Anonymous one-time broadcast using non-interactive dining cryptographer nets with applications to voting

Jeroen van de Graaf

jvdg@ufmg.br

August 2007

Motivation of this research

In 2003, David Chaum proposed a voting protocol with

- Unconditional integrity of the vote count
- Computational privacy of the ballots

His justification: bad people could rig the election, get into power and do away with privacy anyway.

Computational privacy is NOT enough

- Who did Winston Churchill (George Bush) vote for when he was 18?
- After decades of trying a dictator gets elected democratically. He then goes after all people who voted against him (or their sons and daughters).

Reversing the properties is better

A voting protocol with

- Computational integrity of the vote count
- Unconditional (or everlasting) privacy of the ballot

The computational assumption only needs to hold for the **duration** of the election. Once no more appeals are possible, the authorities could make all the secret keys public.

Desirable but apparently impossible properties

A voting protocol with

- Unconditional count integrity
- Unconditional (or everlasting) privacy

This seems very similar to bit commitment, in which the hiding and the biding cannot be both unconditional.
Q: Can we prove something here?

The general idea of the paper

We will modify the Fujioka, Okamoto and Ohta voting protocol:

- Using blind signature techniques, the voter obtaines a signed ballot from the Ballot Authority
- The voter takes out the blinding and submits his vote anonymously, using a **Non-Interactive version** of the Dining Cryptographers protocol

NIDC seems an interesting primitive on its own, perhaps useful for applications other than voting.

Comparison to other work

- Bos (1992) uses interactive DC to implement an election.
- Cramer, Franklin, Schoenmakers, Yung (1996) present a voting protocol with unconditional privacy.
- Moran and Naor (2006, 2007) use non-interactive bit commitment
- Since PunchScan is based on bit commitment, one can use unconditionally hiding BCs.

The remainder of this talk

- NIDC supposing everybody is honest
- Dealing with dishonest people
- Conclusions

The original Dining Cryptographers protocol

- Three cryptographers ($P = 3$)
- Want to know if one of them paid
- Each flips a random coin
- Shows value to left neighbour (R_{ij})
- "My coin and my neighbor's coin are EQUAL/DIFFERENT" (C_i)
- The person who paid, "lies" ($M_i = 1$)
- If the number of people that say "DIFFERENT" is odd, one of them paid; otherwise an outsider paid.

General implementation

- Any number of participants P instead of just 3
- No restriction to neighbors, any graph will do
- Instead of sending only one bit, many bits can be sent
- Participants take turns in sending something
- Collisions can be resolved by resending
- Disrupters can be driven out

Towards a Non-Interactive version

- Use slots whose length is defined as L
- Use S slots, where each slot represents an independent execution of a DC net
- Let each participants choose one slot at random
- Choose S so large that collisions are improbable

Outline of the protocol for Participant i

1. $\mathcal{P}_i$ exchanges random bits R_{ij} with $\mathcal{P}_j$
2. $\mathcal{P}_i$ choose one slot $s_i \in S$ at random and sets the message for each slot:
$$M_i[s] = \begin{cases} V_i & \text{for } s = s_i \\ 0^L & \text{for } s \neq s_i \end{cases}$$
3. $\mathcal{P}_i$ computes his contribution for each slot
$$C_i[s] = M_i[s] \oplus R_{i1} \oplus \cdots \oplus R_{iP}$$
where j ranges over P_i's neighbors

Who occupies which slot

$M_1 =$ 000000000000000000000000001000000000000
$M_2 =$ 002000000000000000000000000000000000000
$M_3 =$ 000000000000000000000000000000000000003

. . . . .

. . . . .

$M_i =$ 000000000000i000000000000000000000000000

. . . . .

. . . . .

$M_P =$ 000000000000000P000000000000000000000000

Output 002000000000i0000P000000000001000000000003

Dealing with dishonest people

No-shower A party that went through the preliminary phase but does not show op.

- Parties would need to recalculate $R_{i1} \oplus \cdots \oplus R_{iP}$
- Alternatively, introduce a small numbers of authorities

Disrupter A party who submitted arbitrary random bits

How to prevent a party from submitting garbage

- Let all parties commit to their random bits R_{ij} and message bits M_i.
- Each bit of the contribution C_i is a linear relation $(\mathrm{mod}\ 2)$ of committed

Needed: an unconditional bit commitment scheme that allows verification of linear relations.

Bit commitment with XOR

$\overrightarrow{x}$ is a vector of ordinary BC pairs that XOR to x

$$\begin{array}{c} \overrightarrow{x} = 1 \\ \overline{0}, \overline{1} \\ \overline{1}, \overline{0} \\ \overline{0}, \overline{1} \\ \overline{0}, \overline{1} \\ \overline{0}, \overline{1} \end{array}$$

How to show equality between to BCXs(1)?

$\mathcal{A}$ $\overrightarrow{x} = 1$	$\mathcal{A}$ $\overrightarrow{y} = 1$
$\overline{0}, \overline{1}$	$\overline{1}, \overline{0}$
$\overline{1}, \overline{0}$	$\overline{1}, \overline{0}$
$\overline{0}, \overline{1}$	$\overline{0}, \overline{1}$
$\overline{0}, \overline{1}$	$\overline{1}, \overline{0}$
$\overline{0}, \overline{1}$	$\overline{0}, \overline{1}$

How to show equality between to BCXs(2)

$\mathcal{A}$ $\overrightarrow{x} = 1$	$\mathcal{A}$ $\overrightarrow{y} = 1$	$\mathcal{A}$
$\overline{0}, \overline{1}$	$\overline{1}, \overline{0}$	$\neq$
$\overline{1}, \overline{0}$	$\overline{1}, \overline{0}$	$=$
$\overline{0}, \overline{1}$	$\overline{0}, \overline{1}$	$=$
$\overline{0}, \overline{1}$	$\overline{1}, \overline{0}$	$\neq$
$\overline{0}, \overline{1}$	$\overline{0}, \overline{1}$	$=$

How to show equality between to BCXs(3)

$\mathcal{A}$ $\overrightarrow{x} = 1$	$\mathcal{A}$ $\overrightarrow{y} = 1$	$\mathcal{A}$	$\mathcal{B}$
$\overline{0}, \overline{1}$	$\overline{1}, \overline{0}$	$\neq$	L
$\overline{1}, \overline{0}$	$\overline{1}, \overline{0}$	$=$	R
$\overline{0}, \overline{1}$	$\overline{0}, \overline{1}$	$=$	L
$\overline{0}, \overline{1}$	$\overline{1}, \overline{0}$	$\neq$	L
$\overline{0}, \overline{1}$	$\overline{0}, \overline{1}$	$=$	R

How to show equality between to BCXs(4)

$\mathcal{A}$ $\overrightarrow{x} = 1$	$\mathcal{A}$ $\overrightarrow{y} = 1$	$\mathcal{A}$	$\mathcal{B}$
$\mathbf{0}, \overline{1}$	$\mathbf{1}, \overline{0}$	$\neq$	L
$\overline{1}, \mathbf{0}$	$\overline{1}, \mathbf{0}$	$=$	R
$\mathbf{0}, \overline{1}$	$\mathbf{0}, \overline{1}$	$=$	L
$\mathbf{0}, \overline{1}$	$\mathbf{1}, \overline{0}$	$\neq$	L
$\overline{0}, \mathbf{1}$	$\overline{0}, \mathbf{1}$	$=$	R

Switching notation

$\mathcal{A}$ $\overrightarrow{x} = 1$	$\mathcal{A}$ $\overrightarrow{y} = 1$	$\mathcal{A}$	$\mathcal{B}$
$0, \overline{1}$	$1, \overline{0}$	1	0
$\overline{1}, 0$	$\overline{1}, 0$	0	1
$0, \overline{1}$	$0, \overline{1}$	0	0
$0, \overline{1}$	$1, \overline{0}$	1	0
$\overline{0}, 1$	$\overline{0}, 1$	0	1

Obvious generalizations of BCX

- Inequality: $\overrightarrow{x} \oplus \overrightarrow{y} = 1$
- Linear relations: $\overrightarrow{x_1} \oplus \overrightarrow{x_2} \oplus \cdots \overrightarrow{x_n} = 1$
- (Addition using a different modulus)

Protocol sketch

- $\mathcal{P}_i$ and $\mathcal{P}_j$ create R_{ij} and commit to them towards the other parties.
- $\mathcal{P}_i$ commits to $M_i = 0^L_1 \cdots 0^L_{s_i-1} V_{s_i} 0^L_{s_i+1} \cdots 0^L_S$ and shows to the other parties that all slots are 0^L except 1.
- $\mathcal{P}_i$ computes and publishes C_i and shows for each bit that $\overrightarrow{C_i} = \overrightarrow{M_i} \oplus \overrightarrow{R_{i1}} \oplus \cdots \oplus \overrightarrow{R_{iP}}$

Showing well-formedness of M_i

$\mathcal{A}$ chooses and commits to a permutation σ and creates M_i'.

	M_i		M_i'
1	$\vec{0}$	$\overline{2}$	$\vec{0}$
2	$\vec{0}$	$\overline{4}$	$\vec{V}$
3	$\vec{0}$	$\overline{1}$	$\vec{0}$
4	$\vec{V}$	$\overline{5}$	$\vec{0}$
5	$\vec{0}$	$\overline{3}$	$\vec{0}$

$\mathcal{B}$ flips a coin:

If 0: Open the permutation and show that equality between the BCX hold, obeying σ.

If 1: Open all the BCXs except one on the right; $\mathcal{B}$ can verify that in all the opened positions the value committed to was 0.

How to deal with collisions?

- Increase S, the number of slots.
- Run T NIDC nets in parallel.
- Allow each party to occupy $T > 1$ slots.
- Use an algorithm to recover from a collision
  - For instance, each message could contain the slot numbers of all the T slots there were used.

Conclusion

- NIDC works in principle resulting in unconditional privacy.
- The current version is very inefficient.
  - The BCX causes a large expansion per bit.
  - The collision probability needs to be optimized
- Major problems when applied to voting:
  - Dealing with no-showers.
  - Ballot marking is possible.