Verifiable Internet Elections with Everlasting Privacy and Minimal Trust

Philipp Locher and Rolf Haenni

VoteID 2015, Bern, September 3, 2015

This work is supported by the Swiss National Science foundation, under the grant 200021L-140650/1

Vote Privacy Assumptions

"Any adversary is polynomial-time bounded."

"A threshold number of authorities is trustworthy."

Protocol Overview

- Goal: Make vote privacy independent of
  - computational intractability assumptions
  - trusted authorities
- Involved parties
  - election administration
  - voters
  - public bulletin board
  - verifiers (the public)
- Cryptographic ingredients: perfectly hiding commitments, non-interactive zero-knowledge proofs (NIZKP)

Step 1: Registration

The voter . . .

- creates a pair of private and public credentials
- sends the public credential to the election administration (over an authentic channel)

Step 2: Election Preparation

The election administration ...

- publishes the list of public voter credentials on bulletin board

Step 3: Vote Casting

The voter ...

- creates ballot consisting of
  - vote
  - commitment to public credential
  - election credential
  - NIZKP
- sends ballot to bulletin board (over an anonymous channel)

Step 4: Public Tallying

The verifier . . .

- retrieves the election data from bulletin board
- checks proofs contained in each ballot
- computes the election result

Cryptographic Setup

- Let $\mathcal{G}_p$ be a cyclic group of prime order p with independent generators g_0, g_1
- Let $\mathbb{G}_q \subset \mathbb{Z}_p^*$ be a sub-group of prime order $q \mid (p-1)$ with independent generators $h_0, h_1, \ldots, h_N$
- Assume that DL has no efficient solution in $\mathcal{G}_p$ and $\mathbb{G}_q$

Set Membership Proof

- Goal: prove that a committed value belongs to a given set

$$NIZKP[(u, r) : c = \mathrm{com}_p(u, r) \wedge u \in U]$$

- Secret inputs
  - $u, r \in \mathbb{Z}_p$
- Public inputs
  - Commitment $c = \mathrm{com}_p(u, r) \in \mathcal{G}_p$
  - Set $U = \{u_1, \ldots, u_M\}$ of values $u_i \in \mathbb{Z}_p$

- General Construction
  - Proposed by Brands et al. (2007)
  - Let $P(X) = \prod_{i=1}^{M}(X - u_i)$ satisfying $P(u_i) = 0$ for all $u_i \in U$

$$NIZKP[(u, r) : c = \mathrm{com}_p(u, r) \wedge P(u) = 0]$$

  - Polynomial evaluation proof by Bayer and Groth (2013)

Representation Proof

- Goal: prove that a commitment contains a DL-representation of another committed value

$$NIZKP[(u, r, v_1, \dots, v_N, s) : c = \mathrm{com}_p(u, r) \wedge \\ d = \mathrm{com}_q(v_1, \dots, v_N, s) \wedge u = h_1^{v_1} \cdots h_N^{v_N}]$$

- Secret inputs
  - $u, r \in \mathbb{Z}_p$
  - $v_1, \dots, v_N, s \in \mathbb{Z}_q$
- Public inputs
  - Commitment $c = \mathrm{com}_p(u, r) \in \mathcal{G}_p$
  - Commitment $d = \mathrm{com}_q(v_1, \dots, v_N, s) \in \mathbb{G}_q$

- Au, Susilo, Mu (2010) proposed an extension of the double discrete logarithm proof by Camenisch and Stadler (1997)

Step 1: Registration

The voter ...

- creates a pair of private and public credentials

$$\alpha, \beta \in_R \mathbb{Z}_q$$
$$u = h_1^{\alpha} h_2^{\beta} \in \mathbb{G}_q$$

- sends the public credential u to the election administration (over an authentic channel)

Step 2: Election Preparation

The election administration ...

- defines the list of public voter credentials $U = \{u_1, \ldots, u_M\}$
- computes coefficients $a_0, \ldots, a_M$ of polynomial

$$P(X) = \prod_{i=1}^{M}(X - u_i) = \sum_{i=0}^{M} a_i X^i$$

- selects independent election generator $\hat{h} \in \mathbb{G}_q$
- publishes $(U, a_0, \ldots, a_M, \hat{h})$ on bulletin board

Step 3: Vote Casting

The voter ...

- selects vote e
- computes election credential $\hat{u} = \hat{h}^{\beta}$
- computes commitment $c = \mathrm{com}_p(u, r)$ and $d = \mathrm{com}_q(\alpha, \beta, s)$ to public credential and private credential, respectively
- computes the following proofs:

$$
\begin{aligned}
\pi_1 &= NIZKP[(u, r) : c = \mathrm{com}_p(u, r) \wedge P(u) = 0],\\
\pi_2 &= NIZKP[(u, r, \alpha, \beta, s) : c = \mathrm{com}_p(u, r) \wedge d = \mathrm{com}_q(\alpha, \beta, s)\\
&\qquad\qquad\qquad\qquad \wedge u = h_1^{\alpha} h_2^{\beta}],\\
\pi_3 &= NIZKP[(\alpha, \beta, s) : d = \mathrm{com}_q(\alpha, \beta, s) \wedge \hat{u} = \hat{h}^{\beta}].
\end{aligned}
$$

- sends ballot $B = (e, \hat{u}, c, d, \pi_1, \pi_2, \pi_3)$ to bulletin board (over an anonymous channel)

Step 4: Public Tallying

The verifier ...

- retrieves the election data from bulletin board

$$U, a_0, \ldots, a_M, \hat{h}, \mathcal{B}$$

- checks proofs π_1, π_2, π_3 contained in each ballot $B \in \mathcal{B}$
- detects ballots with identical values $\hat{u}$ and resolve conflicts
- computes the election result from votes v contained in $\mathcal{B}' \subseteq \mathcal{B}$

Adversary Model

- Present adversaries are polynomial-time bounded and thus ...
  - unable to solve DL efficiently in $\mathcal{G}_p$ and $\mathbb{G}_q$
  - unable to compute $\text{hash}^{-1}(h)$
- Future adversaries will have unrestricted computational resources and are therefore
  - able to solve DL efficiently in $\mathcal{G}_p$ and $\mathbb{G}_q$
  - able to compute $\text{hash}^{-1}(h)$

Correctness

Attack by present adversary (during or shortly after election)

- Case 1: Present adversary $\neq$ voter
  - Find representation (α', β') for some $u \in U$
    $\rightarrow$ equivalent to solving DL
  - Simulate π_1, π_2, π_3 without valid secret inputs (α', β')
    $\rightarrow$ equivalent to solving DL or inverting hash function
- Case 2: Present adversary $=$ voter
  - Use different $\beta' \neq \beta$ in a second ballot and simulate π_3
    $\rightarrow$ equivalent to solving DL or inverting hash function

Privacy

Attack by future adversary (possibly in the far future)

- For every $B = (c, d, e, \hat{u}, \pi_1, \pi_2, \pi_3) \in \mathcal{B}$
  - compute β satisfying $\hat{u} = \hat{h}^{\beta}$
  - compute (α', β) satisfying $u' = h_1^{\alpha'} h_2^{\beta}$ for every $u' \in U$
- Therefore, uncovering β from every ballot does not reveal anything about the links between $\mathcal{B}$ and U
- Note that c, d are perfectly hiding and π_1, π_2, π_3 are perfect zero-knowledge

Extensions

- To achieve fairness, the vote e must be encrypted
  - Generate encryption key pair (sk, pk) during election preparation
  - Encrypt vote using pk during vote casting
  - Publish sk to initiate public tallying
- Extended credentials are required to vote multiple times
  - Private credentials $(\alpha, \beta_1, \ldots, \beta_L)$
  - Public credentials $u = h_1^{\alpha} h_2^{\beta_1} \cdots h_{L+1}^{\beta_L}$
  - Use different β_i for each election
- To allow vote updating, some other minor adjustments are necessary

Implementaiton and Performance

- Performance
  - Ballot size: logarithmic to the number of registered voters
  - Ballot generation and verification: logarithmic number of exponentiations and linearithmic number multiplications
- Implementation
  - Prototype implementation in Java
  - Crypto library: UniCrypt

Performance

$M = \|U\|$	Generation	Verification	
		Single Ballot	M Ballots
10	0.7 sec.	0.6 sec.	6.1 sec.
100	0.7 sec.	0.7 sec.	1.1 min.
1'000	0.9 sec.	0.7 sec.	12.2 min.
10'000	2.2 sec.	0.9 sec.	2.6 hours
100'000	17.0 sec.	2.3 sec.	64.8 hours
1'000'000	3.4 min.	15.9 sec.	4417.5 hours

Table 1: Estimated running times for ballot generation and verification for different number of voters.

$M = \|U\|$	Single Ballot	M Ballots
10	39.0 KB	0.4 MB
100	41.6 KB	4.1 MB
1'000	44.3 KB	43.2 MB
10'000	47.8 KB	466.5 MB
100'000	50.4 KB	4.8 GB
1'000'000	53.9 KB	51.4 GB

Table 2: Ballot size for different numbers of voters.

Implemetation

$M = \|U\|$	Ballot Generation	Ballot Verification
10	1.3 sec.	0.9 sec.
100	1.4 sec.	1.0 sec.
1'000	1.6 sec.	1.1 sec.
10'000	3.0 sec.	1.3 sec.
100'000	18.2 sec.	2.9 sec.
1'000'000	3.3 min.	18.8 sec.

Table 3: Actual running times for generating and verifying a single ballot.

Summary

- New approach based on NIZKP
- Pros
  - Everlasting privacy
  - No trusted authorities (except for fairness)
  - Simplicity of voting process
  - Implementation available in UniCrypt
- Cons
  - Anonymous channel required for vote casting
  - Relatively expensive ballot generation/verification
  - Restricted scalability

Outlook

- Optimize the implementation
  - multi-exponentiation
  - fix-base exponentiation
  - parallel execution on multiple cores
  - use polynomial evaluation proof by Brands et al. (2007) when number of registered voters gets very large
- Add receipt-freeness and coercion-resistance

Questions?