Improved Coercion-Resistant Electronic Elections through Deniable Re-Voting

Jörn Mülle-Quade\textsuperscript{1}, Dirk Achenbach\textsuperscript{1}, Carmen Kempka\textsuperscript{2}, Bernhard Löwe\textsuperscript{1}

\textsuperscript{1)} Karlsruhe Institute of Technology
\textsuperscript{2)} NTT Secure Platform Laboratories
Introduction

What we want to achieve

- **Coercion-resistance**: Even a fully cooperating voter can not convince the adversary that she has followed his instructions in any way which affects her choice.

- **Verifiable Correctness**: Every voter can verify that her ballot is included in the tally and processed correctly, and that the tally result is computed correctly.
Introduction

What we want to achieve

- **Coercion-resistance**: Even a fully cooperating voter can not convince the adversary that she has followed his instructions in any way which affects her choice.

- **Verifiable Correctness**: Every voter can verify that her ballot is included in the tally and processed correctly, and that the tally result is computed correctly.
How to Achieve Coercion-Resistance

How can we defend against observation during the voting process?

- Fake voting credentials, panic passwords
- Our approach: do nothing, then just revote
How to Achieve Coercion-Resistance

How can we defend against observation during the voting process?

- Fake voting credentials, panic passwords
- Our approach: do nothing, then just revote

Introduction

Jörn Müller-Quade, Dirk Achenbach, Carmen Kempka, Bernhard Lüwe – Improved Coercion-Resistant Electronic Elections through Deniable Re-Voting

Conclusion
How to Achieve Coercion-Resistance

How can we defend against observation during the voting process?

- Fake voting credentials, panic passwords
- Our approach: do nothing, then just *revote*

Introduction

Jörn Müller-Quade, Dirk Achenbach, Carmen Kempka, Bernhard Löwe – Improved Coercion-Resistant Electronic Elections through Deniable Re-Voting

Deniable Revoting

Conclusion
Our Contribution

- Modification of the voting scheme of Juels, Catalano and Jakobsson
  - to allow *deniable* revoting
  - instead of or in addition to fake credentials

- First revoting solution which simultaneously offers...
  - ...deniability of the revoting process
  - ...verifiable correctness of the processing of revotes
  - without demanding of the voter to safe state between votes.

- Adaption of the security model of Juels at al. to allow revoting

- Proof of security of our voting scheme
Our Contribution

- Modification of the voting scheme of Juels, Catalano and Jakobsson
  - to allow *deniable* revoting
  - instead of or in addition to fake credentials
- First revoting solution which simultaneously offers...
  - ...deniability of the revoting process
  - ...verifiable correctness of the processing of revotes
  - without demanding of the voter to safe state between votes.
- Adaption of the security model of Juels at al. to allow revoting
- Proof of security of our voting scheme
Our Contribution

- Modification of the voting scheme of Juels, Catalano and Jakobsson
  - to allow *deniable* revoting
  - instead of or in addition to fake credentials
- First revoting solution which simultaneously offers...
  - ...deniability of the revoting process
  - ...verifiable correctness of the processing of revotes
  - without demanding of the voter to safe state between votes.
- Adaption of the security model of Juels at al. to allow revoting
- Proof of security of our voting scheme
Revoting vs. Fake Credentials

Fake credentials pros and cons:

+ Robust against adversary who demands the secret key

- No sound feedback whether authentication has been successful
- Voter needs to be able to create a fake credential “on the fly”, voter needs to run a coercion evasion strategy “online” during coercion.
Fake credentials pros and cons:

+ Robust against adversary who demands the secret key

- No sound feedback whether authentication has been successful
- Voter needs to be able to create a fake credential “on the fly”, voter needs to run a coercion evasion strategy “online” during coercion.
Revoting vs. Fake Credentials

Revoting pros and cons:

+ Sound feedback
+ No evasion strategy during coercion
+ Applicable in elections where deniable revoting is necessary

- Inalienable credentials: Voter must not give away his secret key
- Adversarial observation has to end before the voting phase ends
Revoting vs. Fake Credentials

Revoting pros and cons:

+ Sound feedback
+ No evasion strategy during coercion
+ Applicable in elections where deniable revoting is necessary

- Inalienable credentials: Voter must not give away his secret key
- Adversarial observation has to end before the voting phase ends
Coercion-resistance:

- The adversary must not see whether the voter has overwritten her ballot. 
  ⇒ Revoting needs to be *deniable*.

- 1009 attack (Warren Smith): The adversary must not even see *how often* a ballot was cast using the same credential.

At the same time, we need *verifiable correctness*:

- Of each voter, only one vote - the last - must count.
- Correct handling of the revotes needs to be proven.
Coercion-resistance:

- The adversary must not see whether the voter has overwritten her ballot. 
  ⇒ Revoting needs to be *deniable*.

- 1009 attack (Warren Smith): The adversary must not even see *how often* a ballot was cast using the same credential.

At the same time, we need *verifiable correctness*:

- Of each voter, only one vote - the last - must count.
- Correct handling of the revotes needs to be proven.
Revoting: Attacks and Challenges

Coercion-resistance:
- The adversary must not see whether the voter has overwritten her ballot.
  ⇒ Revoting needs to be *deniable*.
- 1009 attack (Warren Smith): The adversary must not even see *how often* a ballot was cast using the same credential.

At the same time, we need *verifiable correctness*:
- Of each voter, only one vote - the last - must count.
- Correct handling of the revotes needs to be proven.
Our Algorithm…

- ...starts with a list of encrypted ballots,
- ...and ends with a weeded list of encrypted ballots, containing only the newest ballot of each voter.
- Security is proven up to this point.
- The tally of the weeded encrypted ballots can be done with standard techniques.
**Attack Model**

<table>
<thead>
<tr>
<th>pre-election phase</th>
<th>voting phase</th>
<th>post-election phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>voter list</td>
<td>ballot creation and ballot casting</td>
<td>deleting overwritten ballots</td>
</tr>
<tr>
<td>voter registration</td>
<td>tallying</td>
<td></td>
</tr>
<tr>
<td>candidate list</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Key Steps:**
- **trustworthy KeyGen**
- **full coercion / observation**
- **full coercion / observation**
- **time to recast a vote without observation**

**Introduction**

Jörn Müller-Quade, Dirk Achenbach, Carmen Kempka, Bernhard Löwe – Improved Coercion-Resistant Electronic Elections through Deniable Re-Voting
Deniable Revoting: Overview

**Phase 1:** Casting

**Phase 2:** Signature checking

**Phase 3:** Sort out old ballots

**Phase 4:** Open identities

**Phase 5:** Unlink, open ballots and tally
Phase 1 and 2: Ballot Casting

*pk = signature verification key, v = vote, ts = timestamp*

List of Ballots:

\[ b_1 = (E(v_1), E(pk_1), ts_1), \pi_1 \]
\[ b_2 = (E(v_2), E(pk_2), ts_2), \pi_2 \]
\[
\]

- **Phase 1 (Casting):** Voter creates NIZK-proofs \( \pi \) ...  
  - ... of knowledge of signature \( \sigma \) with \( \text{verify}_{pk'}(ballot, \sigma) = 1 \)  
  - ... that \( E(pk) \) contains the key \( pk' \) used in the proof above  

- **Phase 2 (pre-weeding):**  
  - The NIZK-proofs \( \pi \) are checked.  
  - Ballots with invalid proofs are marked invalid and not considered any further.
Phase 1 and 2: Ballot Casting

\( pk \) = signature verification key, \( v \) = vote, \( ts \) = timestamp

List of Ballots:

\[ b_1 = (E(v_1), E(pk_1), ts_1), \pi_1 \]
\[ b_2 = (E(v_2), E(pk_2), ts_2), \pi_2 \]
\[ \ldots \]

- Phase 1 (Casting): Voter creates NIZK-proofs \( \pi \) \ldots
  - \ldots of knowledge of signature \( \sigma \) with \( \text{verify}_{pk'}(ballot, \sigma) = 1 \)
  - \ldots that \( E(pk) \) contains the key \( pk' \) used in the proof above

- Phase 2 (pre-weeding):
  - The NIZK-proofs \( \pi \) are checked.
  - Ballots with invalid proofs are marked invalid and not considered any further.
Phase 1 and 2: Ballot Casting

\( pk = \) signature verification key, \( v = \) vote, \( ts = \) timestamp

List of Ballots:
\[ b_1 = (E(v_1), E(pk_1), ts_1), \pi_1 \]
\[ b_2 = (E(v_2), E(pk_2), ts_2), \pi_2 \]
\[ \ldots \]

- **Phase 1 (Casting):** Voter creates NIZK-proofs \( \pi \ldots 

  - \ldots \) of knowledge of signature \( \sigma \) with \( verify_{pk'}(ballot, \sigma) = 1 \)
  - \ldots that \( E(pk) \) contains the key \( pk' \) used in the proof above

- **Phase 2 (pre-weeding):**
  - The NIZK-proofs \( \pi \) are checked.
  - Ballots with invalid proofs are marked invalid and not considered any further.
Phase 3: Weeding of old Ballots

\[ pk = \text{signature verification key}, \ v = \text{vote}, \ ts = \text{timestamp} \]

List of Ballots:
\[ b_1 = (E(v_1), E(pk_1), ts_1) \]
\[ b_2 = (E(v_2), E(pk_2), ts_2) \]
\[ \ldots \]

- **Phase 3:** Older ballots are sorted out.
Weeding old Ballots: Comparing Identities

Encrypted Plaintext Equality Tests (EPETs) on the credentials:

\[ \text{EPET}(pk_i, pk_j) = \text{Enc}(\left( \frac{pk_i}{pk_j} \right)^R ) = \begin{cases} \text{Enc}(1) & \text{if } pk_i = pk_j \\ \text{Enc}(r) & \text{if } pk_i \neq pk_j \end{cases} \]
How to Accumulate the Differences?

Can we use the homomorphic property of the encryption?
How to Accumulate the Differences?

<table>
<thead>
<tr>
<th>c_i</th>
<th>pk_A</th>
<th>Time</th>
<th>C_1</th>
<th>C_2</th>
<th>C_3</th>
<th>C_4</th>
<th>C_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_1</td>
<td>pk_A</td>
<td>07:08</td>
<td>81</td>
<td>53</td>
<td>1</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>c_2</td>
<td>pk_B</td>
<td>09:13</td>
<td>46</td>
<td>418</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c_3</td>
<td>pk_C</td>
<td>12:25</td>
<td>49</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c_4</td>
<td>pk_A</td>
<td>13:37</td>
<td></td>
<td></td>
<td></td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>c_5</td>
<td>pk_D</td>
<td>17:42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Can we use the homomorphic property of the encryption?
How to Accumulate the Differences?

We can use the homomorphic property of the encryption...

\[
Enc(25) = Enc(81) \cdot Enc(53) \cdot Enc(1) \cdot Enc(48) \mod N \\
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
Enc(139) = Enc(1) \cdot Enc(1) \cdot Enc(139) \cdot Enc(1) \mod N
\]

...if we swap the encryption of an arbitrary number with an encryption of a 1 and vice versa.
Preperation for Conversion

Form tuples $(Enc(ts), d_{ij})$, where $d_{ij} = EPET(pk_i, pk_j)$. 

<table>
<thead>
<tr>
<th>$C_i$</th>
<th>$pk_i$</th>
<th>$t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$pk_A$</td>
<td>07:08</td>
</tr>
<tr>
<td>2</td>
<td>$pk_B$</td>
<td>09:13</td>
</tr>
<tr>
<td>3</td>
<td>$pk_C$</td>
<td>12:25</td>
</tr>
<tr>
<td>4</td>
<td>$pk_A$</td>
<td>13:37</td>
</tr>
<tr>
<td>5</td>
<td>$pk_D$</td>
<td>17:42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$C_i$</th>
<th>$pk_i$</th>
<th>$t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$pk_A$</td>
<td>07:08</td>
</tr>
<tr>
<td>2</td>
<td>$pk_B$</td>
<td>09:13</td>
</tr>
<tr>
<td>3</td>
<td>$pk_C$</td>
<td>12:25</td>
</tr>
<tr>
<td>4</td>
<td>$pk_A$</td>
<td>13:37</td>
</tr>
<tr>
<td>5</td>
<td>$pk_D$</td>
<td>17:42</td>
</tr>
</tbody>
</table>
No matter how we convert, we have that either $d_{ij} = 1$ or $d'_{ij} = 1$. Set $(a, b) := \text{shuffle}(d_{ij}, d'_{ij})$, and show with PET that either $ab = a$, and $ab \neq b$, or vice versa.
No matter how we convert, we have that either \(d_{ij} = 1\) or \(d'_{ij} = 1\). Set \((a, b) := \text{shuffle}(d_{ij}, d'_{ij})\), and show with PET that either \(ab = a\), and \(ab \neq b\), or vice versa.
Conversion

A coordinator reencrypts all $d_{ij}$, and sends them to the voting authority in random order, mixed with fake differences. The voting authority converts them to $d'_{ij}$ “by hand” (decrypt - convert - encrypt). The converted $d_{ij}$ will also act as fake differences.

Converted fake values are discarded.
Prove $d_{ij}d'_{ij} = d_{ij}$ or $d'_{ij}$ for the real ones.
Sort Back and Accumulate

<table>
<thead>
<tr>
<th>Time</th>
<th>Vote</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:13</td>
<td>1</td>
</tr>
<tr>
<td>07:08</td>
<td>1</td>
</tr>
<tr>
<td>07:08</td>
<td>1</td>
</tr>
<tr>
<td>07:08</td>
<td>139</td>
</tr>
<tr>
<td>07:08</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \begin{array}{ll}
    c_1 & \text{pk}_A \ 07:08 \\
    c_2 & \text{pk}_B \ 09:13 \\
    c_3 & \text{pk}_C \ 12:25 \\
    c_4 & \text{pk}_A \ 13:37 \\
    c_5 & \text{pk}_D \ 17:42 \\
\end{array} \]

The first ballot has been overwritten.

The first ballot has been overwritten.

Introduction

Deniable Revoting

Conclusion
<table>
<thead>
<tr>
<th>c_i</th>
<th>pk_A</th>
<th>Time</th>
<th>1</th>
<th>1</th>
<th>139</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_2</td>
<td>pk_B</td>
<td>09:13</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>c_3</td>
<td>pk_C</td>
<td>12:25</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c_4</td>
<td>pk_A</td>
<td>13:37</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c_5</td>
<td>pk_D</td>
<td>17:42</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The first ballot has been overwritten, and is therefore omitted. The remaining ballots are ready for tallying, using standard techniques.
We introduced...

- ... Deniable revoting as an alternative/addition to fake credentials
- ... Showed that deniable revoting is possible while maintaining public verifiability
- Security is proven in an adapted version of the model of Juels, Catalano and Jakobsson
Thank you very much!

Thank you!