

ADDER: A Web-based Internet Voting System

Aggelos Kiayias
Michael Korman
David Walluck

University of Connecticut

June 7, 2005

- 1 Introduction
- 2 Cryptographic Components
- 3 Technical Aspects

Motivation

Internet voting is fraught with perils. We need:

- Privacy
- Transparency
- Auditability
- Well-placed trust

The Need for Open-Source Voting Systems

- When the machinery used to manage an election runs inside a “black box,” there is no way to verify the validity of the election.
- Current e-voting systems often do not have publicly available source code.
- Code may be concealed to avoid the discovery of embarrassing security flaws.
- Even proprietary systems that reveal source code often leave several critical components hidden.
- In order to ensure true democratic elections, voting software must be independently auditable and verifiable by *any* interested third party.

Preliminaries

We will use the ElGamal cryptosystem. To prepare, at the start of an election, the following events occur:

- 1 The server generates a ℓ -bit safe prime number p . Since p is safe, $q = (p - 1)/2$ is prime, as well.
- 2 The server chooses an element $g \in \mathbb{Z}_q^*$ (\mathbb{Z}_q^* is the multiplicative group of the integers modulo q). It is a fact that $\langle g \rangle$ is the subgroup of quadratic residues modulo p .
- 3 The server chooses $a \in_R \mathbb{Z}_q$ and sets $f = g^a$.

Homomorphic Encryption

We use homomorphic encryption as described in [Benaloh87]. Suppose a voter wants to encrypt a vote $v \in \{1, M, M^2, \dots, M^{c-1}\}$.

Algorithm (Additive Homomorphic ElGamal)

- 1 The server chooses $x \in_R \mathbb{Z}_q$ as its private key, and publicizes $h = g^x \pmod p$ as its public key.
- 2 To encrypt, a voter picks $r \in_R \mathbb{Z}_q$ and sends

$$\langle g^r \pmod p, h^r f^v \pmod p \rangle.$$

- 3 To decrypt the message $\langle G, H \rangle$, the server computes

$$f^v = H/G^x \pmod p.$$

The server does a brute-force search on v to find the answer.

Homomorphic Encryption, cont.

Fact

“Additive homomorphic ElGamal” is homomorphic.

Proof.

To illustrate, suppose we have two ballots, an encryption of v_1 and an encryption of v_2 . Then,

$$\begin{aligned}\mathcal{E}(r_1, v_1) \odot \mathcal{E}(r_2, v_2) &= \langle g^{r_1}, h^{r_1} f^{v_1} \rangle \odot \langle g^{r_2}, h^{r_2} f^{v_2} \rangle \\ &= \langle g^{r_1} g^{r_2}, h^{r_1} h^{r_2} f^{v_1} f^{v_2} \rangle \\ &= \langle g^{r_1+r_2}, h^{r_1+r_2} f^{v_1+v_2} \rangle \\ &= \mathcal{E}(r_1 + r_2, v_1 + v_2),\end{aligned}$$

which is the encryption of $v_1 + v_2$. □

Distribution of Trust

- ADDER uses a multi-authority system as described in [CGS97].
- Two sets of users (not necessarily disjoint): *voters* and *authorities*.
 - Secret key is shared among the authorities.
 - A threshold number of authorities must be met in order to decrypt the final result.

Ballot-stuffing

Let us look at a simple yes/no election scheme to see a hazard.

Example

Voter	Choice
V_1	0 (no)
V_2	0 (no)
V_3	1 (yes)
V_4	0 (no)
V_5	4 (???)

V_5 has an bogus ballot, but this will never be known, as it is encrypted. When tabulation occurs, there will be 5 'yes' votes, and 0 'no' votes!

Zero-knowledge Proofs

- We use zero-knowledge proofs to prevent ballot-stuffing.
- Each voter submits a proof that his vote is one of the given options.
- If a proof is rejected, the vote is refused.
- Authorities also use zero-knowledge proofs during the construction of the election secret key.

Stages of an ADDER Procedure

- Administrator selects election parameters and creates voting procedure.
- Authorities create public and private keys and publish their public keys.
- Authorities generate polynomials, evaluating them at the IDs of other authorities, and encrypts those values for each authority.
- Each authority combines his values, arriving at a private key.
- Public key derived from the polynomials of each authority is published.
- Voters cast their votes and the results are tabulated.
- Authorities each partially decrypt the encrypted sum.
- Server combines the partial decryptions and publishes the result.

Ideal Technical Goals

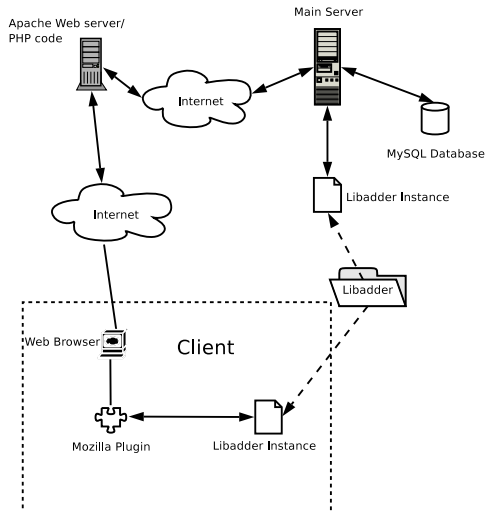
We designed ADDER with several goals in mind.

- Easy-to-use Web interface.
- Cross-platform.
- Must be secure on a standard PC.
- Fault-tolerant database and network.

Components of ADDER

- Cryptographic library.
 - C++ (with GMP) and Java
 - Shared by applet/plugin and main server.
- Java applet/browser plugin.
 - Plugins for Mozilla and IE.
- Main server.
 - Written in C++ (with ACE).
 - Keeps track of running elections.
- Web server.
 - PHP on Apache.
- Database.
 - MySQL.

Diagram



Remaining Problems

- Viruses.
- Authentication.
- Vote-buying and coercion.
- Voter verifiability.
- Denial-of-service attacks.

Further Information

We encourage you to visit the following sites:

- ADDER Web site: <http://www.cse.uconn.edu/~adder/>
- ADDER demo:
<https://lab215-05.engr.uconn.edu/adder/>

Emails may be sent to adder@cse.uconn.edu.