Privacy in eVoting

(joint work with Erik de Vink and Sjouke Mauw)

Hugo Jonker

h.l.jonker@tue.nl

overview

- voting in the “real world”
  - privacy in voting
- voting electronically (digitally / over the internet)
  - (aside) irregularities
  - privacy in evoting
- formalising privacy
  - characterising receipts
  - receipt-freeness as anonymity
  - current / future work

typical elections

- set of candidates
- set of voters
- one vote for one candidate per voter
- result is multiset of cast votes

E.g. national elections in the Netherlands.

preventing cheating

Cheating in elections is prevented by law, procedures and regulations, e.g.:

> At all times during the elections, the chairman and two members of the voting bureau are present
> *Kieswet, Artikel J lid 12 sub 1*

This provides (some) protection against incorrect voting, multiple voting, incorrect counting, etc. etc.

privacy

per-district:

- record kept of who votes
- paper ballots: mixed, so somewhat ok (note: UK elections)
- voting machines: unclear

district size: average of $\pm 1,400$ voters

pro’s & con’s

advantages:

disadvantages:

pro's & con's

advantages:

- more voter convenience ($\overset{?}{\Longrightarrow}$ greater turnout)
- less overhead
- quicker counting
- large scale updates are easy

disadvantages:

pro’s & con’s

advantages:

- more voter convenience ($\stackrel{?}{\Longrightarrow}$ greater turnout)
- less overhead
- quicker counting
- large scale updates are easy

disadvantages:

- costlier
- re-invent the wheel:
  - danger of introducing new flaws
  - risk of forgetting about known flaws
- large scale updates are easy

irregularities

As an aside, some insights / anecdotes on:

- Sdu voting machine reveals votes through radiation
- Nedap voting machines not secure
- elections irregularities in Eindhoven

properties

established voting properties include:

- democracy
- eligibility
- accuracy
- verifiability
  - individual
  - universal
- fairness

privacy

- Anonymity
  vote is private w.r.t. an observer
- receipt-freeness
  no proof
- strong receipt-freeness
  no elimination of possibilities
- coercion-resistance
  - no randomisation
  - no abstention
  - no simulation

intuition

A receipt proves how a voter voted.

intuition

A receipt proves how a voter voted.

Examples:

- Everyone signs their vote.

intuition

A receipt proves how a voter voted.

Examples:

- Everyone signs their vote.
- In Italy, simultaneous elections were held for various posts, using one ballot. The order of posts listed is up to the voter, and is preserved. An attacker (El Mafiosi) can assign each voter a specific order of posts.
  Benaloh & Tuinstra

requirements

More precisely: a receipt r proves that a voter v *cast* a vote for candidate c.

requirements

More precisely: a receipt r proves that a voter v *cast* a vote for candidate c.

- R1: r authenticates v

requirements

More precisely: a receipt r proves that a voter v *cast* a vote for candidate c.

- R1: r authenticates v
- R2: r proves that v chose candidate c

requirements

More precisely: a receipt r proves that a voter v *cast* a vote for candidate c.

- R1: r authenticates v
- R2: r proves that v chose candidate c
- R3: r proves that v cast her vote

requirements

More precisely: a receipt r proves that a voter v *cast* a vote for candidate c.

- R1: r authenticates v
- R2: r proves that v chose candidate c
- R3: r proves that v cast her vote

Note:
- Specific for this type of elections
- Quite strict

ingredients

- voters $v \in \mathcal{V}$, choices $c \in \mathcal{C}$
- ballots $\mathcal{B}$ and results (multisets of choices) $\mathcal{M}(\mathcal{C})$
- a set of received ballots $\mathcal{RB}$, from which the result will be computed
- a choice function $\Gamma \colon \mathcal{V} \to \mathcal{C}$, which specifies how the voters vote

ingredients

- voters $v \in \mathcal{V}$, choices $c \in \mathcal{C}$
- ballots $\mathcal{B}$ and results (multisets of choices) $\mathcal{M}(\mathcal{C})$
- a set of received ballots $\mathcal{RB}$, from which the result will be computed
- a choice function $\Gamma \colon \mathcal{V} \to \mathcal{C}$, which specifies how the voters vote

- the set of receipts $\mathcal{R}$
- $\mathit{Terms}(v)$, the set of all terms that a voter $v \in \mathcal{V}$ can generate
- authentication terms $\mathcal{AT}(v)$:
  $t \in \mathcal{AT}(v) \implies \forall w \neq v \colon t \notin \mathit{Terms}(w)$
- $\mathit{auth} \colon \mathcal{AT} \to \mathcal{V}$, the unique voter that created an AT

decomposing receipts

The following functions are used to decompose receipts:

- $\alpha : \mathcal{R} \rightarrow \mathcal{AT}$, extract authentication term from receipt
- $\beta : \mathcal{R} \rightarrow \mathcal{RB}$, extract ballot from receipt
- $\gamma : \mathcal{R} \rightarrow \mathcal{C}$, extract candidate from receipt

Formalisation of the requirements:

decomposing receipts

The following functions are used to decompose receipts:

- $\alpha\colon \mathcal{R} \rightarrow \mathcal{AT}$, extract authentication term from receipt
- $\beta\colon \mathcal{R} \rightarrow \mathcal{RB}$, extract ballot from receipt
- $\gamma\colon \mathcal{R} \rightarrow \mathcal{C}$, extract candidate from receipt

Formalisation of the requirements:

- R1: $\alpha(r) \in \mathcal{AT}(v)$

decomposing receipts

The following functions are used to decompose receipts:

- $\alpha : \mathcal{R} \to \mathcal{AT}$, extract authentication term from receipt
- $\beta : \mathcal{R} \to \mathcal{RB}$, extract ballot from receipt
- $\gamma : \mathcal{R} \to \mathcal{C}$, extract candidate from receipt

Formalisation of the requirements:

- R1: $\alpha(r) \in \mathcal{AT}(v)$
- R2: $\gamma(r) = \Gamma(v)$

decomposing receipts

The following functions are used to decompose receipts:

- $\alpha : \mathcal{R} \rightarrow \mathcal{AT}$, extract authentication term from receipt
- $\beta : \mathcal{R} \rightarrow \mathcal{RB}$, extract ballot from receipt
- $\gamma : \mathcal{R} \rightarrow \mathcal{C}$, extract candidate from receipt

Formalisation of the requirements:

- R1: $\alpha(r) \in \mathcal{AT}(v)$
- R2: $\gamma(r) = \Gamma(v)$
- R3: $\beta(r) \in \mathcal{RB}$

decomposing receipts

The following functions are used to decompose receipts:

- $\alpha : \mathcal{R} \to \mathcal{AT}$, extract authentication term from receipt
- $\beta : \mathcal{R} \to \mathcal{RB}$, extract ballot from receipt
- $\gamma : \mathcal{R} \to \mathcal{C}$, extract candidate from receipt

Formalisation of the requirements:

- R1: $\alpha(r) \in \mathcal{AT}(v)$
- R2: $\gamma(r) = \Gamma(v)$
- R3: $\beta(r) \in \mathcal{RB}$

So, for valid receipts: $auth(\alpha(r)) = v \implies \gamma(r) = \Gamma(v)$, which is satisfied by $\gamma = \Gamma \circ auth \circ \alpha$.

receipts as terms

Intuitively, a receipt must be derivable from an actual execution of a voting protocol (i.e. receipts generated outside a protocol do not invalidate that protocol).

To facilitate detection of receipts, limit the notion of receipts to terms (i.e. $\mathcal{R} = \emptyset \vee \mathcal{R} \subseteq Terms$).

Now:

- Model the protocol in ACP
- Test suitability of communicated terms as receipts
- Pronounce judgment

receipts as terms

Intuitively, a receipt must be derivable from an actual execution of a voting protocol (i.e. receipts generated outside a protocol do not invalidate that protocol).

To facilitate detection of receipts, limit the notion of receipts to terms (i.e. $\mathcal{R} = \emptyset \vee \mathcal{R} \subseteq Terms$).

Now:

- Model the protocol in ACP (+ tweaks)
- Test suitability of communicated terms as receipts
- Pronounce judgment

suitable terms

Write $t \in t'$ if t is a subterm of t'.

α, β *extract* terms from terms, i.e. they deal with subterms.

suitable terms

Write $t \in t'$ if t is a subterm of t'.

α, β *extract* terms from terms, i.e. they deal with subterms.

Lemma $\forall t \in \mathcal{R} \colon \alpha(t) \in t \;\wedge\; \beta(t) \in t$

suitable terms

Write $t \in t'$ if t is a subterm of t'.

α, β *extract* terms from terms, i.e. they deal with subterms.

Lemma $\forall t \in \mathcal{R} \colon \alpha(t) \in t \;\wedge\; \beta(t) \in t$

(Note that, by definition: $t \in t' \wedge t \in \mathcal{AT}(v) \implies t' \in \mathcal{AT}(v)$.
So receipts are themselves authentication terms)

suitable terms

Write $t \in t'$ if t is a subterm of t'.

α, β *extract* terms from terms, i.e. they deal with subterms.

Lemma $\forall t \in \mathcal{R} \colon \alpha(t) \in t \ \wedge \ \beta(t) \in t$

(Note that, by definition: $t \in t' \wedge t \in \mathcal{AT}(v) \implies t' \in \mathcal{AT}(v)$.
So receipts are themselves authentication terms)

Although this does not capture the entire notion of receipts, it turns out to be strong enough in the examined cases.

RF ≈ anonymity

Anonymity, 3 flavours:

- sender/voter anonymity?
  no, voter tries to prove vote

RF $\approx$ anonymity

Anonymity, 3 flavours:

- sender/voter anonymity?
  no, voter tries to prove vote
- plausible deniability?
  no, sender knows how she voted

RF $\approx$ anonymity

Anonymity, 3 flavours:

- sender/voter anonymity?
  no, voter tries to prove vote
- plausible deniability?
  no, sender knows how she voted
- unlinkability?
  "no link between vote and voter"...

unlinkability

Unlinkability of message m to sender v:

- intruder does not know that v sent m
- intruder cannot rule out that v sent any message m', where $m' \in AS$, the Anonymity Set

unlinkability

Unlinkability of message m to sender v:

- intruder does not know that v sent m
- intruder cannot rule out that v sent any message m', where $m' \in AS$, the Anonymity Set

... “cannot rule out” ...

unlinkability

Unlinkability of message m to sender v:

- intruder does not know that v sent m
- intruder cannot rule out that v sent any message m', where $m' \in AS$, the Anonymity Set

... "cannot rule out" ...

strong rf the intruder cannot rule out *any* vote from the anonymity set.

different approaches

Current situation:

- Delaune, Kremer and Ryan proposed an approach based on bisimilarity
  – ignoring the notion of receipts
- Jonker and De Vink proposed an approach based on the characteristics of a receipt
  – founded on the notion of receipts

different approaches

Current situation:

- Delaune, Kremer and Ryan proposed an approach based on bisimilarity
  – ignoring the notion of receipts
- Jonker and De Vink proposed an approach based on the characteristics of a receipt
  – founded on the notion of receipts

Almost reminiscant of Heisenberg vs. Schrödinger ;-)

unifying approach

- branching bisimilarity as an equivalence seems to strong
  e.g. order in which voters vote does not affect rf
- checking terms J&DV-style seems imprecise
  not a precise notion of receipts
- so unite the two!
  construct an appropriate equivalence notion for voting processes based on identifying receipts

todo

- Combine J&DV and DKR
- How do the various privacy notions relate to eachother?

todo

- Combine J&DV and DKR
- How do the various privacy notions relate to eachother?

Further reading:

- "Formalising Receipt-Freeness", H.L. Jonker and E.P. de Vink. In Information Security Conference 2006, LNCS 4176
- "Receipt-Freeness as a special case of Anonymity in Epistemic Logic", Hugo Jonker and Wolter Pieters, WOTE 2006

todo

- Combine J&DV and DKR
- How do the various privacy notions relate to eachother?

Further reading:

- "Formalising Receipt-Freeness", H.L. Jonker and E.P. de Vink. In Information Security Conference 2006, LNCS 4176
- "Receipt-Freeness as a special case of Anonymity in Epistemic Logic", Hugo Jonker and Wolter Pieters, WOTE 2006

Thanks for your attention

example: BT

- Original receipt-freeness paper by Benaloh & Tuinstra
- Attack found... but not on the main scheme
- Assumes untappable channels and a voting booth
- Uses randomised encryption and “ZKP”

Process for voting authority:

Process for a voter:

example: BT

- Original receipt-freeness paper by Benaloh & Tuinstra
- Attack found... but not on the main scheme
- Assumes untappable channels and a voting booth
- Uses randomised encryption and “ZKP”

Process for voting authority:

$$\begin{aligned} A(v) \quad = \quad & \textstyle\sum_{x \in E(0),\ y \in E(1)} s_{a \to v}(min(x,y), max(x,y)) \cdot \\ & p^*_{a \to v}(x \in E(0) \wedge y \in E(1)) \cdot \big(\ r_{v \to a}(x) + r_{v \to a}(y)\ \big) \end{aligned}$$

Process for a voter:

example: BT

- Original receipt-freeness paper by Benaloh & Tuinstra
- Attack found... but not on the main scheme
- Assumes untappable channels and a voting booth
- Uses randomised encryption and “ZKP”

Process for voting authority:

$$\begin{aligned} A(v) \quad = \quad & \textstyle\sum_{x \in E(0),\ y \in E(1)} s_{a \to v}(min(x,y), max(x,y)) \cdot \\ & p^*_{a \to v}(x \in E(0) \wedge y \in E(1)) \cdot \big(\ r_{v \to a}(x) + r_{v \to a}(y)\ \big) \end{aligned}$$

Process for a voter:

$$\begin{aligned} V \quad = \quad & \textstyle\sum_{x,y} r_{a \to v}(x,y) \cdot \sum_{i \in \{0,1\}} p^*_{a \to v}(x \in E(i) \wedge y \in E(1-i)) \cdot \\ & \big(\Gamma(v) = i \to s_{v \to a}(x) \quad + \quad \Gamma(v) = 1 - i \to s_{v \to a}(y)\ \big) \end{aligned}$$

BT: receipt-free

Let’s examine the voter process:

BT: receipt-free

Let's examine the voter process:

$$V = \sum_{x,y} r_{a \to v}(x, y) \cdot$$

Not an authentication term

BT: receipt-free

Let's examine the voter process:

$$V = \sum_{x,y} r_{a \to v}(x, y) \cdot$$

Not an authentication term

$$\sum_{i \in \{0,1\}} p^*_{a \to v}(x \in E(i) \wedge y \in E(1-i)) \cdot$$

No ballot as a subterm

BT: receipt-free

Let’s examine the voter process:

$$V = \sum_{x,y} r_{a \to v}(x, y) \cdot$$

Not an authentication term

$$\sum_{i \in \{0,1\}} p^{*}_{a \to v}(x \in E(i) \land y \in E(1 - i)) \cdot$$

No ballot as a subterm

$$\big(\Gamma(v) = i \to s_{v \to a}(x) \quad + \quad \Gamma(v) = 1 - i \to s_{v \to a}(y)\ \big)$$

Subterm of first term!

BT: receipt-free

Let's examine the voter process:

$$V = \sum_{x,y} r_{a \to v}(x, y) \cdot$$

Not an authentication term

$$\sum_{i \in \{0,1\}} p^*_{a \to v}(x \in E(i) \wedge y \in E(1-i)) \cdot$$

No ballot as a subterm

$$\big(\Gamma(v) = i \to s_{v \to a}(x) \quad + \quad \Gamma(v) = 1 - i \to s_{v \to a}(y)\ \big)$$

Subterm of first term!

None of the terms from the voter can satisfy $\alpha(t) \in t \wedge \beta(t) \in t$
$\implies$ BT is receipt-free!

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote
2. $v \rightarrow a$: blinded, encrypted vote signed by v

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote
2. $v \longrightarrow a$: blinded, encrypted vote signed by v
3. $a \longrightarrow v$: blinded, encrypted vote signed by a

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote
2. $v \longrightarrow a$: blinded, encrypted vote signed by v
3. $a \longrightarrow v$: blinded, encrypted vote signed by a
4. $v \longrightarrow cnt$: encrypted vote signed by a

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote
2. $v \longrightarrow a$: blinded, encrypted vote signed by v
3. $a \longrightarrow v$: blinded, encrypted vote signed by a
4. $v \longrightarrow cnt$: encrypted vote signed by a
5. cnt: collect all votes

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote
2. $v \longrightarrow a$: blinded, encrypted vote signed by v
3. $a \longrightarrow v$: blinded, encrypted vote signed by a
4. $v \longrightarrow cnt$: encrypted vote signed by a
5. cnt: collect all votes
6. cnt: publish list of received votes

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote
2. $v \longrightarrow a$: blinded, encrypted vote signed by v
3. $a \longrightarrow v$: blinded, encrypted vote signed by a
4. $v \longrightarrow cnt$: encrypted vote signed by a
5. cnt: collect all votes
6. cnt: publish list of received votes
7. $v \longrightarrow cnt$: decryption key, index of vote in list

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote
2. $v \rightarrow a$: blinded, encrypted vote signed by v
3. $a \rightarrow v$: blinded, encrypted vote signed by a
4. $v \rightarrow cnt$: encrypted vote signed by a
5. cnt: collect all votes
6. cnt: publish list of received votes
7. $v \rightarrow cnt$: decryption key, index of vote in list
8. cnt: publish list of received keys

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote
2. $v \longrightarrow a$: blinded, encrypted vote signed by v
3. $a \longrightarrow v$: blinded, encrypted vote signed by a
4. $v \longrightarrow cnt$: encrypted vote signed by a
5. cnt: collect all votes
6. cnt: publish list of received votes
7. $v \longrightarrow cnt$: decryption key, index of vote in list
8. cnt: publish list of received keys

Obvious receipt... but it seems to lose its validity

example: FOO

Rough sketch of the FOO protocol for voter v, admin a and counter cnt:

1. v: create a blinded, encrypted vote
2. $v \rightarrow a$: blinded, encrypted vote signed by v
3. $a \rightarrow v$: blinded, encrypted vote signed by a
4. $v \rightarrow cnt$: encrypted vote signed by a
5. cnt: collect all votes
6. cnt: publish list of received votes
7. $v \rightarrow cnt$: decryption key, index of vote in list
8. cnt: publish list of received keys

Obvious receipt... but it seems to lose its validity
Timestamping $\implies$ no it doesn't!

example: RIES

- Used in Dutch water management board elections
- Handled over 70,000 votes
- Uses a publicly-known hash-function and voter-specific keys
- Obvious receipt

How it works:

example: RIES

- Used in Dutch water management board elections
- Handled over 70,000 votes
- Uses a publicly-known hash-function and voter-specific keys
- Obvious receipt

How it works:

1. $s_{a \to v}$: $key(v)$

example: RIES

- Used in Dutch water management board elections
- Handled over 70,000 votes
- Uses a publicly-known hash-function and voter-specific keys
- Obvious receipt

How it works:

1. $s_{a \to v}$: $key(v)$
2. a: publish list of all possible encrypted votes, hashed:
   $\mathcal{L} = \bigcup_{v \in \mathcal{V}} \{ \langle h(\{c\}_{key(v)}), c \rangle \mid c \in \mathcal{C} \}$

example: RIES

- Used in Dutch water management board elections
- Handled over 70,000 votes
- Uses a publicly-known hash-function and voter-specific keys
- Obvious receipt

How it works:

1. $s_{a \to v}$: $key(v)$
2. a: publish list of all possible encrypted votes, hashed:
   $\mathcal{L} = \bigcup_{v \in \mathcal{V}} \{ \langle h(\{c\}_{key(v)}), c \rangle \mid c \in \mathcal{C} \}$
3. $p_{v \to a}$: $\{\Gamma(v)\}_{key(v)}$

example: RIES

- Used in Dutch water management board elections
- Handled over 70,000 votes
- Uses a publicly-known hash-function and voter-specific keys
- Obvious receipt

How it works:

1. $s_{a \to v}$: $key(v)$
2. a: publish list of all possible encrypted votes, hashed:
   $\mathcal{L} = \bigcup_{v \in \mathcal{V}} \{\langle h(\{c\}_{key(v)}), c\rangle \mid c \in \mathcal{C}\}$
3. $p_{v \to a}$: $\{\Gamma(v)\}_{key(v)}$
4. a: collect all votes

example: RIES

- Used in Dutch water management board elections
- Handled over 70,000 votes
- Uses a publicly-known hash-function and voter-specific keys
- Obvious receipt

How it works:

1. $s_{a \rightarrow v}$: $key(v)$
2. a: publish list of all possible encrypted votes, hashed:
   $\mathcal{L} = \bigcup_{v \in \mathcal{V}} \{\langle h(\{c\}_{key(v)}), c\rangle \mid c \in \mathcal{C}\}$
3. $p_{v \rightarrow a}$: $\{\Gamma(v)\}_{key(v)}$
4. a: collect all votes
5. a: publish outcome

example: RIES

- Used in Dutch water management board elections
- Handled over 70,000 votes
- Uses a publicly-known hash-function and voter-specific keys
- Obvious receipt

How it works:

1. $s_{a \to v}$: $key(v)$
2. a: publish list of all possible encrypted votes, hashed:
   $\mathcal{L} = \bigcup_{v \in \mathcal{V}} \{\langle h(\{c\}_{key(v)}), c\rangle \mid c \in \mathcal{C}\}$
3. $p_{v \to a}$: $\{\Gamma(v)\}_{key(v)}$
4. a: collect all votes
5. a: publish outcome

Notice a receipt?

receipts in RIES

To prove that v cast a vote for candidate c, it suffices to show an k such that $\langle h(\{c\}_k), c\rangle \in \mathcal{L}$.

This is precisely the voter's key!

receipts in RIES

To prove that v cast a vote for candidate c, it suffices to show an k such that $\langle h(\{c\}_k), c\rangle \in \mathcal{L}$.

This is precisely the voter's key!

This means the following in the formalism:

- $\alpha(x) = x$
- $\beta(x) = x$

receipts in RIES

To prove that v cast a vote for candidate c, it suffices to show an k such that $\langle h(\{c\}_k), c\rangle \in \mathcal{L}$.

This is precisely the voter's key!

This means the following in the formalism:

- $\alpha(x) = x$
- $\beta(x) = x$ ... for suitable $\mathcal{RB}$