An Internet Voting Protocol with Distributed Verification Receipt Generation

Kristjan Krips[1,4] **Ivo Kubjas**[2,4] Jan Willemson[1,3]

[1]Cybernetica AS

[2]Smartmatic-Cybernetica Centre of Excellence for Internet Voting

[3]Software Technology and Applications Competence Center

[4]Institute of Computer Science, University of Tartu

October 3th, 2018

Introduction

- Based on Tivi
- Used for Utah GOP caucus in 2016
- 27490 pre-registered voters, 24486 cast votes
- Receipt based verifiability

Integrity requirements

Requirement (Eligibility)

For every ballot in the final tally, there must exists a valid and registered voter who has cast a vote for a specific choice.

Requirement (Tally integrity)

After the voter has received a confirmation from the server that the submitted ballot has been stored, the stored ballot must be included in the final tally.

Requirement (Ballot well-formedness)

A ballot is a unique one-to-one representation of the voter's choice. The algorithms for encoding and decoding the vote are well-defined and correct.

Verifiability requirements

Requirement (Inclusion verifiability)

Each voter can verify that her ballot has been included in the final tally.

Requirement (Liability provability)

The receipt must include information which would allow to determine liability in the case when receipt verification fails.

Requirement (Voter's privacy)

Voter's choice must not become known to the election organizers unless the voter challenges the receipt.

Auditability requirements

Requirement (Eligibility auditability)

The auditor must be able to verify that only eligible voters have been able to vote and that no additional ballots have been added to the tally.

Requirement (Decryption auditability)

The auditor must be able to verify that the decryption operation is performed correctly on the encrypted ballots.

Requirement (Privacy-preserving auditing)

Auditing the election process must not threaten voters' privacy.

Assumptions

Assumption (Existence of trust base)

The participants belonging to the trust base behave according to the protocol and do not leak their private information.

Assumption (Existence of a RABB and ROBB)

There exist append-only strictly linear immutable bulletin board and read-only bulletin board.

Assumption (Existence of voter certification)

The election organizer has prior knowledge of every eligible voter and that there is an independent certification authority which provides confirmation of their identities.

Assumption (Existence of pre-channel to voters)

There exists an authenticated and secure pre-channel between every voter and the election organizer.

Participants

- Voter (VTR)
- Voting client (VC)
- Voting server (VS)
- Tallying server (TS)
- **Keyholder** (KH)
- **Read-append bulletin board** (RABB)
- Read-only bulletin board (ROBB)
- **Certification authority** (CA)

Protocol (init. and submission)

- Distribution of keys
  - Keyholder generates $(\mathsf{ek}, \mathsf{dk})$
  - Certification authority generate $(\mathsf{sk}_{\mathsf{CA}}, \mathsf{vk}_{\mathsf{CA}})$
  - For every voter $(\mathsf{sk}_{\mathsf{VTR}}, \mathsf{vk}_{\mathsf{VTR}}, c_{\mathsf{VTR}})$
  - **Voting server generates** $(\mathsf{sk}_{\mathbf{VS}}, \mathsf{vk}_{\mathbf{VS}})$
- **Vote submission and receipt generation**
  - Receipt as an cryptogram
  - Malicious voting server can deceive the voter
  - Several malicious voting clients can deceive the voter
  - Combined receipt generation
  - Ballot stored with RABB and voting application sent identifier

Protocol (receipt generation)

VTR	VC	VS	CA
$\xrightarrow{\mathsf{sk}_{\mathsf{VTR}}}$	$\mathsf{vk}_{\mathsf{VTR}} = \mathsf{GetVer}(\mathsf{sk}_{\mathsf{VTR}})$	$\xrightarrow{\mathsf{vk}_{\mathsf{VTR}}}$	Get c_{VTR}
	$r_1, \omega \leftarrow_{\$} \Omega$	$\xleftarrow{c_{\mathsf{VTR}}}$	
	$\mathsf{rt}_1 = (g^{\omega}, g^{r_1} \cdot \mathsf{ek}^{\omega})$	$\xrightarrow{\mathsf{rt}_1}$ $r_2 \leftarrow_{\$} \Omega$	
		$\mathsf{rt}_2 = (g^0, g^{r_2} \cdot \mathsf{ek}^0)$	
		$\mathsf{rt}' = \mathsf{rt}_1 \cdot \mathsf{rt}_2$	
		$\mathsf{s}_{\mathsf{rt}'} = \mathsf{Sig}(\mathsf{sk}_{\mathsf{VS}}, \mathsf{rt}')$	
	$\mathsf{rt} = (g^{\omega}, g^{r_1+r_2} \cdot \mathsf{ek}^{\omega})$	$\xleftarrow{\mathsf{s}_{\mathsf{rt}'}, r_2}$ Store $\mathsf{rt}', \mathsf{s}_{\mathsf{rt}'}$	
	Halt if not $\mathsf{Vf}(\mathsf{vk}_{\mathsf{VS}}, \mathsf{s}_{\mathsf{rt}'}, \mathsf{rt})$		
$\xrightarrow{\mathsf{choice}}$	$r_3 \leftarrow_{\$} \Omega$		
	$\mathsf{pt} = \mathsf{Encode}(\mathsf{choice})$		
	$\mathsf{ct} = (g^{r_3}, \mathsf{pt} \cdot \mathsf{ek}^{r_3})$		
	$\mathsf{bt} = (\mathsf{ct}, \mathsf{rt})$		
	$\mathsf{s}_{\mathsf{bt}} = \mathsf{Sig}(\mathsf{sk}_{\mathsf{VTR}}, \mathsf{bt})$	$\xrightarrow{\mathsf{bt}, \mathsf{s}_{\mathsf{bt}}, \mathsf{vk}_{\mathsf{VTR}}, c_{\mathsf{VTR}}}$	
		$(\mathsf{ct}, \mathsf{rt}) = \mathsf{Split}(\mathsf{bt})$	
		Halt if not $\mathsf{Vf}(\mathsf{vk}_{\mathsf{CA}}, c_{\mathsf{VTR}}, \mathsf{vk}_{\mathsf{VTR}})$	
		Halt if not $\mathsf{Vf}(\mathsf{vk}_{\mathsf{VTR}}, \mathsf{s}_{\mathsf{bt}}, \mathsf{bt})$	
		Halt if $\mathsf{rt} \neq \mathsf{rt}'$	
		Halt if not IsNewVoter	
		Store $\mathsf{bt}, \mathsf{s}_{\mathsf{bt}}, \mathsf{vk}_{\mathsf{VTR}}, c_{\mathsf{VTR}}$	

Protocol (finalization)

- Ballot box decryption
  - Ballot anonymization
  - Shuffling
  - Choice and receipt provably decrypted
  - **Pairs of choice and receipt published on ROBB**
- **Vote verification**
  - ROBB public as a key-value database
  - Voter queries using receipt
  - Choice comparison
  - Additional dispute resolution

Requirements implementation

- *Req. 1* (eligibility) - CA gives signing keys only to eligible voters and RABB content can be audited
- *Req. 2* (tally integrity) - skipping ballot storage with RABB detectable during ballot submission. Skipping ballot tallying detectable using verification.
- *Req. 3* (ballot well-formedness) - ✓
- *Req. 4* (inclusion verifiability) - using post-election verification
- *Req. 6* (voter's privacy) - keyholder gives access to decryption key only for decryption. Ballots are anonymized and shuffled.
- *Req. 7* (eligibility auditability) - auditor has a view of RABB and decrypted ballots. Additionally will check the signatures and certificates.
- *Req. 8* (decryption auditability) - tallying server constructs proof of correct ElGamal decryption.
- *Req. 9* (privacy-preserving auditing) - anonymized ballots are shuffled before decryption.

Dispute resolution

1. The values on voter's receipt are compared to elements on RABB. (voter or VC)
2. The voter's verification key is verified against voter's certificate. (election organizer)
3. The ballot signature is verified using voters signing key. (election organizer)
4. The ballot is split into encrypted vote and encrypted receipt. (election organizer)
5. The signature on the receipt is verified using voting server verification key. (voter or VC)
6. The encrypted receipt is decrypted and compared to values on voter's receipt. (voter or VC)
7. The encrypted vote is decrypted and decoded. The choice is compared to the one on ROBB. (election organizer)

Assumptions for modelling

Assumption (Perfectly binding signature scheme)

For any generated signature s *on a message* pt *there does not exist another message* pt' *such that verification succeeds.*

Assumption (Perfectly hiding encryption scheme)

Given a ciphertext ct *on a message* pt, *no adversary can learn the encrypted message.*

Assumption (No cooperation between adversarial parties)

No two adversarial parties cooperate.

Formal model

- Computational model using EasyCrypt
- Available at `https://github.com/krips/uvoting`
- Modeled with a single voter and without mixnet

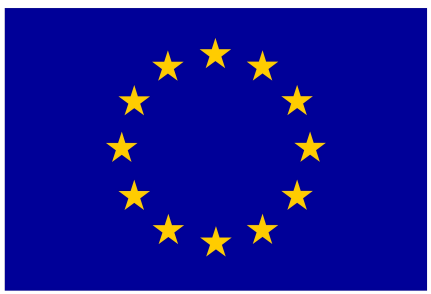
- Lemma `votingCorrectness` shows correctness
- Lemma `badClientSuccess` shows that malicious client only succeeds when does not change the vote
- Lemma `voteDeletingVServer` shows that voting server can not remove votes
- Lemma `voteDeletingTally` shows that tallying server can not skip ballots

Clash attack

- In the described form, proving security against clash attacks did not succeed.
- Implied possibility of clash attack.
- Attack proved as a lemma `clashAttack`.
- Attack can be mitigated by changing the voting client interaction.
- Can we assume that the voter knows how the voting client interacts?

Acknowledgements

The work was supported by the Estonian Research Council under Institutional Research Grant IUT27-1.
The work was supported by the European Regional Development Fund through the Estonian Centre of Excellence in ICT Research (EXCITE) under grant number EU48684.



European Union
European Regional
Development Fund

Investing
in your future