Cryptography Lecture 25
Electronic Voting

Naveen Sastry (guest lecturer; nks@eecs) and Leon Barrett (scribe; lbarrett@eecs)

April 18, 2006

1 Direct Recording Electronic (DRE) Voting Machines

There are advantages to having entirely electronic voting machines, also known as Direct Recording Electronic voting machines (DREs), as anyone familiar with "hanging chads" can tell you. However, there are also disadvantages, as current electronic machines offer few guarantees and require much trust in the manufacturers. We examine the guarantees needed to make a DRE useful in practice and also a scheme designed to make these guarantees.

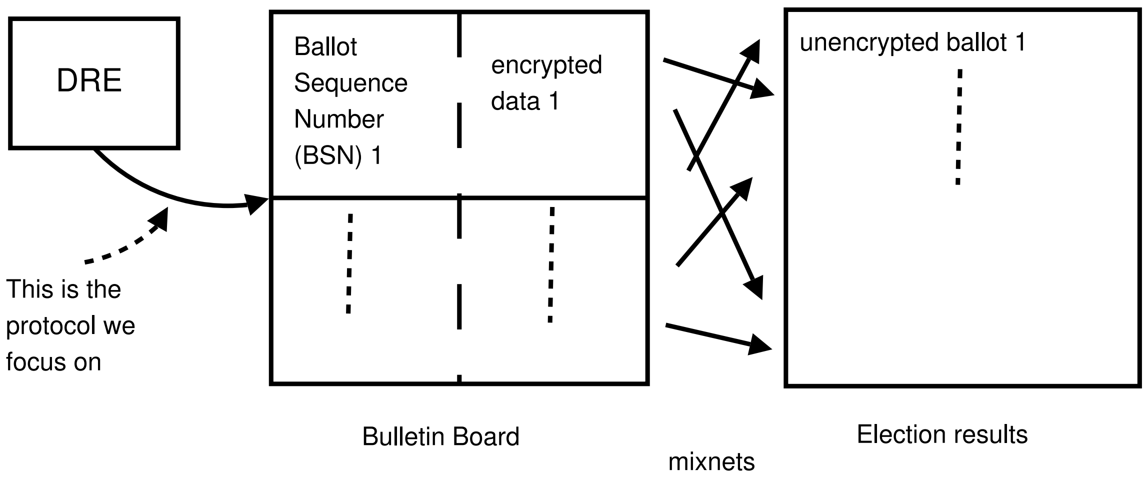

Figure 1: An electronic voting scheme

In general, a voting system would look like Figure 1. It contains a DRE, a bulletin board, and a mixnet. The mixnets can prove, via Zero Knowledge protocols, that they did not alter the ballots, and the presence of multiple mixnets makes it harder to link decrypted ballots with encrypted ones, since even a single honest mixnet operator will result in randomized ballots. Also, the decryption authority can give a ZK proof that it decrypted the resulting ballots correctly, so the only weak link is the DRE. Hence, that is the part of the scheme on which we will focus.

2 Properties

There are many properties we need of a voting scheme.

1. Cast as intended: The voter's ballot should be cast as s/he intended.
2. Counted as cast: The voter's ballot should be counted as it was cast.
3. Verifiability of 1 and 2.
4. One voter, one vote: No voter should be able to vote multiple times.
5. Coercion resustance: The voter should not be able to prove to someone else how s/he voted.
6. Privacy: No one should know, without the voter's help, how s/he voted. This is a weaker form of 5.

3 A Voting Scheme

In this section, we examine C. Andrew Neff's "Vote Here" voting scheme. It has three components: the protocol, the DRE's memory itself, and a receipt the voter gets for verification. The protocol is as follows.

3.1 Step 1

Let us suppose that the voter is selecting one item out of a list to vote for. First, the voter makes a selection, say, item i. They input this into the DRE.

3.2 Step 2

Now the DRE constructs the "VC," or verified choice. This is a set of pairs of bits, stored in memory, as in Figure 2. There is one row of bits for each choice, Ci. Each row contains l pairs of bits, where l is the security parameter (in this example, l=4). Every row other than i has its pairs of bits different, with each pair randomly chosen. Row i, the selected row, has all its pairs the same, again randomly chosen. For our example, we presume that there are 3 choices, and the voter has chosen option 3.

We represent encrypted bits as enclosed by squares and unencrypted bits as enclosed by circles. Unencrypted bits actually hold the encrypted information, plus the randomness used to encrypt them, so we can verify their values. (At this stage, there are no circles.)

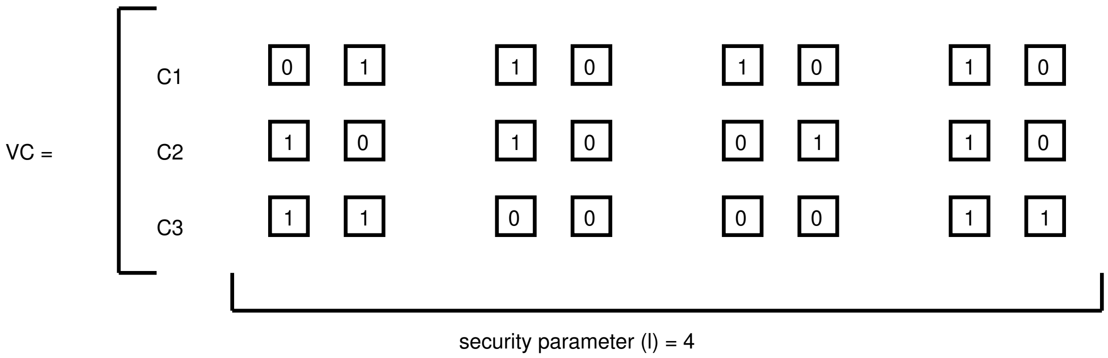

Figure 2: The VC, or Verified Choice

3.3 Step 3

The DRE begins printing the receipt, shown in Figure 3, outputting first the Ballot Sequence Number (BSN) and the hash of the VC. Thus, it has committed to the vote as it is currently stored.

3.4 Step 4

The voter makes a set of challenges $\forall j \neq i$. These are challenges c_j to reveal one bit of each pair. If c_{jk}=0, then the left bit is chosen; if $c_{jk} = 1$ then the right bit is selected. The voting machine then constructs a set of pledges, $p_j = x_j \oplus c_j$, where x_j is the set of left bits from row j of the VC. It also constructs a pledge $p_i = x_i$ for the chosen row.

For our example, we select

$$c_1 = LLRL$$
$$c_2 = RRLL$$

giving us pledges of

$$p_1 = 0101$$
$$p_2 = 0001$$
$$p_3 = 1001$$

3.5 Step 5

The DRE prints the hash of the pledges $(p_1 \dots p_n)$ onto the receipt, thus committing to these pledges.

3.6 Step 6

The voter issues the DRE a challenge for row i, c_i. For our case, suppose $c_3 = LLLR$. Because c_i is chosen *after* the pledges have been committed to, we have a ZK proof that with probability at least $1 - 2^{-l}$, row i consists of matched pairs of bits.

3.7 Step 7

The DRE then prints the challenges $(c_1 \dots c_n)$ to the receipt.

3.8 Step 8

The DRE opens the challenged bits by including the randomness used to encrypt them and posts the resulting open verified choice (OVC), shown in Figure 4, to the bulletin board.

3.9 Receipt

In the end, the voter ends up with receipt shown in Figure 3.

4 Verification

Now that everything is public, the voter can verify the details of his or her ballot.

First, it is easy to verify that the VC is unchanged and that the p_j were all correctly committed to. Because the DRE committed to p_i *before* the voter made random challenge c_i, we know with probability

1. BSN
2. hash(VC)
3. hash($p_1 \ldots p_n$)
4. $c_1 \ldots c_n$

Figure 3: The voter's receipt

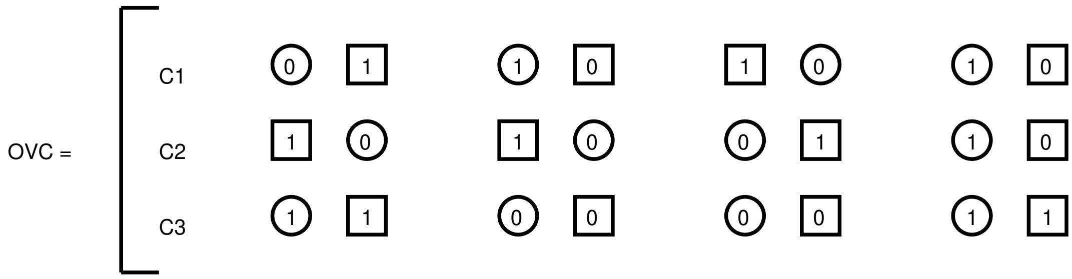

Figure 4: The resulting Open Verified Choice (OVC)

at least $1 - 2^{-l}$ that line i is really all matched pairs of bits. However, as far as I know there is nothing to prevent the machine from setting multiple lines to have paired bits and thereby overvoting. I suppose it could give a ZK proof, but the protocol does not currently include this.

A catch is that if the DRE knows the user's challenges in advance, it can arrange the VC in such a way that the user's vote is incorrect. This means that the voter should have some means of providing real randomness; simply giving his or her birthdate is not sufficient. A coin to toss or die to roll, or even a specially-designed device provided by a trusted source (e.g. ACLU, AARP, GOP, or whatever each voter trusts) could fix this.

Also, there is a rather large loophole in terms of privacy. The machine uses randomness to create the VC, but it need not decide until after the voter has made choice i. Some of this randomness is exposed in the OVC when that is published on the bulletin board. Therefore, if the machine secretly encoded the choice i into the randomness, then it could reveal the voter's choice. How can we stop this? We could verify the random number generator, using some form of trusted hardware or verified software, or even use a PRG with a standard set of keys. A better method is to feed a random tape into the DRE, which can then give a ZK proof that it used the random numbers from the tape. In this way, it can be proven to use random numbers that do not depend on i and thus close the "covert channel" of the exposed random numbers.