new attacks
- considering the exact choice group aids in reasoning about privacy

Future work:

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

final slide

Thank you for your attention.

Questions?