Ballot Casting Assurance

Rolf Haenni

http://e-voting.bfh.ch

Seminar, E-Voting Group, BFH

October 20th, 2011

Outline

Outline

Introduction

The Belanoh Approach

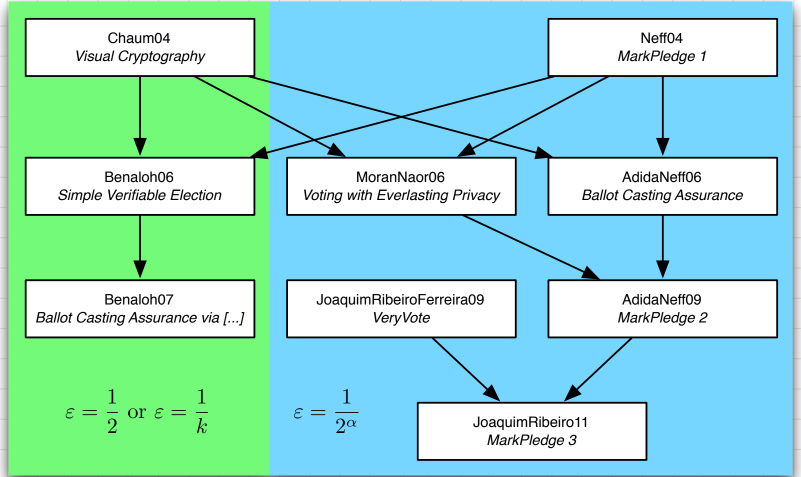
MarkPledge

Outlook

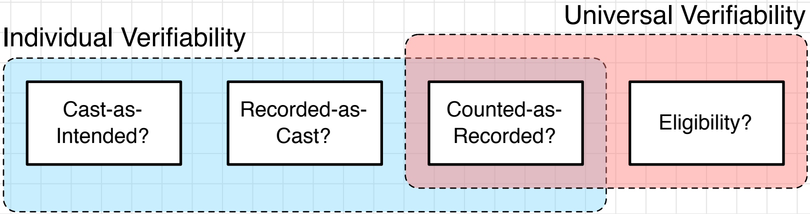
Motivation

- The literature on verifiable elections is mainly focused on the counted-as-cast portion of verifiability
- Protocols usually start with voters encrypting their votes
  - → Voters rely on machines performing the encryption
  - → These machines are assumed to be trustworthy
  - → In the real-world, these machines are very untrustworthy
  - → Secrecy and integrity of vote is at risk
- A few papers are focused on the cast-as-intended portion of verifiability (for voting booths)
  - → Machine provides ZKPs of correct encryption to voters
  - → ZKP is non-transferable (to prevent coercion)
  - → Mechanism is independent of actual tallying procedure

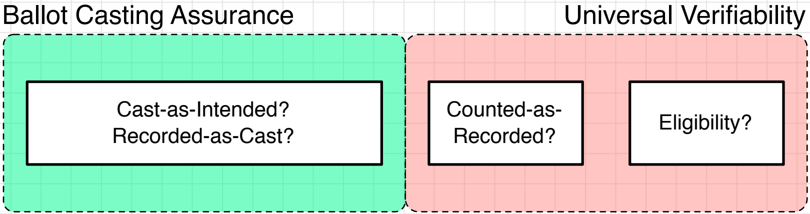
Overview


Chaum04
Visual Cryptography
Neff04
MarkPledge 1
Benaloh06
Simple Verifiable Election
MoranNaor06
Voting with Everlasting Privacy
AdidaNeff06
Ballot Casting Assurance
Benaloh07
Ballot Casting Assurance via [...]
JoaquimRibeiroFerreira09
VeryVote
AdidaNeff09
MarkPledge 2
$\varepsilon = \frac{1}{2}$ or $\varepsilon = \frac{1}{k}$
$\varepsilon = \frac{1}{2^\alpha}$
JoaquimRibeiro11
MarkPledge 3

Individual & Universal Verifiability



Ballot Casting Assurance



General Assumptions

- Public bulletin board: append-only, reliable
- Tallying procedure: privacy-preserving, universally verifiable
- Isolated voting booth:
  - → protects privacy of voter
  - → no side-channel attacks (e.g. no cameras allowed in booth)
- Ballot encryption device:
  - → untrusted for correctness
  - → trusted for not perform subliminal channel attacks on secrecy
- Helper organizations: at least one honest and running correct software
- Voters are potential adversaries (e.g. willing to sell vote)

Outline

Ballot Encryption Device

- Ballot creation and ballot casting is strictly separated
- Ballot encryption device (BED) generates encrypted ballots
  - → can have a rich user interface
  - → needs not to know the identity of voters using the device
  - → needs not to know whether users have the right to vote
  - → needs not limit voters to a single use
  - → needs not record the encrypted ballots it creates
  - → needs not to have remote communication abilities
  - → needs not to be involved in the casting of the ballots
- A variety of media could be used to hand the encrypted ballot over to the voter
- Does not guarantee privacy (e.g. side-channel attacks)

Voter-Initiated Auditing

- General idea: verify ballots which are not cast
  - → after creating the encrypted ballot, voters have the option to immediately decrypt it
  - → if the voter selects this option, the BED provides additional data (e.g. encryption randomness) to allow the decryption
  - → voters can later check that the encrypted ballot matches their intention (helper organization, own trusted software, ...)
- Every uncast ballot multiplies the "undetected cheating probability" by $\frac{1}{k}$ (for k candidates)
  - → even if only 1% of the voters verify an uncast ballot, a cheating BED will be detected with very high probability
  - → ballots to challenge need to be chosen at random
- This method requires virtually nothing more of voters than that to which they are already accustomed

Voter-Initiated Auditing: Setup

- BED must be capable of
  - → receiving and reading ballot-type card (BTC)
  - → encrypting ballot with the election public key
  - → signing the encrypted ballot (one signature key per device)
  - → writing signed encrypted ballots onto BTC
  - → printing short receipts (and printing partial receipts without the voter being able to see what has been printed)
- Poll workers have
  - → a supply of (blank) BTCs
  - → appliances capable of reading from and writing onto BTCs
  - → for each appliance, a counter which is incremented with each use

Voter-Initiated Auditing: Detailed Process I

1. Voter arrives at polling site and is identified by poll worker
2. Poll worker prepares an appropriate BTC containing
   - → information about ballot type
   - → current counter value
3. Voter proceeds to BED and inserts BTC
4. Voter interacts with BED to select the candidate of choice
5. The voter's selection is encrypted and a cryptographic hash thereof is printed onto the paper receipt (invisible to voter)
6. The voter is asked whether this vote should be cast
   - → If YES, the hash of the encrypted ballot and the counter value are signed, and the signature is written onto the BTC and printed onto the receipt

Voter-Initiated Auditing: Detailed Process II

- → If NO, the raw encrypted vote and the encryption randomness is printed onto the receipt

7. The voter removes the BTC from the BED and returns it to the poll worker
8. If the voter has chosen YES and decides to cast the ballot, the poll worker
   - → verifies the signature on the BCT
   - → checks if the counter value matches
   - → records the encrypted ballot as corresponding to the voter and posts it on a public bulletin board
9. The voter takes the receipt home for further verification

Receipts

VOTING RECEIPT

HASH: 4fjk547h

COUNTER: 34

SIGNATURE:
3hj8fjkfk5lfd90kfkr4949034

from cast vote

VOTING RECEIPT

HASH: 4fjk547h

VOTE:
je4jld9lm3kj5j5030fj90fju9fj
kj38uddkdkfi4985ufjfdof94f

RANDOMNESS:
hjdfk34kfk973gujkejg9uie5f

from uncast vote

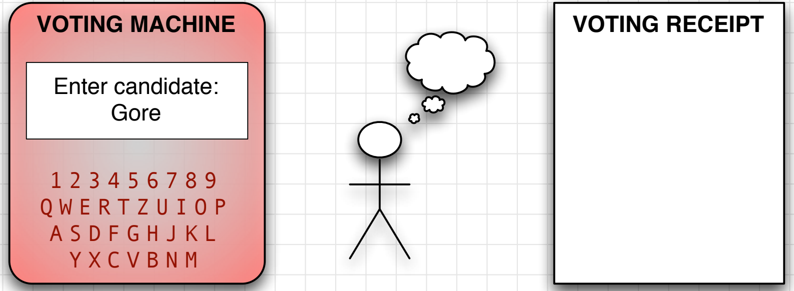
Outline

Assumption and Goals

- Voters cannot be expected to do complicated math
  - → compare short strings (e.g. 4 chars)
  - → compare two icons
- The goal is to achieve optimal correctness according to this computational ability
  - → the best cheating strategy is to guess the short string
  - → $p(\text{correct encryption}|\text{string matches}) = 1 - \frac{1}{35^4} \approx 1 - \frac{1}{10^6}$
- Proof is non-transferable
  - → voter keeps one of the two strings in memory

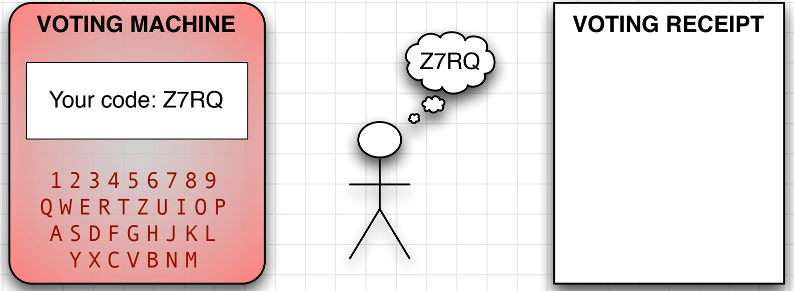
Simplified Proof Procedure

1. Voter enters candidate of choice



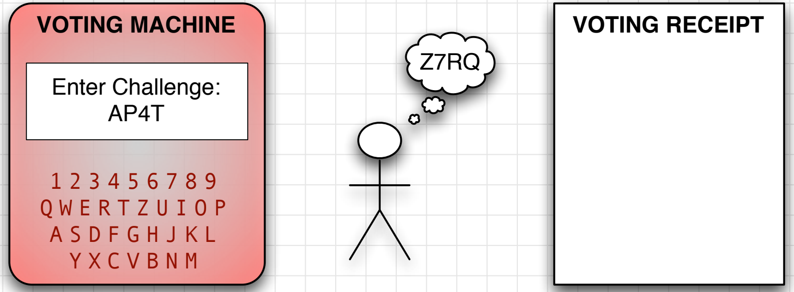
Simplified Proof Procedure

1. Voter enters candidate of choice
2. BED displays a short string to voter (pledge, commitment)



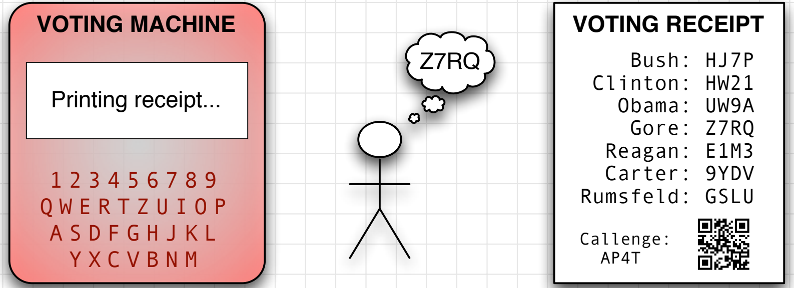
Simplified Proof Procedure

1. Voter enters candidate of choice
2. BED displays a short string to voter (pledge, commitment)
3. Voter enters random string (challenge)



Simplified Proof Procedure

1. Voter enters candidate of choice
2. BED displays a short string to voter (pledge, commitment)
3. Voter enters random string (challenge)
4. BED prints receipt, voter verifies short string



Special Bit Encryption

- Exponential ElGamal: $Enc_y(m, r) = (g^r, g^m \cdot y^r)$
- Bit encryption: $BitEnc_y(b) = \{(u_1, v_1), \ldots, (u_\alpha, v_\alpha)\}$, s.t.
  - → $\forall i : Dec_x(u_i) \oplus Dec_x(v_i) = 1 - b$
- In other words:
  - → $b = 1$ implies that each pair encodes $(0, 0)$ or $(1, 1)$
  - → $b = 0$ implies that each pair encodes $(0, 1)$ or $(1, 0)$
- Let $\mathcal{C} = \{1, \ldots, k\}$ be the candidate slate and $j \in \mathcal{C}$ the voter's candidate choice
- BED computes $c_1, \ldots, c_k$ ($2k\alpha$ ext. ElGamal encryptions)
  - → $c_j = BitEnc_y(1)$
  - → $c_i = BitEnc_y(0)$, for $i \neq j$

Proof Protocol

- Only for the case $b_j = 1$ (the voter's candidate choice)
  - → Protocol input: $c_j = BitEnc_y(b_j)$
  - → Goal of proof: show with soundness $1 - \frac{1}{2^\alpha}$ that $b_j = 1$
- Proof procedure:
  1. Prover sends *commit* of length α to voter, where i-th bit of *commit* corresponds to bit encoded in $(u_i, v_i) \in c_j$
  2. Verifier sends random bit string *chal* of length α to prover
  3. For each i, prover reveals the randomness for u_i if $chal_i = 0$ or for v_i if $chal_i = 1$ (this reveals an α-bit string $response_j$)
  4. Verifier checks that $response_j$ matches *commit*
- Properties of proof:
  - → Proof is clearly complete
  - → Soundness of proof follows from randomness of *chal*
  - → Zero-knowledge follows from straightforward simulation

Voting Process

1. Voter enters candidate of choice $j \in \{1, \ldots, k\}$
2. BED computes k special bit encryptions $c_i = BitEnc_y(b_i)$
   - → For simplicity, we assume that $c_1, \ldots, c_k$ is well-formed
   - → In other words, $b_i \in \{0, 1\}$, $\sum b_i = 1$
3. BED displays *commit*
4. Voter enters *chal*
5. BED completes proof that $b_j = 1$ and generates simulated transcripts that $b_i = 1$, $i \neq j$, using the same challenge *chal*
6. BED prints receipt containing
   - → the voter's challenge *chal*
   - → the list of candidates i together with responses $response_i$
   - → other machine-readable data (encrypted votes, transcripts)
7. Voter checks that *chal* is correct and that $response_j \stackrel{?}{=} commit$

Ballot Casting Assurance

- To achieve ballot casting assurance, some additional measures must be added to guarantee that
  - → the encrypted ballot is well-formed
  - → the voter's challenge is chosen at random
  - → the encrypted ballot reaches the public board
  - → the encrypted ballot and the proof transcripts are internally consistent
- Possible measures:
  - → printer with partial shield prints commitment (see Benaloh)
  - → receipt is digitally signed by BED
  - → receipt is handed over to helper organization(s) for internal consistency checks and correct posting
  - → possibility of late revoting (in case of complaints)
  - → existence of independent verification software

MarkPledge 2

- MarkPledge 2 is similar to MarkPledge 1 (same general strategy)
- Underlying bit encryption scheme is more efficient
- Plaintexts are elements of $SO(2, q)$, the special orthogonal group of 2-by-2 matrices with elements in $\mathbb{Z}_q$
- Ballots are much shorter
- Math is much more complicated ...

MarkPledge 3

- MarkPledge 3 is similar to MarkPledge 1/2 (same general strategy)
- Bit encryption is different: soundness $= 1 - \frac{2}{q}$
  - → $c = BitEnc_y(b) = (u, v)$, $b \in \{1, -1\}$
  - → $u = Enc_y(b, r)$
  - → $v = Enc_y(commit, s)$, $commit = \mathbb{Z}_q$
- Proof is different:
  1. Prover send $u, v, commit$
  2. Verifier sends $chal \in \mathbb{Z}_q$ to prover
  3. Provers sends $R = (b{\cdot}c - c + commit){\cdot}b^{-1}$, $S = r(c - R) + s$ to verifier (note that $R \in \{commit, 2c - commit\}$)
  4. Verifier checks $commit \stackrel{?}{=} R$
  5. Voters sends $u, v, chal, R, S$ to helper organization
  6. Helper organization checks $u^{c-R}{\cdot}v \stackrel{?}{=} (g^S, g^c{\cdot}y^S) = Enc_y(c, S)$

Comparison

- Vote encryption times for a ballot with 10 candidates
- Results according to Joaquim and Ribeiro (2011)
- p and q are ElGamal parameters

	MP1	MP2	MP3
Number of ciphertexts	480	20	20
JavaCard (|p|=1024, |q|=512)	8.5 min	15 hours	1.5 min
MULTOS card (|p|=1024, |q|=512)	5 min	30 min	43 sec
MULTOS card (|p|=1024, |q|=160)	4 min	2.8 min	28 sec

Outline

Can we use these techniques for Internet voting?

Can we apply these techniques to our “Wahlgerät”?