Quantifying voter-controlled privacy

Hugo Jonker, Sjouke Mauw and Jun Pang

hugo.jonker@uni.lu

SaToSS group, University of Luxembourg

FM group, Eindhoven University of Technology

Introduction
-privacy in voting
-acquiring privacy
-setting

Syntax

Semantics

Privacy

Attacking privacy

Application

wrapping up

privacy in voting

Privacy in voting is a must for democracy.

privacy in voting

Introduction
-privacy in voting
-acquiring privacy
-setting

Syntax

Semantics

Privacy

Attacking privacy

Application

wrapping up

Privacy in voting is a must for democracy.

Vote-privacy: link voter-candidate.

privacy in voting

Introduction
-privacy in voting
-acquiring privacy
-setting

Syntax

Semantics

Privacy

Attacking privacy

Application

wrapping up

Privacy in voting is a must for democracy.

Vote-privacy: link voter-candidate.

Voter-controlled privacy: control voter has over her privacy

privacy in voting

Introduction
-privacy in voting
-acquiring privacy
-setting

Syntax

Semantics

Privacy

Attacking privacy

Application

wrapping up

Privacy in voting is a must for democracy.

Vote-privacy: link voter-candidate.

Voter-controlled privacy: control voter has over her privacy

Goal:

Quantify voter-controlled privacy

acquiring privacy

Introduction
-privacy in voting
-acquiring privacy
-setting

Syntax

Semantics

Privacy

Attacking privacy

Application

wrapping up

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

- homomorphic encyption, $\{\!|m|\!\}_k$
- blind signatures, $[\![m]\!]_k$

acquiring privacy

Introduction
-privacy in voting
-acquiring privacy
-setting

Syntax

Semantics

Privacy

Attacking privacy

Application

wrapping up

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

- homomorphic encyption, $\{\!|m|\!\}_k$
- blind signatures, $[\![m]\!]_k$
  where $\{[\![m]\!]_k\}_{sk(A)} \vdash \{m\}_{sk(A)}$ and
  $\{\!|m1|\!\}_k \otimes \{\!|m2|\!\}_k = \{\!|m1 \oplus m2|\!\}_k$

acquiring privacy

Introduction
-privacy in voting
-acquiring privacy
-setting

Syntax

Semantics

Privacy

Attacking privacy

Application

wrapping up

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

- homomorphic encyption, $\{\!|m|\!\}_k$
- blind signatures, $[\![m]\!]_k$
  where $\{[\![m]\!]_k\}_{sk(A)} \vdash \{m\}_{sk(A)}$ and
  $\{\!|m1|\!\}_k \otimes \{\!|m2|\!\}_k = \{\!|m1 \oplus m2|\!\}_k$

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

setting

Introduction
-privacy in voting
-acquiring privacy
-setting
Syntax
Semantics
Privacy
Attacking privacy
Application
wrapping up

- 1 voter, 1 vote.
- every vote has equal weight.
- election process is phased.
- how voters vote is given (γ).
- cast votes made public in last phase ($\mathcal{RB}$).
- result is a function on $\mathcal{RB}$.

Introduction
Syntax
-terms
-term derivation
-specifying agents
-voting system
Semantics
Privacy
Attacking privacy
Application
wrapping up

terms

- voters $v \in \mathcal{V}$
- candidates $c \in \mathcal{C}$
- choice function $\gamma\colon \mathcal{V} \to \mathcal{C}$
- sets: variables $Vars$, keys $Keys$, nonces $Nonces$

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

terms

Introduction

Syntax
-terms
-term derivation
-specifying agents
-voting system

Semantics

Privacy

Attacking privacy

Application

wrapping up

- voters $v \in \mathcal{V}$
- candidates $c \in \mathcal{C}$
- choice function $\gamma \colon \mathcal{V} \to \mathcal{C}$
- sets: variables Vars, keys Keys, nonces Nonces

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

Matching: $\mathrm{match}(\varphi_a, \varphi_b) \equiv$

$$\begin{aligned}
&\varphi_a = \varphi_b \;\lor\; \varphi_b \in \mathit{Vars} \lor \\
&\langle\, \exists \varphi'_a, \varphi'_b, k\colon\; \mathrm{match}(\varphi'_a, \varphi'_b) \;\land ((\varphi_a = \{\varphi'_a\}_k \land \varphi_b = \{\varphi'_b\}_k) \lor \\
&\quad (\varphi_a = \{\!|\varphi'_a|\!\}_k \land \varphi_b = \{\!|\varphi'_b|\!\}_k) \lor (\varphi_a = [\![\varphi'_a]\!]_k \land \varphi_b = [\![\varphi'_b]\!]_k))\rangle \\
&\lor \langle \exists \varphi'_a, \varphi''_a, \varphi'_b, \varphi''_b \colon \varphi_a = (\varphi'_a, \varphi''_a) \land \varphi_b = (\varphi'_b, \varphi''_b) \land \\
&\quad \mathrm{match}(\varphi'_a, \varphi'_b) \land \mathrm{match}(\varphi''_a, \varphi''_b)\rangle
\end{aligned}$$

terms

Introduction
Syntax
-terms
-term derivation
-specifying agents
-voting system
Semantics
Privacy
Attacking privacy
Application
wrapping up

- voters $v \in \mathcal{V}$
- candidates $c \in \mathcal{C}$
- choice function $\gamma \colon \mathcal{V} \to \mathcal{C}$
- sets: variables $Vars$, keys $Keys$, nonces $Nonces$

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

Matching: $\mathrm{match}(\varphi_a, \varphi_b) \equiv$

$$\varphi_a = \varphi_b \;\vee\; \varphi_b \in Vars \vee$$
$$\langle\, \exists \varphi'_a, \varphi'_b, k\colon\; \mathrm{match}(\varphi'_a, \varphi'_b) \;\wedge ((\varphi_a = \{\varphi'_a\}_k \wedge \varphi_b = \{\varphi'_b\}_k) \vee$$
$$(\varphi_a = \{\!|\varphi'_a|\!\}_k \wedge \varphi_b = \{\!|\varphi'_b|\!\}_k) \vee (\varphi_a = [\![\varphi'_a]\!]_k \wedge \varphi_b = [\![\varphi'_b]\!]_k))\rangle$$
$$\vee \langle \exists \varphi'_a, \varphi''_a, \varphi'_b, \varphi''_b\colon \varphi_a = (\varphi'_a, \varphi''_a) \wedge \varphi_b = (\varphi'_b, \varphi''_b) \wedge$$
$$\mathrm{match}(\varphi'_a, \varphi'_b) \wedge \mathrm{match}(\varphi''_a, \varphi''_b)\rangle$$

variable instantiation by $\mathrm{vm}(\varphi_a, \varphi_b)$ (skipped)

term derivation

Introduction

Syntax
-terms
-term derivation
-specifying agents
-voting system

Semantics

Privacy

Attacking privacy

Application

wrapping up

$$T \cup \{\varphi\} \vdash \varphi$$

$$T \vdash \varphi_1,\ T \vdash \varphi_2 \quad \Longrightarrow \quad T \vdash (\varphi_1, \varphi_2)$$

$$T \vdash (\varphi_1, \varphi_2) \quad \Longrightarrow \quad T \vdash \varphi_1,\ T \vdash \varphi_2$$

$$T \vdash \varphi_1,\ T \vdash k \quad \Longrightarrow \quad T \vdash \{\varphi_1\}_k$$

$$T \vdash \{\varphi_1\}_k,\ T \vdash k^{-1} \quad \Longrightarrow \quad T \vdash \varphi_1$$

$$T \vdash \varphi_1, T \vdash k \quad \Longrightarrow \quad T \vdash \{\!|\varphi_1|\!\}_k$$

$$T \vdash \{\!|\varphi_1|\!\}_k, T \vdash k^{-1} \quad \Longrightarrow \quad T \vdash \varphi_1$$

$$T \vdash \varphi_1, T \vdash k \quad \Longrightarrow \quad T \vdash [\![\varphi_1]\!]_k$$

$$T \vdash \{[\![\varphi_1]\!]_k\}_{sk(a)}, T \vdash k \quad \Longrightarrow \quad T \vdash \{\varphi_1\}_{sk(a)}$$

$$T \vdash \{\!|\varphi_1|\!\}_k, T \vdash \{\!|\varphi_2|\!\}_k \quad \Longrightarrow \quad T \vdash \{\!|\varphi_1 \oplus \varphi_2|\!\}_k$$

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

specifying agents

Introduction
Syntax
-terms
-term derivation
-specifying agents
-voting system
Semantics
Privacy
Attacking privacy
Application
wrapping up

Agent behaviour = list of events.

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

Agent specification:

$$Spec = \mathcal{P}(Terms) \times (Vars \rightarrow Terms) \times Events^{\star}.$$

voting system

Introduction

Syntax
-terms
-term derivation
-specifying agents
-voting system

Semantics

Privacy

Attacking privacy

Application

wrapping up

Definition 1 (voting system) *The class of voting systems, $Prot$, is defined as $Prot = Agents \to Spec$. Instantiation of a voting system $\mathcal{VS} \in Prot$ with choice function γ is denoted as $\mathcal{VS}(\gamma)$. $\mathcal{VS}(\gamma)(a) =$*

$$\begin{cases} \mathcal{VS}(a) & \text{if } a \notin \mathcal{V} \\ (\pi_1(\mathcal{VS}(a)), \mu_a(\pi_2(\mathcal{VS}(a))), \pi_3(\mathcal{VS}(a))) & \text{if } a \in \mathcal{V} \end{cases}$$

where $\mu_a = vc \mapsto \gamma(a)$.

operational semantics

- send:

$$\frac{k_a \vdash \varphi' \wedge sp = a\colon (k_a, \mu, s(a,y,\varphi)\cdot\sigma) \in S \wedge \mu(\varphi) = \varphi'}{(K_I, S) \xrightarrow{s(a,y,\varphi')} (K_I \cup \{\varphi'\}, S \cup \{a\colon (k_a, \mu, \sigma)\} \setminus \{sp\}\ )}$$

operational semantics

- send:

$$\frac{k_a \vdash \varphi' \wedge sp = a\colon (k_a, \mu, s(a,y,\varphi)\cdot\sigma) \in S \wedge \mu(\varphi) = \varphi'}{(K_I, S) \xrightarrow{s(a,y,\varphi')} (K_I \cup \{\varphi'\}, S \cup \{a\colon (k_a, \mu, \sigma)\} \setminus \{sp\}\ )}$$

- receive:

$$\frac{K_I \vdash \varphi \wedge sp = a\colon (k_a, \mu, r(x,a,\varphi')\cdot\sigma) \in S \wedge \mu' = \mathrm{vm}(\varphi, \mu(\varphi')) \circ \mu \wedge \mathrm{match}(\varphi, \mu(\varphi'))}{(K_I, S) \xrightarrow{r(x,a,\varphi)} (K_I, S \cup \{a\colon (k_a \cup \{\varphi\}, \mu', \sigma)\} \setminus \{sp\}\ )}$$

operational semantics

- send:

$$\frac{k_a \vdash \varphi' \wedge sp = a\colon (k_a, \mu, s(a,y,\varphi)\cdot\sigma) \in S \wedge \mu(\varphi) = \varphi'}{(K_I, S) \xrightarrow{s(a,y,\varphi')} (K_I \cup \{\varphi'\}, S \cup \{a\colon (k_a, \mu, \sigma)\} \setminus \{sp\}\ )}$$

- receive:

$$\frac{K_I \vdash \varphi \wedge sp = a\colon (k_a, \mu, r(x,a,\varphi')\cdot\sigma) \in S \wedge \mu' = \mathrm{vm}(\varphi, \mu(\varphi')) \circ \mu \wedge \mathrm{match}(\varphi, \mu(\varphi'))}{(K_I, S) \xrightarrow{r(x,a,\varphi)} (K_I, S \cup \{a\colon (k_a \cup \{\varphi\}, \mu', \sigma)\} \setminus \{sp\}\ )}$$

- untappable communication:

$$\frac{\begin{array}{l} sp_a = a\colon (k_a, \mu_a, us(a,b,\varphi_a)\cdot\sigma_a) \in S\ \wedge k_a \vdash \varphi' \wedge \mu_a(\varphi_a) = \varphi' \wedge \\ sp_b = b\colon (k_b, \mu_b, ur(a,b,\varphi_b)\cdot\sigma_b) \in S \wedge \mu_b' = \mathrm{vm}(\varphi_a, \mu(\varphi_b)) \circ \mu_b \wedge \mathrm{match}(\varphi_a, \mu_b(\varphi_b)) \end{array}}{(K_I, S) \xrightarrow{\tau} (K_I, S \cup \{a\colon (k_a, \mu_a, \sigma), b\colon (k_b \cup \{\varphi'\}, \mu_b', \sigma_b)\} \setminus \{sp_a, sp_b\})}$$

operational semantics

- send:

$$\frac{k_a \vdash \varphi' \wedge sp = a\colon (k_a, \mu, s(a, y, \varphi) \cdot \sigma) \in S \wedge \mu(\varphi) = \varphi'}{(K_I, S) \xrightarrow{s(a,y,\varphi')} (K_I \cup \{\varphi'\}, S \cup \{a\colon (k_a, \mu, \sigma)\} \setminus \{sp\}\ )}$$

- receive:

$$\frac{K_I \vdash \varphi \wedge sp = a\colon (k_a, \mu, r(x, a, \varphi') \cdot \sigma) \in S \wedge \mu' = \mathrm{vm}(\varphi, \mu(\varphi')) \circ \mu \wedge \mathrm{match}(\varphi, \mu(\varphi'))}{(K_I, S) \xrightarrow{r(x,a,\varphi)} (K_I, S \cup \{a\colon (k_a \cup \{\varphi\}, \mu', \sigma)\} \setminus \{sp\}\ )}$$

- untappable communication:

$$\frac{\begin{array}{l} sp_a = a\colon (k_a, \mu_a, us(a, b, \varphi_a) \cdot \sigma_a) \in S\ \wedge k_a \vdash \varphi' \wedge \mu_a(\varphi_a) = \varphi' \wedge \\ sp_b = b\colon (k_b, \mu_b, ur(a, b, \varphi_b) \cdot \sigma_b) \in S \wedge \mu_b' = \mathrm{vm}(\varphi_a, \mu(\varphi_b)) \circ \mu_b \wedge \mathrm{match}(\varphi_a, \mu_b(\varphi_b)) \end{array}}{(K_I, S) \xrightarrow{\tau} (K_I, S \cup \{a\colon (k_a, \mu_a, \sigma), b\colon (k_b \cup \{\varphi'\}, \mu_b', \sigma_b)\} \setminus \{sp_a, sp_b\})}$$

- phase synchronisation:

$$\frac{Phase = \{a\colon (k_a, \mu_a, phase(i) \cdot \sigma_a) \in S\} \wedge \forall a \in Aut\colon a\colon (k_a, \mu_a, phase(i) \cdot \sigma_a) \in S}{(K_I, S) \xrightarrow{phase(i)} (K_I, S \cup \{a\colon (k_a, \mu_a, \sigma_a) \mid a\colon (k_a, \mu_a, phase(i) \cdot \sigma_a) \in S\} \setminus Phase)}$$

traces

Introduction

Syntax

Semantics
-operational semantics
-traces

Privacy

Attacking privacy

Application

wrapping up

Traces are lists of labels:

$$Labels = \{s(a, a', \varphi), r(a, a', \varphi), as(a', \varphi), ar(a', \varphi), \tau,$$
$$phase(i) \mid a, a' \in Agents, \varphi \in Terms, i \in \mathbb{N}\}.$$

Introduction
Syntax
Semantics
-operational semantics
-traces
Privacy
Attacking privacy
Application
wrapping up

Traces are lists of labels:

$$\begin{aligned} \mathit{Labels} = \{&s(a,a',\varphi), r(a,a',\varphi), as(a',\varphi), ar(a',\varphi), \tau, \\ &\mathit{phase}(i) \mid a, a' \in \mathit{Agents}, \varphi \in \mathit{Terms}, i \in \mathbb{N}\}. \end{aligned}$$

Traces of a voting system choice function γ:

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

traces

Introduction
Syntax
Semantics
-operational semantics
-traces
Privacy
Attacking privacy
Application
wrapping up

Traces are lists of labels:

$$Labels = \{s(a, a', \varphi), r(a, a', \varphi), as(a', \varphi), ar(a', \varphi), \tau, \\ phase(i) \mid a, a' \in Agents, \varphi \in Terms, i \in \mathbb{N}\}.$$

Traces of a voting system choice function γ:

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

Traces of $\mathcal{VS}$:

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

modelling privacy

Introduction

Syntax

Semantics

Privacy
-modelling privacy
-reinterpretation
-indistinguishability
-measuring privacy

Attacking privacy

Application

wrapping up

Standard approach:
Intruder observes a trace t, with which traces t' is this compatible?

Original idea:
When can the intruder distinguish $t \in Tr(\mathcal{VS}(\gamma))$ from $t' \in Tr(\mathcal{VS}(\gamma'))$?

modelling privacy

Introduction
Syntax
Semantics
Privacy
-modelling privacy
-reinterpretation
-indistinguishability
-measuring privacy
Attacking privacy
Application
wrapping up

Standard approach:
Intruder observes a trace t, with which traces t' is this compatible?

Original idea:
When can the intruder distinguish $t \in Tr(\mathcal{VS}(\gamma))$ from $t' \in Tr(\mathcal{VS}(\gamma'))$?

New idea:
When can the intruder distinguish $Tr(\mathcal{VS}(\gamma))$ from $Tr(\mathcal{VS}(\gamma'))$?

reinterpretation

Introduction
Syntax
Semantics
Privacy
-modelling privacy
-reinterpretation
-indistinguishability
-measuring privacy
Attacking privacy
Application
wrapping up

Definition 2 (reinterpretation (GHPvR05)) *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.*

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

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

Introduction
Syntax
Semantics
Privacy
-modelling privacy
-reinterpretation
-indistinguishability
-measuring privacy
Attacking privacy
Application
wrapping up

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, γ' are indistinguishable to the intruder, notation $\gamma \simeq_{\mathcal{VS}} \gamma'$ iff*

$$\forall t \in Tr(\mathcal{VS}(\gamma))\colon \exists t' \in Tr(\mathcal{VS}(\gamma'))\colon t \sim t' \quad \wedge$$
$$\forall t \in Tr(\mathcal{VS}(\gamma'))\colon \exists t' \in Tr(\mathcal{VS}(\gamma))\colon t \sim t'$$

Introduction
Syntax
Semantics
Privacy
-modelling privacy
-reinterpretation
-indistinguishability
-measuring privacy
Attacking privacy
Application
wrapping up

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

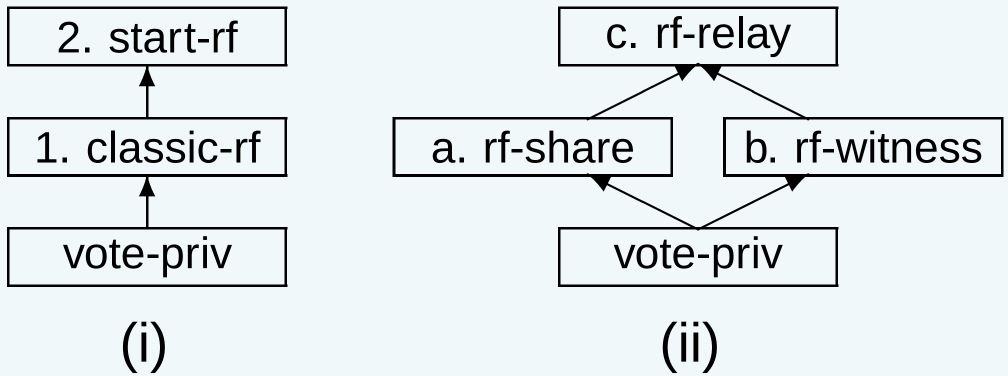
$$cg_v(\mathcal{VS}, \gamma) = \{\gamma'(v) \mid \gamma' \in cg(\mathcal{VS}, \gamma)\}.$$

Introduction
Syntax
Semantics
Privacy
Attacking privacy
-recall
-conspiracy
-modelling conspiracy
-receipt-freeness
Application
wrapping up

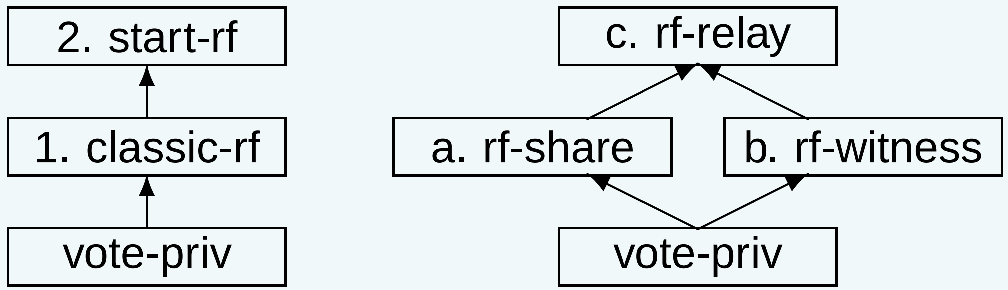
Privacy techniques:

- secret initial knowledge
- encyption, $\{m\}_k, \{m\}_{pk(A)}$
- signatures, $\{m\}_{sk(A)}$
- homomorphic encyption, $\{\!|m|\!\}_k$
- blind signatures, $[\![m]\!]_k$
- alternate communication channels
  a. public channel
  b. anonymous channel
  c. untappable channel

Introduction
Syntax
Semantics
Privacy
Attacking privacy
-recall
-conspiracy
-modelling conspiracy
-receipt-freeness
Application
wrapping up



conspiracy



(i) (ii)

$$\mathrm{vars}(v,\varphi) = \begin{cases} \{\varphi\} & \text{if } \varphi \in \mathit{Vars} \\ \mathrm{vars}(v,\varphi_a) \cup \mathit{vars}(v,\varphi_b) & \text{if } \varphi = (\varphi_a,\varphi_b) \\ \mathrm{vars}(v,\varphi') & \\ \qquad \text{if } (\varphi = \{\varphi'\}_k \vee \varphi = \{\!|\varphi'|\!\}_k \vee \varphi = [\![\varphi']\!]_k), & \\ & k \in \mathit{Keys}_v \\ \emptyset & \text{otherwise} \end{cases}$$

modelling conspiracy

Introduction
Syntax
Semantics
Privacy
Attacking privacy
-recall
-conspiracy
-modelling conspiracy
-receipt-freeness
Application
wrapping up

- $\delta_1(v, sp) = sp \cdot is(k_v)$

modelling conspiracy

Introduction
Syntax
Semantics
Privacy
Attacking privacy
-recall
-conspiracy
-modelling conspiracy
-receipt-freeness
Application
wrapping up

- $\delta_1(v, sp) = sp \cdot is(k_v)$
- $\delta_2(v, sp) = is(k_v) \cdot sp$

modelling conspiracy

Introduction
Syntax
Semantics
Privacy
Attacking privacy
-recall
-conspiracy
-modelling conspiracy
-receipt-freeness
Application
wrapping up

- $\delta_1(v, sp) = sp \cdot is(k_v)$
- $\delta_2(v, sp) = is(k_v) \cdot sp$
- $\delta_a(v, ev \cdot sp) =$
$$\begin{cases} ur(ag, v, \varphi) \cdot is(\varphi) \cdot \delta_a(v, sp) & \text{if } ev = ur(ag, v, \varphi) \\ ev \cdot \delta_a(v, sp) & \text{otherwise} \end{cases}$$

modelling conspiracy

Introduction
Syntax
Semantics
Privacy
Attacking privacy
-recall
-conspiracy
-modelling conspiracy
-receipt-freeness
Application
wrapping up

- $\delta_1(v, sp) = sp \cdot is(k_v)$
- $\delta_2(v, sp) = is(k_v) \cdot sp$
- $\delta_a(v, ev \cdot sp) =$
$$\begin{cases} ur(ag, v, \varphi) \cdot is(\varphi) \cdot \delta_a(v, sp) & \text{if } ev = ur(ag, v, \varphi) \\ ev \cdot \delta_a(v, sp) & \text{otherwise} \end{cases}$$
- $\delta_b(v, ev \cdot sp) =$
$$\begin{cases} is(\mathrm{vars}(v, \varphi)) \cdot ir(\varphi') \cdot us(v, ag, \varphi'') \cdot \delta_b(v, sp) & \\ \qquad \text{if } ev = us(v, ag, \varphi) & \\ ev \cdot \delta_b(v, sp) & \text{otherwise} \end{cases}$$

modelling conspiracy

Introduction
Syntax
Semantics
Privacy
Attacking privacy
-recall
-conspiracy
-modelling conspiracy
-receipt-freeness
Application
wrapping up

- $\delta_1(v, sp) = sp \cdot is(k_v)$
- $\delta_2(v, sp) = is(k_v) \cdot sp$
- $\delta_a(v, ev \cdot sp) =$
$$\begin{cases} ur(ag, v, \varphi) \cdot is(\varphi) \cdot \delta_a(v, sp) & \text{if } ev = ur(ag, v, \varphi) \\ ev \cdot \delta_a(v, sp) & \text{otherwise} \end{cases}$$
- $\delta_b(v, ev \cdot sp) =$
$$\begin{cases} is(\text{vars}(v, \varphi)) \cdot ir(\varphi') \cdot us(v, ag, \varphi'') \cdot \delta_b(v, sp) & \\ & \text{if } ev = us(v, ag, \varphi) \\ ev \cdot \delta_b(v, sp) & \text{otherwise} \end{cases}$$
- $\delta_c(v, sp) = \delta_b(v, \delta_a(v, sp))$.

receipt-freeness

Introduction
Syntax
Semantics
Privacy
Attacking privacy
-recall
-conspiracy
-modelling conspiracy
-receipt-freeness
Application
wrapping up

classical notion:

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

Our definition:

Definition 6 (receipt-freeness) *Voting system $\mathcal{VS}$ is receipt-free for conspiring behaviour i iff*

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

Dutch elections

Introduction
Syntax
Semantics
Privacy
Attacking privacy
Application
-Dutch elections
-ThreeBallot
wrapping up

CDA		Wilders		...	SP	
CDA-1	☐	Wilders-1	☐	...	SP-1	☐
⋮		⋮		⋮	⋮	
CDA-x	☐	Wilders-y	☐	...	SP-z	☐

vote for candidate = implicit vote for party.

Problems:

- reducing privacy to party may suffice
- elimination of one particular party may suffice.

ThreeBallot

ballot 1a		ballot 1b		ballot 1c	
can 1	☐	can 1	☐	can 1	☐
⋮	⋮	⋮	⋮	⋮	⋮
can N	☐	can N	☐	can N	☐
identifier 1a		identifier 1b		identifier 1c	

ThreeBallot

Introduction
Syntax
Semantics
Privacy
Attacking privacy
Application
-Dutch elections
-ThreeBallot
wrapping up

ballot 1a	ballot 1b	ballot 1c
can 1 ☐	can 1 ☐	can 1 ☐
⋮ ⋮	⋮ ⋮	⋮ ⋮
can N ☐	can N ☐	can N ☐
identifier 1a	identifier 1b	identifier 1c

- not all sub-ballots form correct ballots: $cg_v^1(\mathit{3BS}, \gamma) \subseteq cg_v(\mathit{3BS}, \gamma)$.

ThreeBallot

Introduction

Syntax

Semantics

Privacy

Attacking privacy

Application

-Dutch elections

-ThreeBallot

wrapping up

ballot 1a	ballot 1b	ballot 1c
can 1 ☐	can 1 ☐	can 1 ☐
⋮ ⋮	⋮ ⋮	⋮ ⋮
can N ☐	can N ☐	can N ☐
identifier 1a	identifier 1b	identifier 1c

- not all sub-ballots form correct ballots:
  $cg^1_v(\mathit{3BS}, \gamma) \subseteq cg_v(\mathit{3BS}, \gamma)$.
- signature attack:
  $cg^b_v(\mathit{3BS}, \gamma) \subseteq cg^1_v(\mathit{3BS}, \gamma)$.

concluding

Introduction
Syntax
Semantics
Privacy
Attacking privacy
Application
wrapping up
-concluding

Future work:

- extend syntax (deterministic and non-deterministic choice)
- extend definitions (coercion resistance)
- consider spec transformations of authorities
- detailed application
- extend with probabilism (election result)

Introduction
Syntax
Semantics
Privacy
Attacking privacy
Application
wrapping up
-concluding

Thank you for your attention.

Questions?