Improved Coercion-Resistant Electronic Elections through Deniable Re-Voting

Jörn Müller-Quade[1], Dirk Achenbach[1], Carmen Kempka[2], Bernhard Löwe[1]

Karlsruhe Institute of Technology, NTT Secure Platform Laboratories

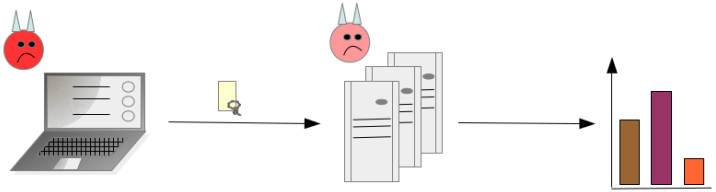


1) Karlsruhe Institute of Technology
2) NTT Secure Platform Laboratories

www.kit.edu
www.seclab.ecl.ntt.co.jp/e/

Introduction

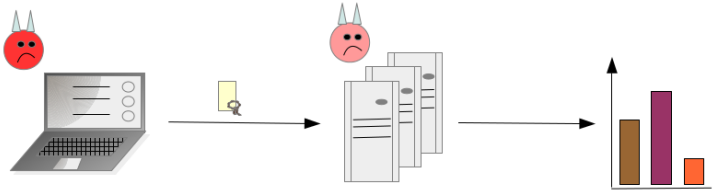
What we want to achieve



- **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

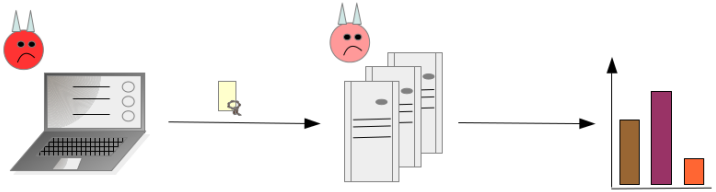
What we want to achieve



- **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

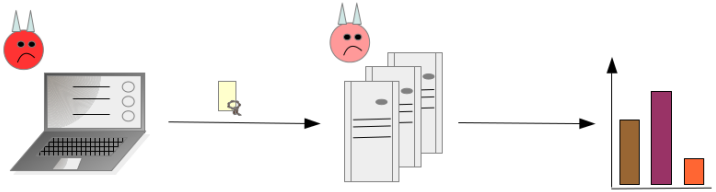
How can we defend against observation during the voting process?



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

How to Achieve Coercion-Resistance

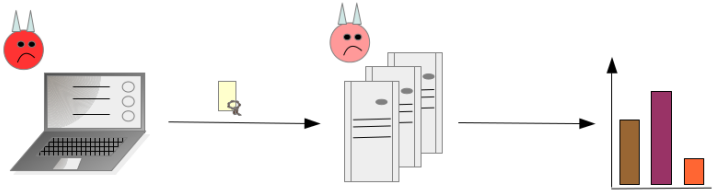
How can we defend against observation during the voting process?



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

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*

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 ceredentials 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

Fake ceredentials 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

Revoting: Attacks and Challenges

Coercion-resistance:

- The adversary must not see whether the voter has overwritten her ballot.
  $\Rightarrow$ 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.
  $\Rightarrow$ 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

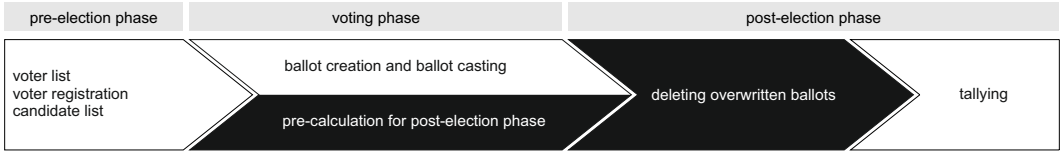
Coercion-resistance:

- The adversary must not see whether the voter has overwritten her ballot.
  $\Rightarrow$ 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.

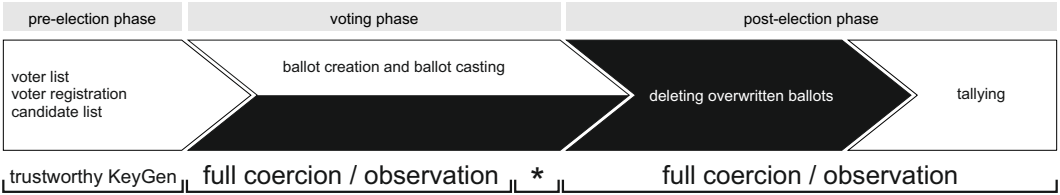
Deniable Revoting



Our Algorithm...

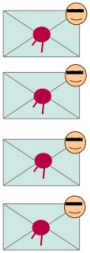
- ...starts with a list of encrypted ballots,
- ...and ends with a weeded list of encrypted ballots, containing only the newest ballot of each voter.
- Security is proven up to this point.
- The tally of the weeded encrypted ballots can be done with standard techniques.

Attack Model

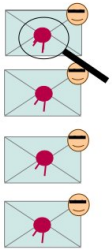


Deniable Revoting: Overview

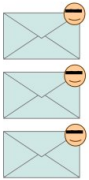
Phase 1: Casting



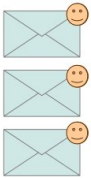
Phase 2: Signature checking



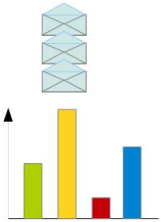
Phase 3: Sort out old ballots



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 π ...
  - ... of knowledge of signature σ with $verify_{pk'}(ballot, \sigma) = 1$
  - ... that $E(pk)$ contains the key pk' used in the proof above
- **Phase 2 (pre-weeding):**
  - The NIZK-proofs π 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:
$b_1 = (E(v_1), E(pk_1), ts_1), \pi_1$
$b_2 = (E(v_2), E(pk_2), ts_2), \pi_2$
...

- **Phase 1 (Casting):** Voter creates NIZK-proofs π ...
  - ... of knowledge of signature σ with $verify_{pk'}(ballot, \sigma) = 1$
  - ... that $E(pk)$ contains the key pk' used in the proof above
- **Phase 2 (pre-weeding):**
  - The NIZK-proofs π 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:
$b_1 = (E(v_1), E(pk_1), ts_1), \pi_1$
$b_2 = (E(v_2), E(pk_2), ts_2), \pi_2$
...

- **Phase 1 (Casting):** Voter creates NIZK-proofs π ...
  - ... of knowledge of signature σ with $verify_{pk'}(ballot, \sigma) = 1$
  - ... that $E(pk)$ contains the key pk' used in the proof above
- **Phase 2 (pre-weeding):**
  - The NIZK-proofs π are checked.
  - Ballots with invalid proofs are marked invalid and not considered any further.

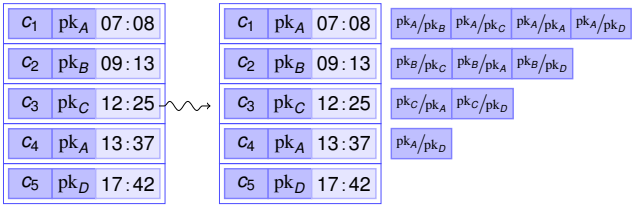
Phase 3: Weeding of old Ballots

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

List of Ballots:
$b_1 = (E(v_1), E(pk_1), ts_1)$
$b_2 = (E(v_2), E(pk_2), ts_2)$
...

- **Phase 3:** Older ballots are sorted out.

Weeding old Ballots: Comparing Identities



Encrypted Plaintext Equality Tests (EPETs) on the credentials:

$$EPET(pk_i, pk_j) = Enc\left(\left(\frac{pk_i}{pk_j}\right)^R\right) = \begin{cases} Enc(1) & \text{if } pk_i = pk_j \\ Enc(r) & \text{if } pk_i \neq pk_j \end{cases}$$

How to Accumulate the Differences?

c_1	pk_A	07:08	81	53	1	48
c_2	pk_B	09:13	46	418	28	
c_3	pk_C	12:25	49	9		
c_4	pk_A	13:37	13			
c_5	pk_D	17:42				

Can we use the homomorphic property of the encryption?

How to Accumulate the Differences?

c_1	pk_A	07:08	81	53	1	48
c_2	pk_B	09:13	46	418	28	
c_3	pk_C	12:25	49	9		
c_4	pk_A	13:37	13			
c_5	pk_D	17:42				

Can we use the homomorphic property of the encryption?

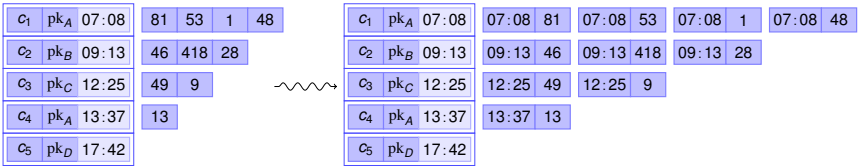
How to Accumulate the Differences?

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

$$
\begin{aligned}
Enc(25) = \; & Enc(81) \cdot Enc(53) \cdot Enc(1) \cdot Enc(48) \mod N \\
& \downarrow \qquad \downarrow \qquad \downarrow \qquad \downarrow \\
Enc(139) = \; & Enc(1) \cdot Enc(1) \cdot Enc(139) \cdot Enc(1) \mod N
\end{aligned}
$$

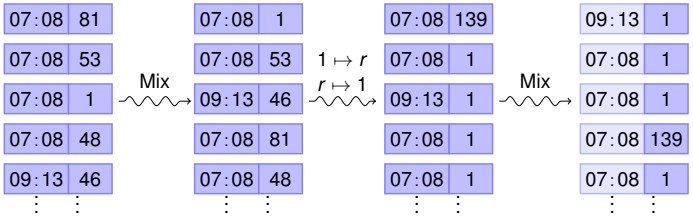
...if we swap the encryption of an arbitrary number with an encryption of a 1 and vice versa.

Preperation for Conversion



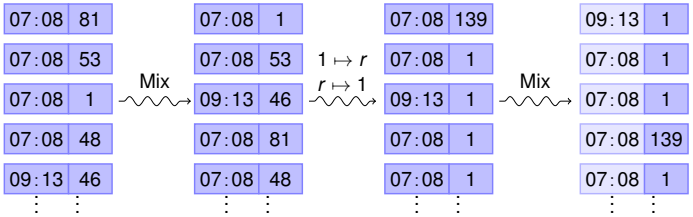
Form tuples $(Enc(ts), d_{ij})$, where $d_{ij} = EPET(pk_i, pk_j)$.

Conversion



No matter how we convert, we have that *either* $d_{ij} = 1$ *or* $d'_{ij} = 1$.
Set $(a, b) := \mathit{shuffle}(d_{ij}, d'_{ij})$, and show with PET that either $ab = a$, and $ab \neq b$, or vice versa.

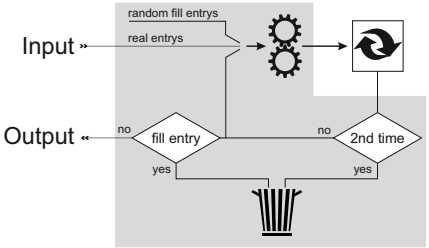
Conversion



No matter how we convert, we have that *either* $d_{ij} = 1$ *or* $d'_{ij} = 1$.
Set $(a, b) := \mathit{shuffle}(d_{ij}, d'_{ij})$, and show with PET that either $ab = a$, and $ab \neq b$, or vice versa.

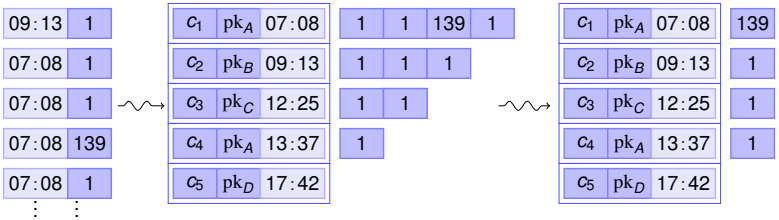
Conversion

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.



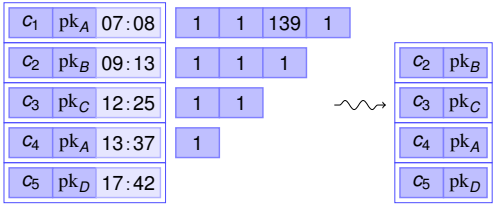
Converted fake values are discarded.
Prove $d_{ij}d'_{ij} = d_{ij}$ or d'_{ij} for the real ones.

Sort Back and Accumulate



The first ballot has been overwritten.

Result



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

Conclusion

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!