VoteBox
a tamper-evident, verifiable voting system

DANIEL R. SANDLER
KYLE DERR
DAN S. WALLACH
RICE UNIVERSITY

USENIX SECURITY '08
AUGUST 1, 2008
electronic voting research results
DRE: widely deployed
DRE: deeply flawed
DRE: on the way out...?
voters prefer electronic voting

Greene et al. Is newer always better? The usability of electronic voting machines versus traditional methods. CHI ’08.
legitimate benefits
accessibility
feedback
flexibility
(satisfaction)
can we make a better DRE?
“better” = ?
goals

minimized software stack
less code to audit → more practical software audits

resistance to failure & tampering
prevent or minimize data loss

tamper-evidence
if resistance is futile

verifiability
cast-as-intended; counted-as-cast
talk outline

the problem with DREs
software independence

the design of VoteBox
trustworthiness, reliability, tamper-evidence, verifiability

implementation notes
on writing evil code

conclusion
software independence
software independence

briefly,

an undetected system problem cannot create an undetectable change in the results

how?

paper—directly inspect the ballot before casting

electronic—?

current DREs fail this test miserably
toward software independence for DREs
techniques
reduce the trusted computing base
keep believable audit logs
cryptography
non-technique:

“logic & accuracy testing”
goals
for the VoteBox project

minimized software stack
resistance to failure & tampering
tamper-evidence
verifiability

DRE user experience
techniques used in VoteBox

1. PRUI: pre-rendered user interfaces
   DRE user experience; minimized software stack

2. AUDITORIUM: network layer
   resistance to failure; tamper-evidence

3. immediate ballot challenge
   verifiability
PRUI pre-rendered user interfaces

very restricted graphics API

\[
\text{blit(bitmap, } x, y) \\
\text{next\_event()} \to \text{keyboard or } (x, y) \text{ input}
\]

what's not here?

windowing system; widgets; fonts & text rendering

inspiration: Pvote

pioneering work on PRUI in e-voting

(Yee, EVT '06 & '07)
To make your choice, click on the candidate's name or on the box next to his/her name. A green checkmark will appear next to your choice. If you want to change your choice, just click on a different candidate or box.

<table>
<thead>
<tr>
<th>President and Vice President of the United States (You may vote for one)</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ Gordon Bearce            Nathan Maclean  REP</td>
</tr>
<tr>
<td>□ Vernon Stanley Albury   Richard Rigby      DEM</td>
</tr>
<tr>
<td>✓ Janette Froman           Chris Aponte       LIB</td>
</tr>
</tbody>
</table>
To make your choice, click on the candidate's name or on the box next to his/her name. A green checkmark will appear next to your choice. If you want to change your choice, click on a different candidate or box.

President and Vice President of the United States
(You may vote for one)

- Gordon Bearce
  Republican
- Vernon Stanley Albury
  Democratic
- Janet Froman
  Libertarian

Click to go back to instructions
Click to go forward to next race
VoteBox ballot creator

...where the pre-rendering happens
even honest voting machines fail!

we can’t trust voting machines with critical election data

at least, not without redundancy
the AUDITORIUM polling place network joins all voting machines together all election events are signed and broadcast each broadcast is logged by every machine
hash chains

key ingredient in AUDITORIUM
every signed broadcast includes SHA(earlier events)
we can now reason about our audit logs
provable ordering & completeness of the record
...crucial in the voting context

query the log at runtime or offline

Sandler et al. Finding the evidence in tamper-evident logs. SADFE '08.
cryptography

recall our verifiability goal
cast as intended, counted as cast

ballot discussion

1. structure
2. encryption scheme
3. tallying
4. challenge procedure
cast ballot representation

a list of counters, one per candidate

e.g., for one race with three candidates:

\[ \text{ballot} = (a, b, c) \quad a, b, c \in \{ 0, 1 \} \]

ballots may therefore be summed

\[ \text{tally} = \sum \text{ballot}_i = (\sum a_i, \sum b_i, \sum c_i) \]
encryption

ballots should be encrypted

...of course!

preserving the secrecy of the ballot from the voter to the tabulator
Tabulating encrypted ballots

Two basic approaches

Mixnets

Randomize the order of ballots before decrypting & summing

Homomorphic encryption

Sum without decrypting individual ballots

\[ E(x) \odot E(y) = E(x + y) \]

...this is what we use
Additively homomorphic El Gamal

\[
E(c, r, g^a) = \langle g^r, (g^a)^r f^c \rangle \\
D(\langle g^r, g^{ar} f^c \rangle, a) = \frac{g^{ar} f^c}{(g^r)^a} \\
D(\langle g^r, g^{ar} f^c \rangle, r) = \frac{g^{ar} f^c}{(g^a)^r}
\]

\[
\begin{align*}
E(c, r, g^a) &= \langle g^r, (g^a)^r f^c \rangle \\
D(\langle g^r, g^{ar} f^c \rangle, a) &= \frac{g^{ar} f^c}{(g^r)^a} \\
D(\langle g^r, g^{ar} f^c \rangle, r) &= \frac{g^{ar} f^c}{(g^a)^r}
\end{align*}
\]

- \(f, g\): group generators
- \(c\): plaintext (counter)
- \(r\): random (chosen at encryption time)
- \(a\): (private) encryption key
- \(g^a\): (public) encryption key
“cast as intended”

the biggest challenge for DREs

how can the voter be sure the computer:
  captured the voter’s choices faithfully,
  encrypted the ballot correctly,
  and broadcast it in the Auditorium?

unlike “counted as cast,” no amount of procedure
or post facto auditing can correct this
ballot challenge

a technique due to Benaloh (EVT ’07)

at the end of the voting session:

1. force the machine to commit to the ballot it is about to cast

2. the voter chooses to cast the ballot or challenge the machine to reveal its commitment
ballot challenge

voter makes selections

voting machine commits publicly to voter’s choices

voter’s choice

- cast the ballot
- challenge
  - reveal commitment
  - spoil ballot
in Benaloh’s proposal, this is a printed ballot behind an opaque shield

the computer cannot “un-print” the ballot

in VoteBox, this is an Auditorium broadcast message
how do we challenge?

an El Gamal encrypted counter can be decrypted with the private decryption key, or the random value \( r \) supplied at encryption time

NB: we typically throw \( r \) away

challenge: “prove you encrypted the ballot faithfully”

to meet this challenge, VoteBox must reveal \( r \)
(by broadcasting it in AUDITORIUM)
polling place

challenge center

internet

tap

uploader

“data diode”

internet device

observers

challenger

voter

commitments & challenge responses

challenge verification results

INTERNET

5
implementation notes
writing evil code

human factors research

VoteBox’s other customers: CHIL @ Rice

HF studies require behavior that should never be in a real voting machine

*data collection*

copious records of what the voter selected and when

*malice*

actually altering or omitting the voter’s own choices (!)
evil containment

original solution

two branches

functions named “EVIL”—grep the source

current implementation

code preprocessing—compile the evil in or out

#ifdef EVIL (custom preprocessor)
other anecdotes in the paper
conclusion
why?
lots of research on individual pieces of the e-voting problem
VoteBox integrates these techniques in a single system trustworthy reliable tamper-evident verifiable
thanks

students who have worked on VoteBox
Emily Fortuna, George Mastrogiannis, Kevin Montrose, Corey Shaw, Ted Torous
designers of the VoteBox ballot
Mike Byrne, Sarah Everett, Kristen Greene
others who have offered ideas and criticism
Ben Adida, Josh Benaloh, Peter Neumann, Chris Piekert, Brent Waters

NSF/ACCURATE
votebox.cs.rice.edu

(coming soon)