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?
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.
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 (⇒ greater voter turnout)
- Less overhead to set up elections

Disadvantage:
pro’s & con’s

Advantages:

- Greater convenience for voter (⇒ 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:
Advantages:
- Greater convenience for voter (\(\Rightarrow\) 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
Advantages:
- Greater convenience for voter (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
Several properties have been established for e-voting protocols, such as:
Several properties have been established for e-voting protocols, such as:

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

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

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

- Democracy
- Eligibility
- Accuracy
- Verifiability
  - Individual
  - Universal
Several properties have been established for e-voting protocols, such as:

- Democracy
- Eligibility
- Accuracy
- Verifiability
  - Individual
  - Universal
- Privacy
Several properties have been established for e-voting protocols, such as:

- Democracy
- Eligibility
- Accuracy
- Verifiability
  - Individual
  - Universal
- Privacy
- Fairness
Several properties have been established for e-voting protocols, such as:

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

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

- Democracy
- Eligibility
- Accuracy
- Verifiability
  - Individual
  - Universal
- Privacy
- Fairness
- ...
- Receipt-freeness(!)
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.
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_
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
More precisely: a receipt $r$ proves that a voter $v$ cast a vote for candidate $c$.

- **R1:** $r$ authenticates $v$
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 $c$
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
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
Rough sketch of the FOO protocol for voter $v$, admin $a$ and counter $cnt$:

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

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

Timestamping = no it doesn't!
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 \( a \)

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

Timestamping =

no it doesn’t!
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 $a$
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. $v \rightarrow a$: 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
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
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. $v \rightarrow a$: 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
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 $v \in V$, choices $c \in C$
- ballots $B$ and results (multisets of choices) $\mathcal{M}(C)$
- a set of received ballots $\mathcal{R}B$, from which the result will be computed
- a choice function $\Gamma: V \rightarrow C$, 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 $\Gamma: \mathcal{V} \rightarrow \mathcal{C}$, which specifies how the voters vote

To denote receipts, the following syntax is used:
- the set of receipts $\mathcal{R}$
- $\text{Terms}(v)$, the set of all terms that a voter $v \in \mathcal{V}$ can generate
- authentication terms $\mathcal{AT}(v)$:
  $t \in \mathcal{AT}(v) \implies \forall w \neq v: t \notin \text{Terms}(w)$
- $\text{auth}: \mathcal{AT} \rightarrow \mathcal{V}$, the unique voter that created an AT
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:

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

Formalisation of the requirements:

- $R1: \alpha(r) \in AT(v)$
decomposing receipts

The following functions are used to decompose receipts:

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

Formalisation of the requirements:

- R1: $\alpha(r) \in AT(v)$
- R2: $\gamma(r) = \Gamma(v)$
decomposing receipts

The following functions are used to decompose receipts:

- \( \alpha : R \rightarrow AT \), extract authentication term from receipt
- \( \beta : R \rightarrow RB \), extract ballot from receipt
- \( \gamma : R \rightarrow C \), extract candidate from receipt

Formalisation of the requirements:

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

So, for valid receipts: \( \text{auth}(\alpha(r)) = v \implies \gamma(r) = \Gamma(v) \), which is satisfied by \( \gamma = \Gamma \circ \text{auth} \circ \alpha \).
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. $\mathcal{R} = \emptyset \lor \mathcal{R} \subseteq Terms$).

Now:
- Model the protocol in ACP
- Test suitability of communicated terms as receipts
- Pronounce judgment
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. $\mathcal{R} = \emptyset \lor \mathcal{R} \subseteq Terms$).

Now:
- Model the protocol in ACP (+ tweaks)
- Test suitability of communicated terms as receipts
- Pronounce judgment
Write \( t \in t' \) if \( t \) is a subterm of \( t' \).

\( \alpha, \beta \) extract terms from terms, i.e. they deal with subterms.
Write $t \in t'$ if $t$ is a subterm of $t'$.

$\alpha, \beta$ extract terms from terms, i.e. they deal with subterms.

**Lemma** $\forall t \in R: \alpha(t) \in t \land \beta(t) \in t$
Write \( t \in t' \) if \( t \) is a subterm of \( t' \).

\( \alpha, \beta \) extract terms from terms, i.e. they deal with subterms.

**Lemma** \( \forall t \in \mathcal{R}: \alpha(t) \in t \land \beta(t) \in t \)

(Note that, by definition: \( t \in t' \land t \in \mathcal{AT}(v) \implies t' \in \mathcal{AT}(v) \).
So receipts are themselves authentication terms)
Write $t \in t'$ if $t$ is a subterm of $t'$.

$\alpha, \beta$ extract terms from terms, i.e. they deal with subterms.

**Lemma** $\forall t \in \mathcal{R}: \alpha(t) \in t \land \beta(t) \in t$

(Note that, by definition: $t \in t' \land t \in \mathcal{AT}(v) \implies t' \in \mathcal{AT}(v)$. 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.
- 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:

\[
A(v) = \sum_{x \in E(0), y \in E(1)} s_{a \rightarrow v}(\min(x, y), \max(x, y)) \cdot p_{a \rightarrow v}^*(x \in E(0) \land y \in E(1)) \cdot (r_{v \rightarrow a}(x) + r_{v \rightarrow a}(y))
\]

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:
\[
A(v) = \sum_{x \in E(0), y \in E(1)} s_{a \rightarrow v}(\min(x, y), \max(x, y)) \cdot p_{a \rightarrow v}^*(x \in E(0) \land y \in E(1)) \cdot (r_{v \rightarrow a}(x) + r_{v \rightarrow a}(y))
\]

Process for a voter:
\[
V = \sum_{x, y} r_{a \rightarrow v}(x, y) \cdot \sum_{i \in \{0, 1\}} p_{a \rightarrow v}^*(x \in E(i) \land y \in E(1 - i)) \cdot (\Gamma(v) = i \rightarrow s_{v \rightarrow a}(x) + \Gamma(v) = 1 - i \rightarrow s_{v \rightarrow a}(y))
\]
Let’s examine the voter process:
Let’s examine the voter process:

$$V = \sum_{x, y} r_a \cdot v(x, y).$$
Let’s examine the voter process:

\[ V = \sum_{x,y} r_{a \rightarrow v}(x, y) \cdot \]

Not an authentication term
Let’s examine the voter process:

\[ V = \sum_{x,y} r_{a \rightarrow v}(x, y) \cdot \]

*Not an authentication term*

\[ \sum_{i \in \{0,1\}} p_{a \rightarrow v}^{*}(x \in E(i) \land y \in E(1 - i)) \cdot \]
Let’s examine the voter process:

\[ V = \sum_{x,y} r_{a\rightarrow v}(x, y). \]

*Not an authentication term*

\[ \sum_{i \in \{0,1\}} p_{a\rightarrow v}^*(x \in E(i) \land y \in E(1 - i)). \]

*No ballot as a subterm*
Let’s examine the voter process:

\[ V = \sum_{x,y} r_{a \rightarrow v}(x, y). \]

Not an authentication term

\[ \sum_{i \in \{0, 1\}} p_{a \rightarrow v}^*(x \in E(i) \land y \in E(1 - i)). \]

No ballot as a subterm

\[ (\Gamma(v) = i \rightarrow s_{v \rightarrow a}(x) \quad + \quad \Gamma(v) = 1 - i \rightarrow s_{v \rightarrow a}(y) ) \]
Let’s examine the voter process:

\[
V = \sum_{x,y} r_{a \rightarrow v}(x, y).
\]

*Not an authentication term*

\[
\sum_{i \in \{0,1\}} p^*_{a \rightarrow v}(x \in E(i) \land y \in E(1 - i)).
\]

*No ballot as a subterm*

\[
(\Gamma(v) = i \rightarrow s_{v \rightarrow a}(x) \quad + \quad \Gamma(v) = 1 - i \rightarrow s_{v \rightarrow a}(y))
\]

*Subterm of first term!*
BT: receipt-free

Let’s examine the voter process:

\[ V = \sum_{x,y} r_{a \rightarrow v}(x, y). \]

Not an authentication term

\[ \sum_{i \in \{0,1\}} p_{a \rightarrow v}^*(x \in E(i) \land y \in E(1 - i)). \]

No ballot as a subterm

\[ \left( \Gamma(v) = i \rightarrow s_{v \rightarrow a}(x) \quad + \quad \Gamma(v) = 1 - i \rightarrow s_{v \rightarrow a}(y) \right) \]

Subterm of first term!

None of the terms from the voter can satisfy \( \alpha(t) \in t \land \beta(t) \in t \)
Let’s examine the voter process:

\[ V = \sum_{x,y} r_{a \rightarrow v}(x, y). \]

\textit{Not an authentication term}

\[ \sum_{i \in \{0, 1\}} p_{a \rightarrow v}^*(x \in E(i) \land y \in E(1 - i)). \]

\textit{No ballot as a subterm}

\[ (\Gamma(v) = i \rightarrow s_{v \rightarrow a}(x) \quad \land \quad \Gamma(v) = 1 - i \rightarrow s_{v \rightarrow a}(y)) \]

\textit{Subterm of first term!}

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

\[ \implies \text{BT is receipt-free!} \]
- 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. \( a \rightarrow v: key(v) \)
- 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. \( a: \text{publish list of all possible encrypted votes, hashed:} \)
   \[
   \mathcal{L} = \bigcup_{v \in V} \{< h(\{c\}key(v)), c > | c \in C\}
   \]
- 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. \( a \rightarrow v : \text{key}(v) \)
2. \( a : \text{publish list of all possible encrypted votes, hashed:} \)
   \[ \mathcal{L} = \bigcup_{\nu \in \mathcal{V}} \{ < h(\{c\}\text{key}(\nu)), c > \mid c \in C \} \]
3. \( p_{\nu \rightarrow a} : \{ \Gamma(\nu) \}_\text{key}(\nu) \)
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. $a \rightarrow v: \text{key}(v)$

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

3. $p_{v \rightarrow a}: \{\Gamma(v)\}_{\text{key}(v)}$

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

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