

Paillier/ZKP

Ballot encryption, ZKP and weighted tallying in the Paillier cryptosystem

Reto Bürki, Adrian-Ken Rügsegger
University of Applied Sciences Rapperswil, Switzerland

6/11/2012
Master seminar: E-Voting

Outline

1 Introduction

- System Architecture
- Our Modules

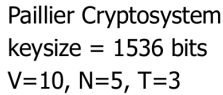
2 Theoretical background

- ZKP verification
- Weighted tallying

3 Implementation

- Overview
- Source
- Demo

1



Our Modules (I)

Module 2

- Ballot encryption
- Zero-knowledge proof for ballot of voter v
- Input
 - Public key generated by module 1
 - Vote instruction (Candidate choice)
- Output
 - Encrypted vote (ballot)
 - ZKP for vote (a, e, z)
 - Election & voter id

Our Modules (II)

Module 3

- Zero-knowledge proof check of ballot
- Weighted tallying
- Input
 - Encrypted vote (ballot)
 - ZKP for vote (a , e , z)
 - Election & voter id
 - Voter registry (shares per voter)
- Output
 - Encrypted weighted tally (ct)
 - Election id

ZKP verification (I)

Prover must show that all u_k 's are n^{th} powers

$$c = g^{m_i} \cdot r^n \bmod n^2 \quad (1)$$

$$u_k = c \cdot (g^{m_k})^{-1} \bmod n^2 \quad (2)$$

- k number of candidates
- i selected candidate
- m_k valid voting messages
- u_k bulletin board values

- Only possible for $k = i$
- Without disclosing random r !
- \rightarrow Use a_k, e_k and z_k arrays to prove it

ZKP verification (II)

Sum of all e_k 's must be equal to challenge e

$$e_k = \begin{cases} e - \sum_{k \neq i} e_k \bmod 2^b & k = i \\ e_k & k \neq i \end{cases} \quad (3)$$

$$\sum e_k \equiv e \bmod 2^b \quad (4)$$

$$b = \frac{\text{size}_2(n)}{2} \quad (5)$$

- b size of challenge in bits (768)
- e challenge (hashed voter/election data & commitment)
- e_k response array e

ZKP verification (III)

All z_k 's must be n^{th} powers

$$a_k = \begin{cases} a_i = z_i^n \bmod n^2 & k = i \\ a_k = z_k^n \cdot (u_k^{e_k})^{-1} \bmod n^2 & k \neq i \end{cases} \quad (6)$$

$$z_k = \begin{cases} z_i = z_i \cdot r^{e_i} \bmod n & k = i \\ z_k & k \neq i \end{cases} \quad (7)$$

$$z_k^n \equiv a_k \cdot u_k^{e_k} \bmod n^2 \quad (8)$$

a_k commitment

e_k response array e

z_k response array z

Weighted tallying

Encrypted tally is product of all encrypted votes modulo n^2

$$ct = \prod_{i=1}^{N_v} c_i^w \bmod n^2 \quad (9)$$

cw weighted encrypted vote
 w weight (number of shares)
 ct encrypted tally
 N_v number of voters

Additive homomorphic properties

- $D(E(m_1) \cdot E(m_2) \bmod n^2) = m_1 + m_2 \bmod n$
- $D(E(m)^k \bmod n^2) = k \cdot m \bmod n$

Overview

Project information

- Programming Language: Ada
- Methodology: Test driven development (TDD)
- Coverage:
 - `-fptest-coverage -fprofile-arcs -fprofile-generate`
 - `lcov`
 - `genhtml`
- Dependencies:
 - Ahven (Unit test library)
 - GMP (Bignum)
 - GNATCOLL (JSON)

Project source

- Source code is freely available
- Opensource license: GPLv3+
- <http://git.codelabs.ch/?p=paillier-zkp.git>
- `git clone http://git.codelabs.ch/git/paillier-zkp.git`

Demo

Talk is cheap. Show me the code.
- *Linus Torvalds*

Questions

Thank you for your attention!