On E-Vote Integrity in the Case of Malicious Voter Computers

Sven Heiberg Helger Lipmaa Filip Van Laenen

Cybernetica AS, Estonia

Computas AS, Norway

September 21, 2010

Outline I

Motivation

- Internet voting:
  - Everbody uses their own PCs to participate in state/local/. . . elections
- Accessibility++
- Cost++
- Security?
  - Voting servers can be protected by organizational means and standard cryptography
  - Voter PCs become the new security bottleneck

On E-Voting Security

- Objectives:
  - Correctness/integrity/robustness:
    - every vote counts (once and correctly)
  - Privacy:
    - Not known how anyone votes
- Adversaries:
  - Voting servers
  - Internet
  - This presentation: voter's PC

Practical Motivation

- We competed in a tender to organize nationwide Internet voting in Norway
- The client wanted to achieve security against malicious voter PCs
  - *under reasonable usability assumptions*
- We showed that it is possible

Privacy against Malicious Voter PC

- Original goal of our client
- Difficult to achieve without hurting usability
- For example, code voting:
  - To vote, voter enters long random code, and to verify correctness, verifies another code
  - For *real* Internet voting, too cumbersome, and too reliant on everyone getting the codes
  - Usability is important!

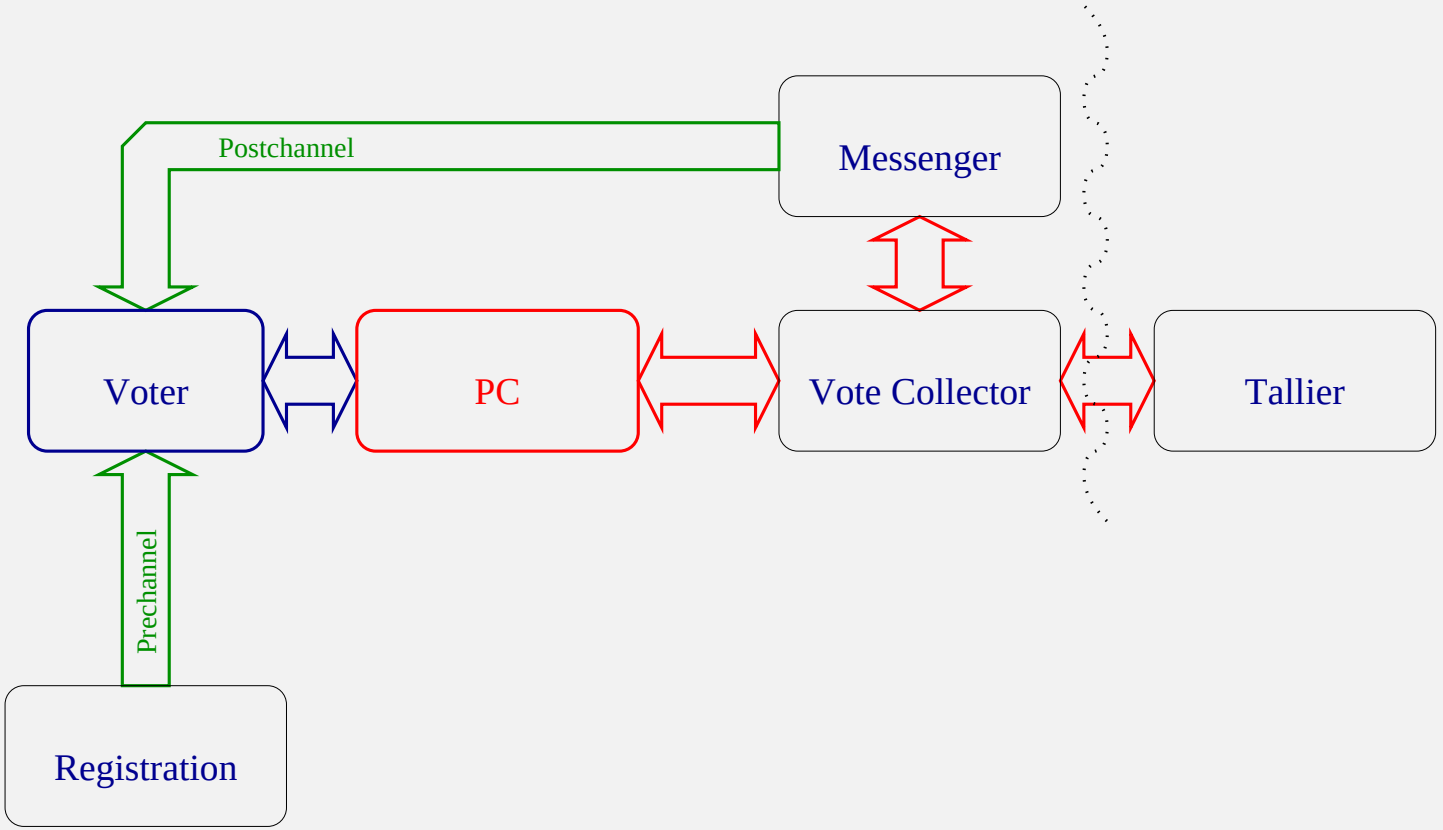
Integrity with Malicious PC

- Voters will be alerted on whether what they voted for reached the voting servers even in the presence of a malicious voter PC
- Without changing user experience much
- Trust model: threshold model is bad (independency of servers?)
- Goal #4: Efficiency?
- (Further adventures of the e-vote can be secured by using standard cryptographic means)

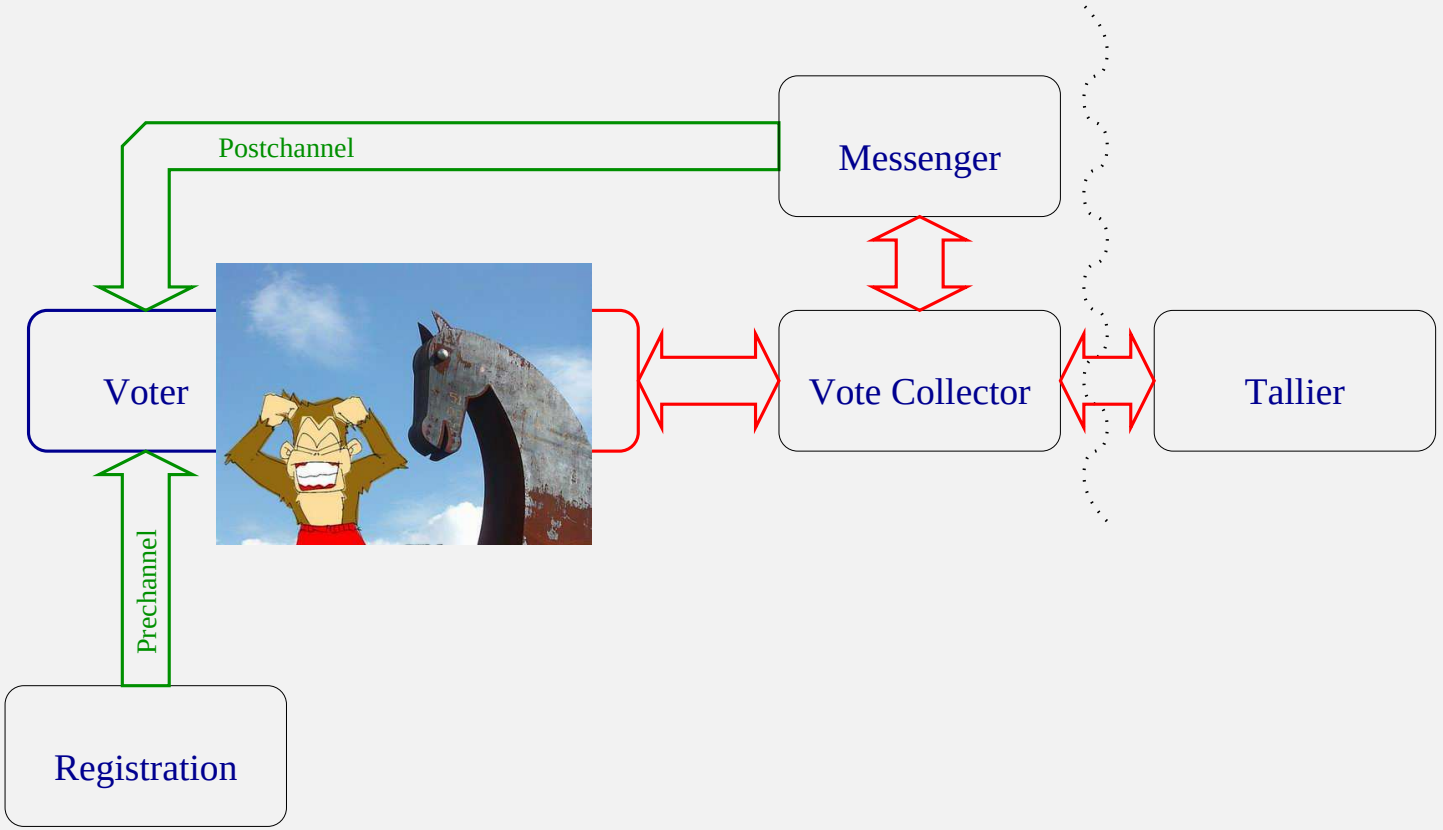
Integrity with Malicious PC

- We need two extra channels to the voter
  - Both must be independent of PC and trusted
  - Independence is really needed since one can revote several times — PC could memorize check codes corresponding to earlier votes
  - Possible coercion/family voting is the main reason implementation of e-voting has been delayed in several countries
- Channels are easy to implement
  - At least in Norway

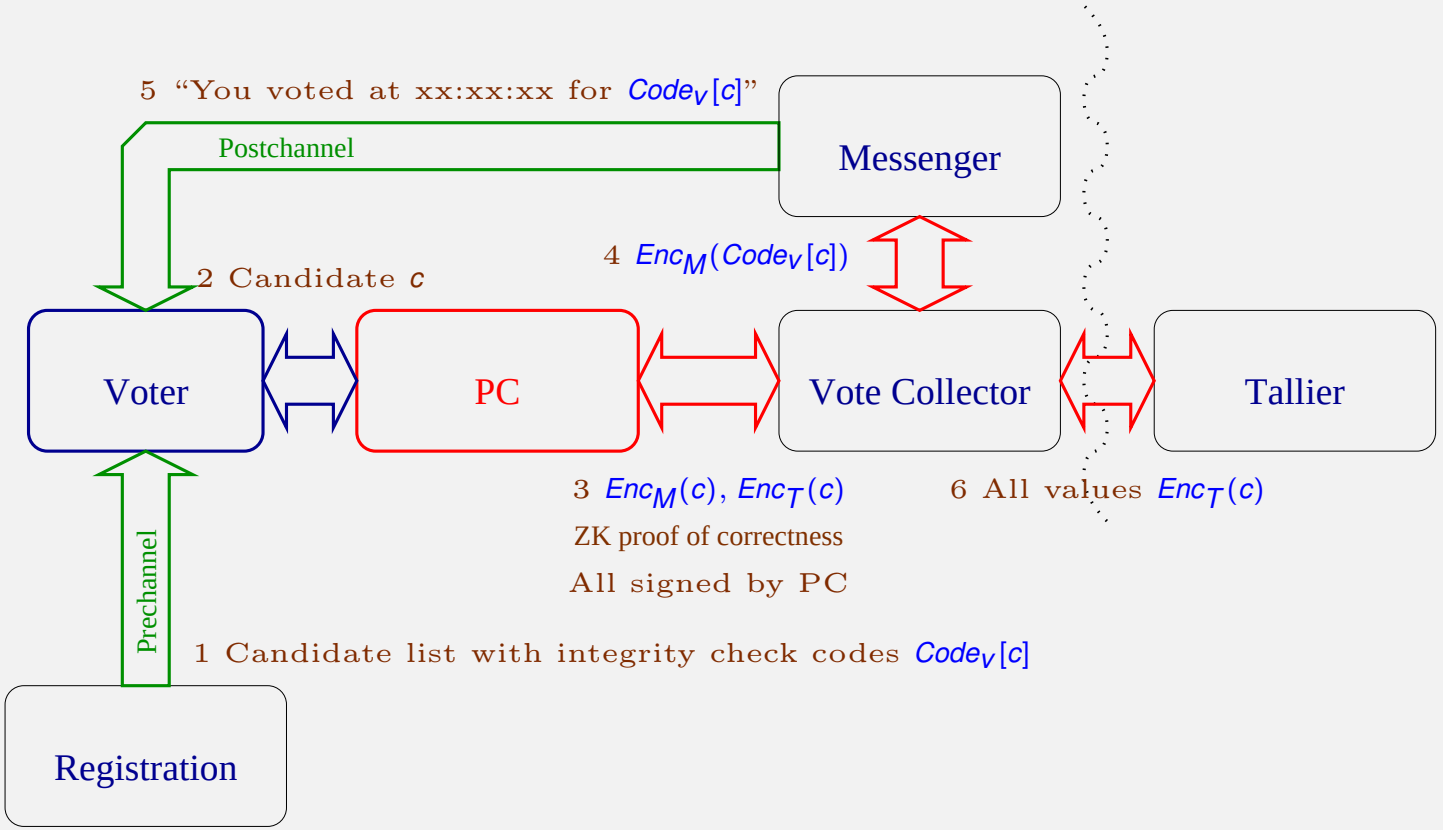
E-voting Process



E-voting Process — Reality



Basic Idea


5 "You voted at xx:xx:xx for $Code_V[c]$"
Postchannel
Messenger
4 $Enc_M(Code_V[c])$
2 Candidate c
Voter
PC
Vote Collector
Tallier
3 $Enc_M(c)$, $Enc_T(c)$
ZK proof of correctness
All signed by PC
6 All values $Enc_T(c)$
Prechannel
1 Candidate list with integrity check codes $Code_V[c]$
Registration

Assumptions behind Our Solution

- Statewise PKI for signing/verification keys
  - check, going to be implemented in parallel
  - . . . although latest news are not so positive anymore . . .
- Minimal PKI to distribute the public encryption keys of voting servers
  - check, easy to implement if you have signing/verification keys

Assumptions behind Our Solution

- Prechannel to distribute check codes to voters
  - (mostly) check, all Norwegians get a voter registration notification on paper anyways
- Extra server (messenger) to notify noters of the success of their actions— check, one extra computer is cheap
- Postchannel between messenger and voters— (mostly) check, can use SMS etc
- Efficient, easily understandable cryptography— ???

Cryptographic Protocol: In A Nutshell

- PC sends $Enc_M(c)$ to vote collector, vote collector applies proxy oblivious transfer to obtain $Enc_M(Code_v[c])$
  - Fairly simple, but costly to implement — VC has to do $2 \cdot \sharp candidates$ exponentiations
- PC proves correctness of its actions
  - ZK proof that $Enc_M(c)$ and $Enc_T(c)$ "encrypt" to the same valid candidate c
  - ZK proof looks complex but is in fact much more efficient than POT

Proxy Oblivious Transfer: Definition

- Chooser has an index $x \in \{0, \ldots, n-1\}$, sender has a database $f = (f_0, \ldots, f_{n-1})$
- Functionality: Proxy obtains f_x
- Privacy: chooser gets no new information, sender obtains nothing about x, proxy only obtains f_x (and no x)!
- In our case, f is the list of codes, x is the concrete candidate, proxy obtains $f_x = Code_v[c]$

Current Status

- We have a mock-up implementation
  - Sandbox (unoptimized) implementation ready
  - One vote collector processes ≈ 3000 votes per hour at 80 candidates
    - In recent Estonian elections, there were ≈ 4500 e-votes in the peak hour (usually much less)
- Considered step: implementation by using a Hardware Security Module — $10+$ times speedup

Current Status

- Norwegian government's representative at NordSec 2009 in Oslo was using slides inspired by our solution
  - Prechannel, postchannel, ...
  - The setting is going to be used
- The final Norwegian protocol is faster but not as secure

Questions?

- Full version at

  `http://eprint.iacr.org/2010/195`
- Further work: we do have more efficient yet secure solutions (not published yet)
  - $> 50\,000$ votes per hour