Formalising Receipt-Freeness

Hugo Jonker h.l.jonker@tue.nl
Erik de Vink e.p.d.vink@tue.nl

Evoting

Safe and secure elections over a hostile network

Security properties of evoting protocols include:

- Democracy
- Accuracy
- Individual verifiability
- Universal verifiability
- Privacy
  - voter privacy
  - **receipt-freeness**
  - coercion-resistance

intuition

receipt: proof of how a voter has voted

Non-existent in pre-1994 protocols

Example:

> In the FOO92 protocol, a voter can prove how she voted by disclosing the position of her vote on the published list of received votes and by disclosing the used encryption key.

requirements

"receipt: proof of how a voter has voted"

More precisely:

"receipt r proves that voter v cast *a vote for candidate c"*

This means any receipt must satisfy the following:

requirements

"receipt: proof of how a voter has voted"

More precisely:

"receipt r proves that voter v cast *a vote for candidate c"*

This means any receipt must satisfy the following:

- R1: r authenticates v

requirements

"receipt: proof of how a voter has voted"

More precisely:

"receipt r proves that voter v cast *a vote for candidate c"*

This means any receipt must satisfy the following:

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

requirements

> *"receipt: proof of how a voter has voted"*

More precisely:

> *"receipt r proves that voter v* cast *a vote for candidate c"*

This means any receipt must satisfy the following:

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

ingredients

- voters $v \in \mathcal{V}$, choices $c \in \mathcal{C}$
- ballots $\mathcal{B}$ and results $\mathcal{M}(\mathcal{C})$
- received ballots $\mathcal{RB}$, from which the result will be computed
- choice function $\Gamma : \mathcal{V} \rightarrow \mathcal{C}$ specifying how voters vote

ingredients

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

To denote receipts, the following syntax is used:

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

decomposing receipts

The following functions are used to decompose receipts:

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

Formalisation of requirements R1-3 for receipt r:

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 requirements R1-3 for receipt r:

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

decomposing receipts

The following functions are used to decompose receipts:

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

Formalisation of requirements R1-3 for receipt r:

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

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 requirements R1-3 for receipt r:

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

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 requirements R1-3 for receipt r:

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

For valid receipts: $auth(\alpha(r)) = v \implies \gamma(r) = \Gamma(v)$
Sufficient: $\gamma = \Gamma \circ auth \circ \alpha$.

receipts as terms

Receipts must be derivable from an execution of a protocol.

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

Analyzing protocols:

- 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: $at \in t' \wedge at \in \mathcal{AT}(v) \implies t' \in \mathcal{AT}(v)$.
Therefore, receipts are 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: $at \in t' \wedge at \in \mathcal{AT}(v) \implies t' \in \mathcal{AT}(v)$.
Therefore, receipts are authentication terms)

This does not capture the entire notion of receipts,
but turns out to be strong enough in the examined cases.

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:

- 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:

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

Process for a voter:

- 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:

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

Process for a voter:

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

BT: receipt-free

Let’s examine the voter process:

BT: receipt-free

Let's examine the voter process:

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

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

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

Subterm of first term!

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 these terms can satisfy the lemma!

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

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

Subterm of first term!

None of these terms can satisfy the lemma!

Thus: BT is receipt-free.

Conclusions

- A constructive approach to uncovering receipts
- But limited to terms
- BT, SK95, HS and ALBD analysis indicates receipt-freeness
- RIES and FOO analysis demonstrates receipts
- Further details in paper

Conclusions

- A constructive approach to uncovering receipts
- But limited to terms
- BT, SK95, HS and ALBD analysis indicates receipt-freeness
- RIES and FOO analysis demonstrates receipts
- Further details in paper

Thank you for your attention!

h.l.jonker@tue.nl
www.win.tue.nl/~hjonker

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. $v \rightarrow a$: 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. $v \longrightarrow a$: 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. $v \longrightarrow a$: 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

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
Using timestamping on the receipt $\Longrightarrow$ no loss of validity

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:
   $\mathcal{L} = \bigcup_{v \in \mathcal{V}} \{ \langle h(\{c\}_{key(v)}), c \rangle \mid c \in \mathcal{C} \}$

- 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. $a \longrightarrow v$: $key(v)$
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. $p_{v \rightarrow a}$: $\{\Gamma(v)\}_{key(v)}$

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

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

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. $a \longrightarrow v$: $key(v)$
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 $\mathcal{RB}$