Secure Internet Voting on Limited Devices with Anonymized DSA Public Keys

Rolf Haenni and Oliver Spycher

Bern University of Applied Sciences, Switzerland
http://e-voting.bfh.ch

EVT/WOTE'11, San Francisco

August 9th, 2011

Outline

Outline

Requirements

- Correctness:
  - → Only authorized voters can vote (eligibility)
  - → No voter can vote more than once (uniqueness)
  - → Votes can not be altered (integrity)
  - → All valid votes are counted (completeness)
  - → Invalid votes are not counted (soundness)
- Verifiability: Correctness is publicly verifiable
- Privacy: Votes cannot be linked to voters
- Fairness: No preliminary results are revealed
- Coercion-resistance: Voters cannot be influenced by others

Requirements

- Correctness:
  - → Only authorized voters can vote (eligibility)
  - → No voter can vote more than once (uniqueness)
  - → Votes can not be altered (integrity)
  - → All valid votes are counted (completeness)
  - → Invalid votes are not counted (soundness)
- Verifiability: Correctness is publicly verifiable
- Privacy: Votes cannot be linked to voters
- Fairness: No preliminary results are revealed
- ~~Coercion-resistance: Voters cannot be influenced by others~~

Extended Privacy

- Privacy: Votes cannot be linked to voters
  - → Nobody can learn *how* somebody voted (secrecy)
  - → Nobody can learn *that* somebody voted (anonymity)
- Anonymity is important for fair elections
  - → Take a subset of voters with a predictable voting behavior, e.g. members of a political party
  - → Observe their turnout during the voting period
  - → Mobilize the abstaining party members in case of a low turnout
- The same two properties must hold for any subset of voters

Outline

Introduction

Signature-Based Voting Schemes

Shuffling DSA Public Keys

Protocol Description

Conclusion

Signature-Based Voting Schemes

- To guarantee eligibility, some voting schemes require votes to be digitally signed
- Simplified protocol:
  1. Registration: Establish PKI over electorate
  2. Ballot preparation: Digitally sign encrypted vote
  3. Vote casting: Post ballot to public bulletin board
  4. Pre-tallying: Check signatures
  5. Tallying: Decrypt and count votes
- To guarantee fairness, the decryption key is shared
- To guarantee privacy, additional measures are necessary

Approach 1: Homomorphic Tallying

- Simplified protocol:
  1. Registration: Establish PKI over electorate
  2. Ballot preparation: Digitally sign encrypted vote
  3. Vote casting: Post ballot to public bulletin board
  4. Pre-tallying: Check signatures
  5. Tallying: ~~Decrypt and count votes~~ *Combine encrypted votes and decrypt result*
- To guarantee uniqueness, non-interactive zero-knowledge proofs (NIZKP) must be added to ballots
- NIZKPs are expensive for complex elections (see *Helios*)
- No anonymity

Approach 2: Mixnet-Based Shuffling of Votes

- Simplified protocol:
  1. Registration: Establish PKI over electorate
  2. Ballot preparation: Digitally sign encrypted vote
  3. Vote casting: Post ballot to public bulletin board
  4. Pre-tallying: Check signatures, *shuffle encrypted votes in a verifiable re-encryption mixnet*
  5. Tallying: Decrypt and count votes
- Does not require expensive NIZKPs
- No anonymity

Approach 3: Mixnet-Based Shuffling of Keys

- Simplified protocol:
    1. Registration: Establish PKI over electorate
    2. *Election setup: Anonymize public keys in verifiable mixnet*
    3. Ballot preparation: Digitally sign encrypted vote
    4. Vote casting: Post ballot to public bulletin board *over an anonymous channel*
    5. Pre-tallying: Check signatures *using the anonymous keys*
    6. Tallying: Decrypt and count votes
- Does not require expensive NIZKPs
- Guarantees anonymity

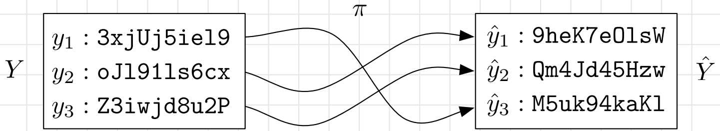
Outline

DSA Signature Scheme

- Standard ElGamal setup:
  - → Large (safe) primes p and q such that $q|p-1$
  - → Generator g of sub-group $G_q \subset \mathbb{Z}_p^*$
  - → Private key: random value $x \in \mathbb{Z}_q$
  - → Public key: $y = g^x \in G_q$
- Signature: $s = (a, b) = \text{Sign}_x(m)$ with
  - → $a = g^r$
  - → $b = (H(m) + a \cdot x) \cdot r^{-1}$
- Verification: $\text{Verify}_y(s, m)$ checks if $a = g^u \cdot y^v$ holds for
  - → $u = H(m) \cdot b^{-1}$
  - → $v = a \cdot b^{-1}$

Shuffling DSA Public Keys

- Input: $Y = (y_1, \ldots, y_n)$ = list of public keys relative to g
- Output: $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_n)$ = list of public keys relative to $\hat{g}$
  - → α = random value from $\mathbb{Z}_q$
  - → $\hat{g} = g^{\alpha}$
  - → π = permutation on $\{1, \ldots, n\}$
  - → $\hat{y}_i = y_{\pi(i)}^{\alpha}$
- This works, because: $\hat{y} = y^{\alpha} = (g^x)^{\alpha} = (g^{\alpha})^x = \hat{g}^x$

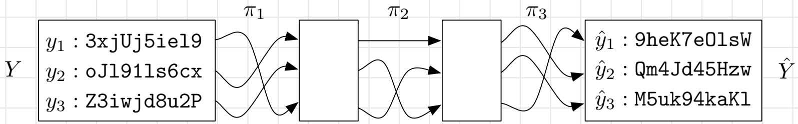


Anonymous DSA Signature Scheme

- Standard ElGamal setup:
  - → Private key: random value $x \in \mathbb{Z}_q$
  - → Public key: $y = g^x \in G_q$
- Anonymous public key: $\hat{y} = y^\alpha$
- New generator: $\hat{g} = g^\alpha$
- Signature: $s = (a, b) = \mathsf{Sign}_x(m)$ with
  - → $a = \hat{g}^r$
  - → $b =$ as defined before
- Verification: $\mathsf{Verify}_{\hat{y}}(s, m)$ checks if $a = \hat{g}^u \cdot \hat{y}^v$ holds for
  - → u, v as defined before

Repeated Shuffling

- To disallow a single shuffling authority to know π or α, let multiple authorities do the shuffling
- Repeated shuffling using $(\alpha_1, \pi_1), \ldots, (\alpha_m, \pi_m)$:
  - → $\alpha = \alpha_1 \cdots \alpha_m$
  - → $\pi = \pi_m \circ \cdots \circ \pi_1$
- Hence, no single party can link the anonymous keys with the public keys



Verifiable Shuffling

- The shuffling authorities must provide NIZKPs for doing the shuffle correctly
- At least three approaches:
  - → Use solution for *"General n-Shuffle Problem"* (Neff, 2001)
  - → Consider y as an ElGamal encryption $e = (1, y)$ and apply re-encryption mixnet (Groth, 2010; Wikström, 2009)
  - → Use *"Randomized Partial Checking"* type of proof (Jakobsson et al., 2002)
- All three approaches require linear-size proofs and linear-time verification

Outline

Protocol Steps (1/2)

1. Registration: Provide voters with key pair x, y (or use existing DSA/ElGamal-based PKI)
2. Election Setup:
   → Publish electoral register Y
   → Perform shuffling and publish $\hat{g}$, $\hat{Y}$, NIZKPs
3. Ballot Preparation:
   → Encrypt vote: $e = \text{Encrypt}(v)$
   → Sign encrypted vote: $s = \text{Sign}_x(e)$ using $\hat{g}$
   → Compute anonymous key $\hat{y} = \hat{g}^x$
   → Compose ballot $B = (e, s, \hat{y})$

Protocol Steps (2/2)

4. Vote Casting: Send $B = (e, s, \hat{y})$ to public bulletin board over an anonymous channel
5. Pre-Tallying: Determine valid ballots
   - → Check if $\hat{y} \in \hat{Y}$
   - → Check if s is a valid signature (using $\hat{g}$)
   - → Check if B is the only ballot for $\hat{y}$ (if not, select one)
6. Tallying: Decrypt and count votes

Optional Protocol Enhancements

- Prevent copying votes from bulletin board
  - → add NIZKP to ballot (knowledge of encryption randomness)
- Avoid decrypting invalid votes
  - → perform efficient PET-based tests (in linear time)
- Protect privacy in case of an imperfect anonymous channel
  - → shuffle the encrypted votes in a re-encryption mixnet

Outline

Conclusion

- Shuffling DSA public keys is an alternative privacy mechanism in remote electronic elections
- If provides an extended notion of privacy:
  - → Secrecy of the vote
  - → Anonymity of the voter
- The main computational task is performed *before* the election
- The voter is not required to produce expensive NIZKPs
- A prototype implementation *"Selectio Helvetica"* is currently under construction (see www.baloti.ch)