A network voting system using a mix-net in a Japanese private organization

Kazue Sako
NEC Corporation
2004.5.27
Background: Electronic Voting in Japan

• Law established in 2001, effective 2002
  – voting at polling place
  – for local government election only
  – no network between polling place and tallying center
  – absentees ballot are still paper-based, all write-ins

• Held in nine local elections
  – Objections raised in two elections
    • Unable to vote over an hour for machine problems
    Mismatch in # of voters and # of votes by 6.
    • 2582 blank votes in a 49 votes difference race (60,000 votes)
Overview of our work

• Aim: a voting system for \textit{private} organization
  – That votes are cast over network
  – That uses \textit{verifiable mix-net} for tallying

• The system was actually used
  – For voting and anonymous surveys
  – With 17,000 eligible voters
  – uses intranet
  – On a regular basis starting Feb 2004. Second vote was held in April 2004, and the third scheduled in June
Technical descriptions

• Universally verifiable mix-net implementation
• History of speed for 10,000 votes, 3 mixers using 3 PC(1Ghz CPU)
  – before 2000: estimation 100hrs cut &choose
  – 2000 implementation: 8 hrs, cut&choose
  – permutation matrix-based proof scheme[Crypto 01]
  – [FC 02] 20 minutes (ordinary Zp*)
  – Now FC02 algorithm implemented using Elliptic Curve 6.5 minutes
Proving a shuffle using Permutation Matrix

A description of a shuffle using matrix

ex) 3 inputs

\[
\begin{pmatrix}
\beta \\
\gamma \\
\alpha
\end{pmatrix} :=
\begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
\alpha \\
\beta \\
\gamma
\end{pmatrix}
\]

\[
\begin{pmatrix}
a & b & c \\
d & e & f \\
g & h & i
\end{pmatrix}
\]
is permutation matrix

for all \((x, y, z)\) the following are satisfied

\[
(ax + by + cz)^3 + (dx + ey + fz)^3 + (gx + hy + iz)^3 = x^3 + y^3 + z^3
\]

\[
(ax + by + cz)^2 + (dx + ey + fz)^2 + (gx + hy + iz)^2 = x^2 + y^2 + z^2
\]
Technical descriptions(II)

• History of permutation matrix-based proof scheme
  (# exponentiations prove+verify, n voters)
  – CRYPTO 01 (9n+12n)
  – FC 02 (9n+10n) merged shuffle+dec proof
  – PKC 04 (8n+6n) with special q
• cf. Groth PKC03 (7n+8n) ZK
• Neff (webpage) (8n+10n) ZK
Why not Zero-knowledge

• Zero knowledge:
  – for any $V^*$, exists a simulator, s.t. no Distinguisher succeeds in distinguishing between a real protocol and simulated result for any input $x$.
  – Our non-ZKIP protocol:
    A distinguisher who can decrypt input encryption can distinguish!
    (ZKIP definition is too strong)
New notion on security

• Whatever adversary can learn about permutation from the protocol is what he could have learned by himself. (permutation hiding)

• All of our scheme satisfies this notion

• Proving and verifying modules are casettable:
Implementational Aspects

- disclaimer: I did not implement all
Mix-net as is described as:

1. Encrypted vote
2. Mixers:
   - Mixer #1
   - Mixer #2
   - Mixer #3
3. Shuffle & Decrypt
4. Result
System Model

Election Policy Committee:
- Determine Policies
- Assign Centers

Output the result of Decryption + Shuffling

List of Encrypted votes

Voting Center:
- Identify voters
- Collect encrypted votes

Shuffling Management Center:
- Decryption + Shuffling

Encoder

Voter

Vote

List of Encrypted votes

Result of decryption
Protocol (Vote Casting)

1. Receive parameters from
2. Encrypt a vote
3. Send it to the Voting Center with a proof of knowledge of the vote $m$ (which prevent the vote duplication attack)
4. Authenticate voter, verify he hasn’t voted before
5. Acknowledge reception

Voting Center

voter
1. Send the list of encrypted votes

2. Perform Shuffle-and-Decrypt

3. Send the result with a proof of correctness, signed

4. Check the signature of mixer₁

5. Verify the proof of SC₁

Protocol (Tallying)
How we modified it to our customer

• They wanted it used their own member authentication system (based on passwords)
• Voters to vote from their PCs: vote casting software in Java Applets
• Members in 6 different divisions: tallying in each divisions
• A mixer is made active only by an operator with a smart card.
• Faster output of outcome. Correctness proofs and verification in an idle time.
• Proofs are locally stored at election committee.
How they liked it

- Flexible number of mixers.
- Speed (3 mixers)
  - Largest (6500 voters) 80 sec tally + 150 sec verify
  - Smallest (700 voters) 13 sec tally + 19 sec verify
- Less claims from its members
- Running cost is 1/10 compared to previous paper voting (mostly manpower cost)
- Invalid ballots were decreased to 1/4.
- Stable show-up rates (80%-85%)
That’s all.
Thank you!