technische universiteit eindhoven

Formalising Receipt-freeness

(joint work with Erik de Vink)

Hugo Jonker

hjonker@win.tue.nl

overview

- How does voting work in the “real world”?
- Why vote digital?
- What is this “receipt-freeness” anyway?
- What is this “receipt-freeness” anyway – in a more formal sense?
- Aha! But how would you use this?

typical elections

The type of elections we consider (1V1V):

- Various candidates
- Each voter may cast one vote
- All votes carry equal weight
- The result can be seen as the collection (multiset) of cast votes (ballots)

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.

pro’s & con’s

Advantages:

Disadvantage:

pro's & con's

Advantages:

- Greater convenience for voter ($\stackrel{?}{\Longrightarrow}$ greater voter turnout)
- Less overhead to set up elections

Disadvantage:

pro's & con's

Advantages:

- Greater convenience for voter ($\stackrel{?}{\Longrightarrow}$ greater voter turnout)
- Less overhead to set up elections

Disadvantage: Re-invent the wheel:

- How to do elections in a digital environment?
- What attacks are possible?
- How to prevent those attacks?

Which means:

pro's & con's

Advantages:

- Greater convenience for voter ($\stackrel{?}{\Longrightarrow}$ greater voter turnout)
- Less overhead to set up elections

Disadvantage: Re-invent the wheel:

- How to do elections in a digital environment?
- What attacks are possible?
- How to prevent those attacks?

Which means:

- Danger of introducing new flaws

pro’s & con’s

Advantages:

- Greater convenience for voter ($\stackrel{?}{\Longrightarrow}$ greater voter turnout)
- Less overhead to set up elections

Disadvantage: Re-invent the wheel:

- How to do elections in a digital environment?
- What attacks are possible?
- How to prevent those attacks?

Which means:

- Danger of introducing new flaws
- Risc of forgetting about known flaws

properties

Several properties have been established for e-voting protocols, such as:

properties

Several properties have been established for e-voting protocols, such as:

- Democracy

properties

Several properties have been established for e-voting protocols, such as:

- Democracy
- Eligibility

properties

Several properties have been established for e-voting protocols, such as:

- Democracy
- Eligibility
- Accuracy

properties

Several properties have been established for e-voting protocols, such as:

- Democracy
- Eligibility
- Accuracy
- Verifiability
  - Individual
  - Universal

properties

Several properties have been established for e-voting protocols, such as:

- Democracy
- Eligibility
- Accuracy
- Verifiability
  - Individual
  - Universal
- Privacy

TU/e

properties

Several properties have been established for e-voting protocols, such as:

- Democracy
- Eligibility
- Accuracy
- Verifiability
  - Individual
  - Universal
- Privacy
- Fairness

properties

Several properties have been established for e-voting protocols, such as:

- Democracy
- Eligibility
- Accuracy
- Verifiability
  - Individual
  - Universal
- Privacy
- Fairness
- ...

properties

Several properties have been established for e-voting protocols, such as:

- Democracy
- Eligibility
- Accuracy
- Verifiability
  - Individual
  - Universal
- Privacy
- Fairness
- ...
- Receipt-freeness(!)

intuition

A receipt is an object which enables a voter to prove how she voted.

intuition

A receipt is an object which enables a voter to prove how she voted.

Examples:

Everyone signs their vote.

intuition

A receipt is an object which enables a voter to prove how she 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 1V1V elections
- Quite strict

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. $v \rightarrow a$: blinded, encrypted vote signed by v
3. $a \rightarrow v$: 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. $v \rightarrow a$: 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

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. $v \to a$: blinded, encrypted vote signed by v
3. $a \to v$: blinded, encrypted vote signed by a
4. $v \to cnt$: encrypted vote signed by a
5. cnt: collect all votes
6. cnt: publish list of received votes
7. $v \to cnt$: 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!

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

To denote receipts, the following syntax is used:

- 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:

- $\alpha : \mathcal{R} \to \mathcal{AT}$, 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:

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

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:

- $\alpha : \mathcal{R} \to \mathcal{AT}$, 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: $\beta(r) \in \mathcal{RB}$

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

receipts as terms II

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

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

receipts as terms II

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

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

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

receipts as terms II

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)

receipts as terms II

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.

in advance

- Formalisation not yet complete (for terms)
- Goal in this talk is a high-level analysis using the formalism

- 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:

- 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:

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

Process for a voter:

- 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:

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

Process for a voter:

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

BT: receipt-free

Let's examine the voter process:

BT: receipt-free

Let's examine the voter process:

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

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

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

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

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

No ballot as a subterm

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

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!

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

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

Subterm of first term!

None of the terms from the voter can satisfy $\alpha(t) \in t \wedge \beta(t) \in t$

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

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

Subterm of first term!

None of the terms from the voter can satisfy $\alpha(t) \in t \land \beta(t) \in t$

$\implies$ BT is receipt-free!

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:

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. $a \rightarrow v$: $key(v)$

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. $a \rightarrow v$: $key(v)$
2. a: publish list of all possible encrypted votes, hashed:
   $\mathcal{L} = \bigcup_{v \in \mathcal{V}} \{< h(\{c\}_{key(v)}), c > \ | \ c \in \mathcal{C}\}$

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

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

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

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. $a \rightarrow v$: $key(v)$
2. a: publish list of all possible encrypted votes, hashed:
   $\mathcal{L} = \bigcup_{v \in \mathcal{V}} \{< h(\{c\}_{key(v)}), c > \mid c \in \mathcal{C}\}$
3. $p_{v \rightarrow a}$: $\{\Gamma(v)\}_{key(v)}$
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 $< h(\{c\}_k), c > \in \mathcal{L}$.

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 $< h(\{c\}_k), c >\in \mathcal{L}$.

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 $< h(\{c\}_k), c > \in \mathcal{L}$.

This is precisely the voter's key!

This means the following in the formalism:

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

Conclusions

- We're doing nice work here!
- ... but we're not yet done
- BT, SK95, HS and ALBD analysis indicates receipt-freeness
- RIES and FOO analysis demonstrates receipts
- More information in paper (submitted)...
- ... or the tech report (to appear)

Conclusions

- We're doing nice work here!
- ... but we're not yet done
- BT, SK95, HS and ALBD analysis indicates receipt-freeness
- RIES and FOO analysis demonstrates receipts
- More information in paper (submitted)...
- ... or the tech report (to appear)

Questions?

Conclusions

- We're doing nice work here!
- ... but we're not yet done
- BT, SK95, HS and ALBD analysis indicates receipt-freeness
- RIES and FOO analysis demonstrates receipts
- More information in paper (submitted)...
- ... or the tech report (to appear)

Questions?

Take care of yourself...
... and each other!
Jerry Springer