Internet Voting Protocols with Everlasting Privacy

Jeroen van de Graaf
Joint work with Denise Demirel e Roberto Samarone

jvdg@dcc.ufmg.br

June 2012

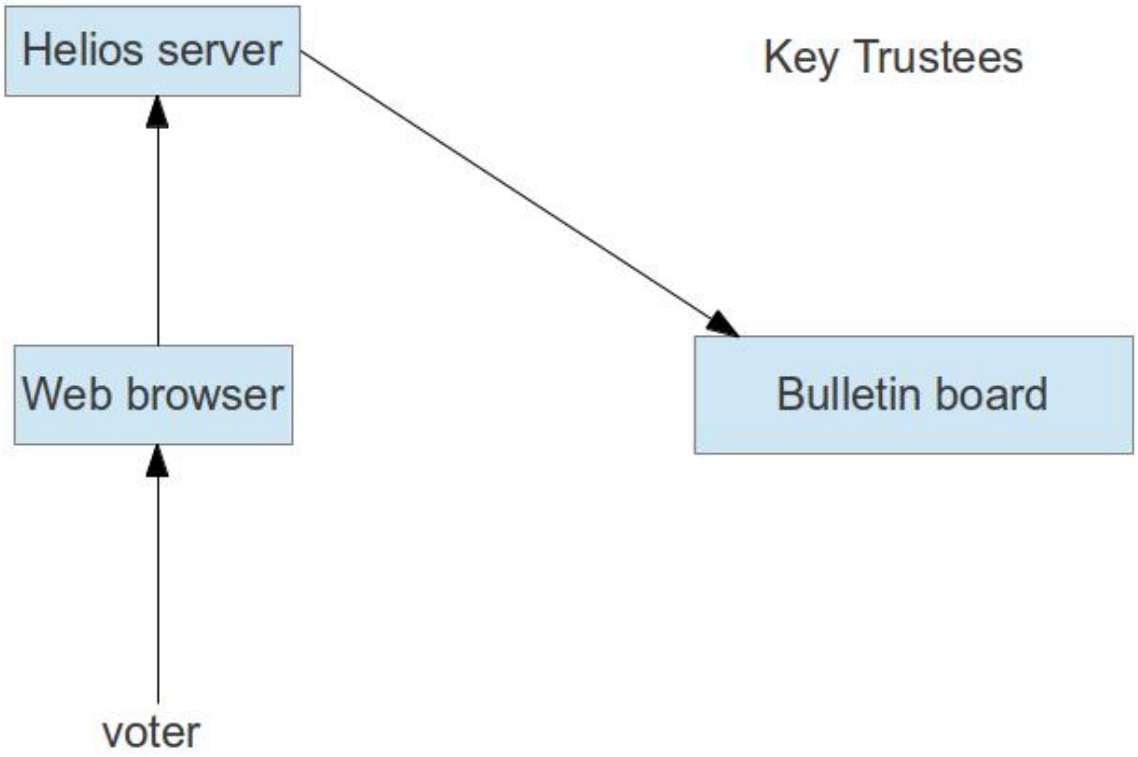
Outline of this talk

(1) A looong introduction to internet voting/Helios
(2) Shortcomings
(3) Our improved protocol

The Helios voting system

- www.heliosvoting.org
- internet voting application
- **not** for official election
- good for department head; IACR board of directors; SBC directors
- developed by Ben Adida, PhD student of Ron Rivest
- you vote using your browser

Components of the system



User perspective

(1) The voter receives user name and election-specific password by email, and a URL

(2) A JavaScript application is downloaded

(3) (a) The voter makes a choice;
(b) her vote is encrypted

(4) The voter can decide to audit the encrypted vote. In this case, the browser opens additional information allowing verification of correct encryption. Then go back to step 1.

(5) (a) The additional information is destroyed;
(b) the user authenticates herself and casts the vote.

(6) The voter receives a confirmation message.

The election web page

Voters	*Cand 1*	*Cand 2*	...	*Cand l*
Voter 1	$u(0)$	$u(1)$	...	$u(0)$
Voter 2	$u(1)$	$u(0)$	...	$u(0)$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
Voter V	$u(0)$	$u(1)$	...	$u(0)$
Total	$u(t_1^*)$	$u(t_2^*)$	...	$u(t_l^*)$

The election web page

Voters	*Cand 1*	*Cand 2*	...	*Cand l*
Voter 1	$u(0)$	$u(1)$	...	$u(0)$
Voter 2	$u(1)$	$u(0)$	...	$u(0)$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
Voter V	$u(0)$	$u(1)$	...	$u(0)$
Total	$u(t_1^*)$	$u(t_2^*)$	...	$u(t_l^*)$

- Counting of the votes is based on homomorphic encryption:

$$u(t_1)u(t_2) = u(t_1 + t_2)$$

- The Helios server, with help of the Key Trustees, decrypts the totals to find the results $t_1^*, t_2^*, \ldots, t_l^*$ where $t_i^* = \sum t_i(j)$

ElGamal encryption

Helios implements Cramer-Gennaro-Schoenmakers:

(1) Alice choose P, α, x and computes $\beta = \alpha^x$ mod P. She publishes P, α, β and keeps x private

(2) Bob sends a message m with a random s as follows:
$E(m, s) = \langle \alpha^s, \beta^s m \rangle = \langle c_1, c_2 \rangle$

(3) Alice decrypts: $m' = c_2(c_1^x)^{-1} = (\beta^s t)(\alpha^s)^{-x} = m$

ElGamal encryption

Helios implements Cramer-Gennaro-Schoenmakers:

(1) Alice choose P, α, x and computes $\beta = \alpha^x$ mod P. She publishes P, α, β and keeps x private

(2) Bob sends a message m with a random s as follows:
$E(m, s) = \langle \alpha^s, \beta^s m \rangle = \langle c_1, c_2 \rangle$

(3) Alice decrypts: $m' = c_2(c_1^x)^{-1} = (\beta^s t)(\alpha^s)^{-x} = m$

(4) ElGamal preserves multiplication:

$$E(m_1, s_1)E(m_2, s_2) = E(m_1 m_2, s_1 s_2)$$

ElGamal encryption

Helios implements Cramer-Gennaro-Schoenmakers:

(1) Alice choose P, α, x and computes $\beta = \alpha^x$ mod P. She publishes P, α, β and keeps x private

(2) Bob sends a message m with a random s as follows:
$E(m, s) = \langle \alpha^s, \beta^s m \rangle = \langle c_1, c_2 \rangle$

(3) Alice decrypts: $m' = c_2(c_1^x)^{-1} = (\beta^s t)(\alpha^s)^{-x} = m$

(4) ElGamal preserves multiplication:

$$E(m_1, s_1)E(m_2, s_2) = E(m_1 m_2, s_1 s_2)$$

(5) Exponential ElGamal preserves addition: choose $m = \delta^t$ then

$$E'(t_1, s_1)E'(t_2, s_2) = E'(t_1 + t_2, s_1 s_2)$$

The election web page

Voters	*Cand 1*	*Cand 2*	...	*Cand l*
Voter 1	$E(t_1(1))$	$E(t_2(1))$	...	$E(t_l(1))$
Voter 2	$E(t_1(2))$	$E(t_2(2))$	...	$E(t_l(2))$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
Voter V	$E(t_1(V))$	$E(t_2(V))$	...	$E(t_l(V))$
TOTAL	$\prod E(t_1(j))$	$\prod E(t_2(j))$	...	$\prod E(t_l(j))$
equals	$E(\sum t_1(j))$	$E(\sum t_2(j))$	...	$E(\sum(t_l(j))$

The election web page

Voters	*Cand 1*	*Cand 2*	...	*Cand l*
Voter 1	$E(t_1(1))$	$E(t_2(1))$	...	$E(t_l(1))$
Voter 2	$E(t_1(2))$	$E(t_2(2))$	...	$E(t_l(2))$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
Voter V	$E(t_1(V))$	$E(t_2(V))$	...	$E(t_l(V))$
TOTAL	$\prod E(t_1(j))$	$\prod E(t_2(j))$	...	$\prod E(t_l(j))$
equals	$E(\sum t_1(j))$	$E(\sum t_2(j))$	...	$E(\sum(t_l(j))$

- Pedersen has a protocol for distributed decryption using a distributed, private ElGamal key

The election web page

Voters	*Cand 1*	*Cand 2*	...	*Cand l*
Voter 1	$E(t_1(1))$	$E(t_2(1))$	...	$E(t_l(1))$
Voter 2	$E(t_1(2))$	$E(t_2(2))$	...	$E(t_l(2))$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
Voter V	$E(t_1(V))$	$E(t_2(V))$	...	$E(t_l(V))$
TOTAL	$\prod E(t_1(j))$	$\prod E(t_2(j))$	...	$\prod E(t_l(j))$
equals	$E(\sum t_1(j))$	$E(\sum t_2(j))$	...	$E(\sum(t_l(j))$

- Pedersen has a protocol for distributed decryption using a distributed, private ElGamal key
- ElGamal decryption results in $m = \delta^{t^*} \bmod p$.

The election web page

Voters	*Cand 1*	*Cand 2*	...	*Cand l*
Voter 1	$E(t_1(1))$	$E(t_2(1))$	...	$E(t_l(1))$
Voter 2	$E(t_1(2))$	$E(t_2(2))$	...	$E(t_l(2))$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
Voter V	$E(t_1(V))$	$E(t_2(V))$	...	$E(t_l(V))$
TOTAL	$\prod E(t_1(j))$	$\prod E(t_2(j))$	...	$\prod E(t_l(j))$
equals	$E(\sum t_1(j))$	$E(\sum t_2(j))$	...	$E(\sum(t_l(j))$

- Pedersen has a protocol for distributed decryption using a distributed, private ElGamal key
- ElGamal decryption results in $m = \delta^{t^*} \bmod p$.
- Finding t^* is called the Discrete Logarithm problem.
- Discrete Log is difficult in general, but here the values are small.

Security properties of Helios

As a result Helios offers

- Individual verifiability
- Universal verifiability
- Unconditional integrity of the vote count
- Computational privacy of the ballots

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).
- Your boss at 47 might have been the president of your student association when you were 22.

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.

The basic idea

- Use Pedersen commitments as an alternative encoding of the votes
- Expressions of the form

$$u(t, s) = \alpha^s \beta^t \in \mathbb{Z}_p^*$$

- Actually first presented in [CDG87]

Properties of this encoding

Homomorphic:

$$u(t_1, s_1)u(t_2, s_2) = \alpha^{s_1}\beta^{t_1}\alpha^{s_2}\beta^{t_2} = \alpha^{s_1+s_2}\beta^{t_1+t_2} = u(t_1 + t_2, s_1 + s_2)$$

Properties of this encoding

Unconditional privacy:

$$u(t, s) = \alpha^s \beta^t \in \mathbb{Z}_p^*$$

Proof: Given u, each possible t is equiprobable provided that both α and β are generators and s is chosen randomly in $\mathbb{Z}_p^*$.

Properties of this encoding

Decrypting (opening) to a different value is impossible provided Discrete Log is hard.
Proof:

$$\alpha^{s_1}\beta^{t_1} = \alpha^{s_2}\beta^{t_2} \iff \alpha^{s_1-s_2} = \beta^{t_2-t_1} \iff \alpha = \beta^{\frac{t_2-t_1}{s_1-s_2}}$$

The election web page

Voters	*Cand 1*	*Cand 2*	...	*Cand l*
Voter 1	$u(t_1(1), s_1(1))$	$u(t_2(1), s_2(1))$	...	$u(t_l(1), s_l(1))$
Voter 2	$u(t_1(2), s_1(2))$	$u(t_2(2), s_2(2))$	...	$u(t_l(2), s_l(2))$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
Voter V	$u(t_1(V), s_1(V))$	$u(t_2(V), s_2(V))$	...	$u(t_l(V), s_l(V))$
TOTAL	$\prod u(t_1(j), s_1(j))$	$\prod u(t_2(j), s_2(j))$	...	$\prod u(t_l(j), s_l(j))$
	u_1^*	u_2^*	...	u_l^*

The election web page

Voters	*Cand 1*	*Cand 2*	...	*Cand l*
Voter 1	$u(t_1(1), s_1(1))$	$u(t_2(1), s_2(1))$	...	$u(t_l(1), s_l(1))$
Voter 2	$u(t_1(2), s_1(2))$	$u(t_2(2), s_2(2))$	...	$u(t_l(2), s_l(2))$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
Voter V	$u(t_1(V), s_1(V))$	$u(t_2(V), s_2(V))$	...	$u(t_l(V), s_l(V))$
TOTAL	$\prod u(t_1(j), s_1(j))$	$\prod u(t_2(j), s_2(j))$	...	$\prod u(t_l(j), s_l(j))$
	u_1^*	u_2^*	...	u_l^*

- We have that $u_1^* = \alpha^{\sum s_1(j)} \beta^{\sum t_1(j)} = \alpha^{s_1^*} \beta^{t_1^*}$

The election web page

Voters	*Cand 1*	*Cand 2*	...	*Cand l*
Voter 1	$u(t_1(1), s_1(1))$	$u(t_2(1), s_2(1))$	...	$u(t_l(1), s_l(1))$
Voter 2	$u(t_1(2), s_1(2))$	$u(t_2(2), s_2(2))$	...	$u(t_l(2), s_l(2))$
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
Voter V	$u(t_1(V), s_1(V))$	$u(t_2(V), s_2(V))$	...	$u(t_l(V), s_l(V))$
TOTAL	$\prod u(t_1(j), s_1(j))$	$\prod u(t_2(j), s_2(j))$	...	$\prod u(t_l(j), s_l(j))$
	u_1^*	u_2^*	...	u_l^*

- We have that $u_1^* = \alpha^{\sum s_1(j)} \beta^{\sum t_1(j)} = \alpha^{s_1^*} \beta^{t_1^*}$
- **Problem:** How to decrypt? We need to recover the s_i^* and t_i^*
- Discrete Log is difficult in general, and here the values are not small.

Enter Paillier encryption

Solution: The values $s_i(j)$ and $t_i(j)$ are sent to the Election Authority over a *private channel* using suitable homomorphic encryption.
We choose to use Paillier encryption, which uses an additional random value:

$$v(s, r) = \gamma^s r^N \bmod N^2$$

$$w(t, r') = \gamma^s (r')^N \bmod N^2$$

Here $N = p_1 p_2$ is the public key.
The primes p_1 and p_2 are the private key.
We will need that $(p_1 - 1)/2$ and $(q_1 - 1)/2$ are prime too.

Encoding of a vote

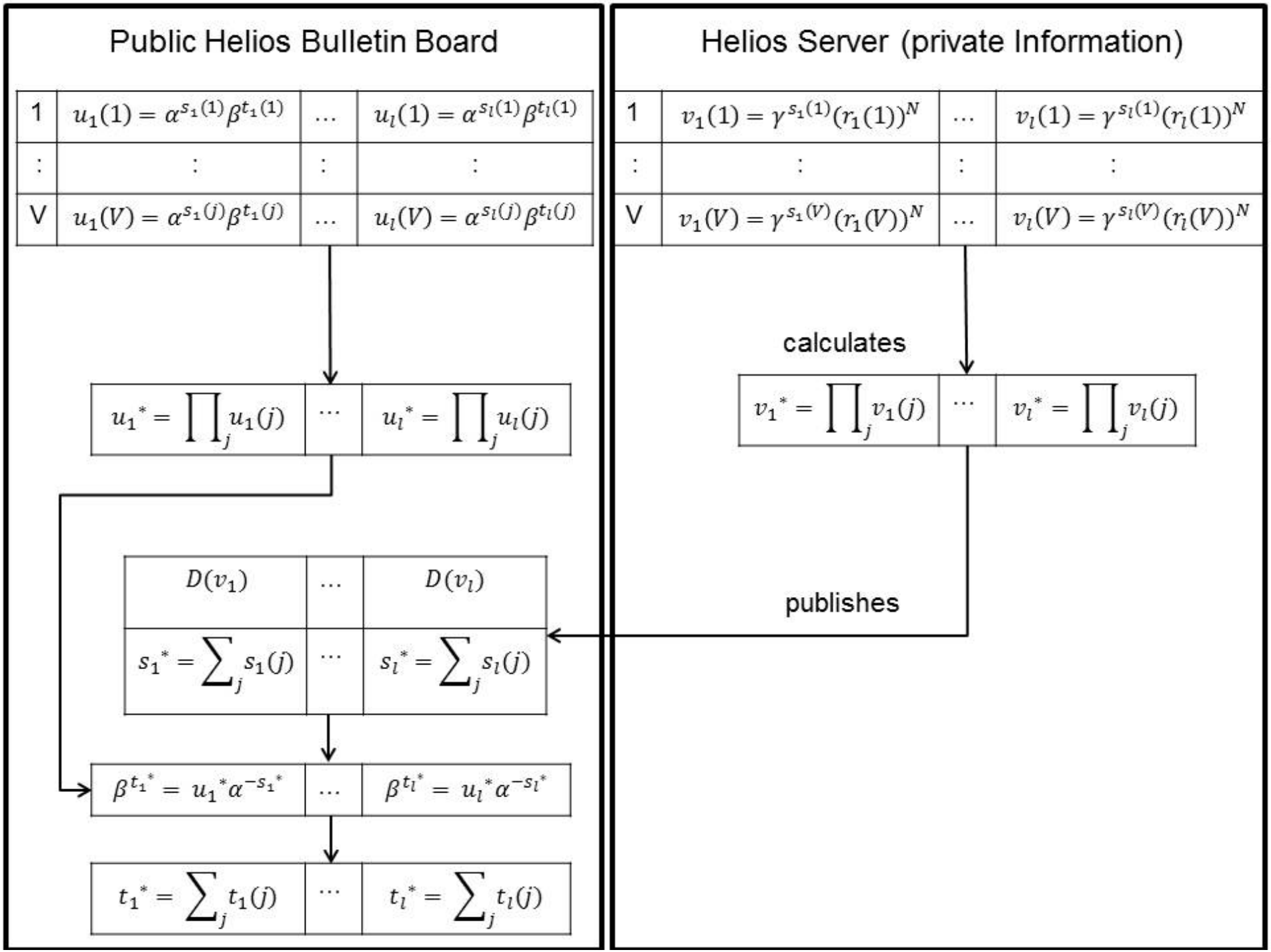
So the encoding of t takes three random values and has three components, one that is public, and two sent privately to the server:

$$Enc(t, s, r, r') = \langle u, v, w \rangle = \langle \alpha^s \beta^t, \gamma^s r^N, \gamma^t (r')^N \rangle$$

By carefully choosing the groups we get

$$\mathrm{Enc}(t_1, s_1, r_1, r_1') * \mathrm{Enc}(t_2, s_2, r_2, r_2') = \mathrm{Enc}(t_1 + t_2, s_1 + s_2, r_1 \cdot r_2, r_1' \cdot r_2')$$

$*$ is componentwise multiplication in $\mathbb{Z}^*_{4N+1} \times \mathbb{Z}^*_{N^2} \times \mathbb{Z}^*_{N^2}$
$+$ is addition in $\mathbb{Z}_N$
$\cdot$ is multiplication in $\mathbb{Z}^*_{N^2}$


Public Helios Bulletin Board
Helios Server (private Information)
1
$u_1(1) = \alpha^{s_1(1)}\beta^{t_1(1)}$
…
$u_l(1) = \alpha^{s_l(1)}\beta^{t_l(1)}$
:
:
:
:
V
$u_1(V) = \alpha^{s_1(j)}\beta^{t_1(j)}$
…
$u_l(V) = \alpha^{s_l(j)}\beta^{t_l(j)}$
1
$v_1(1) = \gamma^{s_1(1)}(r_1(1))^N$
…
$v_l(1) = \gamma^{s_l(1)}(r_l(1))^N$
:
:
:
:
V
$v_1(V) = \gamma^{s_1(V)}(r_1(V))^N$
…
$v_l(V) = \gamma^{s_l(V)}(r_l(V))^N$
calculates
$u_1^* = \prod_j u_1(j)$
…
$u_l^* = \prod_j u_l(j)$
$v_1^* = \prod_j v_1(j)$
…
$v_l^* = \prod_j v_l(j)$
publishes
$D(v_1)$
…
$D(v_l)$
$s_1^* = \sum_j s_1(j)$
…
$s_l^* = \sum_j s_l(j)$
$\beta^{t_1^*} = u_1^* \alpha^{-s_1^*}$
…
$\beta^{t_l^*} = u_l^* \alpha^{-s_l^*}$
$t_1^* = \sum_j t_1(j)$
…
$t_l^* = \sum_j t_l(j)$

Proofs that a vector encoding corresponds to a valid vote

When submitting, it must be proven that the vote vector is correctly formatted:

(1) all values t_i are 0 or 1

(2) $\sum_i t_i = 1$.

(3) The values s_i and t_i must be used consistently, that is, the s_i and t_i used in the unconditional encryption equals the one used in the two homomorphic encryptions.

(1) and (2) needs to be proven publicly, whereas (3) needs to be proven towards the Helios server only.
We discuss (2) before (1), then (3)

(2) $\sum_i t_i = 1$

Recall that $u(t, s) = \alpha^s \beta^t = \alpha^s \beta^1$, so if well-formatted, then

$$\theta(j) := \prod_{i=1}^{l} u_i(j) \beta^{-1} = \alpha^{s^\dagger(j)}$$

where $s^\dagger(j) = \sum_{i=1}^{l} s_i(j)$. So it is enough to show knowledge of a DL of $\theta(j)$ with respect to α.

(2) Proof of knowledge of a Discrete Log

	Alice		**Election Authority**
0	$\theta = \alpha^s$	$\longrightarrow\longrightarrow^{\theta}\longrightarrow\longrightarrow$	
1	$\theta' = \alpha^{s'}$	$\longrightarrow\longrightarrow^{\theta'}\longrightarrow\longrightarrow$	
2		$\longleftarrow\longleftarrow^{c}\longleftarrow\longleftarrow$	c is a random challenge
3	$\theta'' = g^{cs+s'}$	$\longrightarrow\longrightarrow^{\theta''}\longrightarrow\longrightarrow$	$\theta'' \stackrel{?}{=} \theta^c\theta'$

(2) Proof of knowledge of a Discrete Log

	Alice		**Election Authority**
0	$\theta = \alpha^s$	$\longrightarrow\longrightarrow^{\theta}\longrightarrow\longrightarrow$	
1	$\theta' = \alpha^{s'}$	$\longrightarrow\longrightarrow^{\theta'}\longrightarrow\longrightarrow$	
2		$\longleftarrow\longleftarrow^{c}\longleftarrow\longleftarrow$	c is a random challenge
3	$\theta'' = g^{cs+s'}$	$\longrightarrow\longrightarrow^{\theta''}\longrightarrow\longrightarrow$	$\theta'' \stackrel{?}{=} \theta^c\theta'$

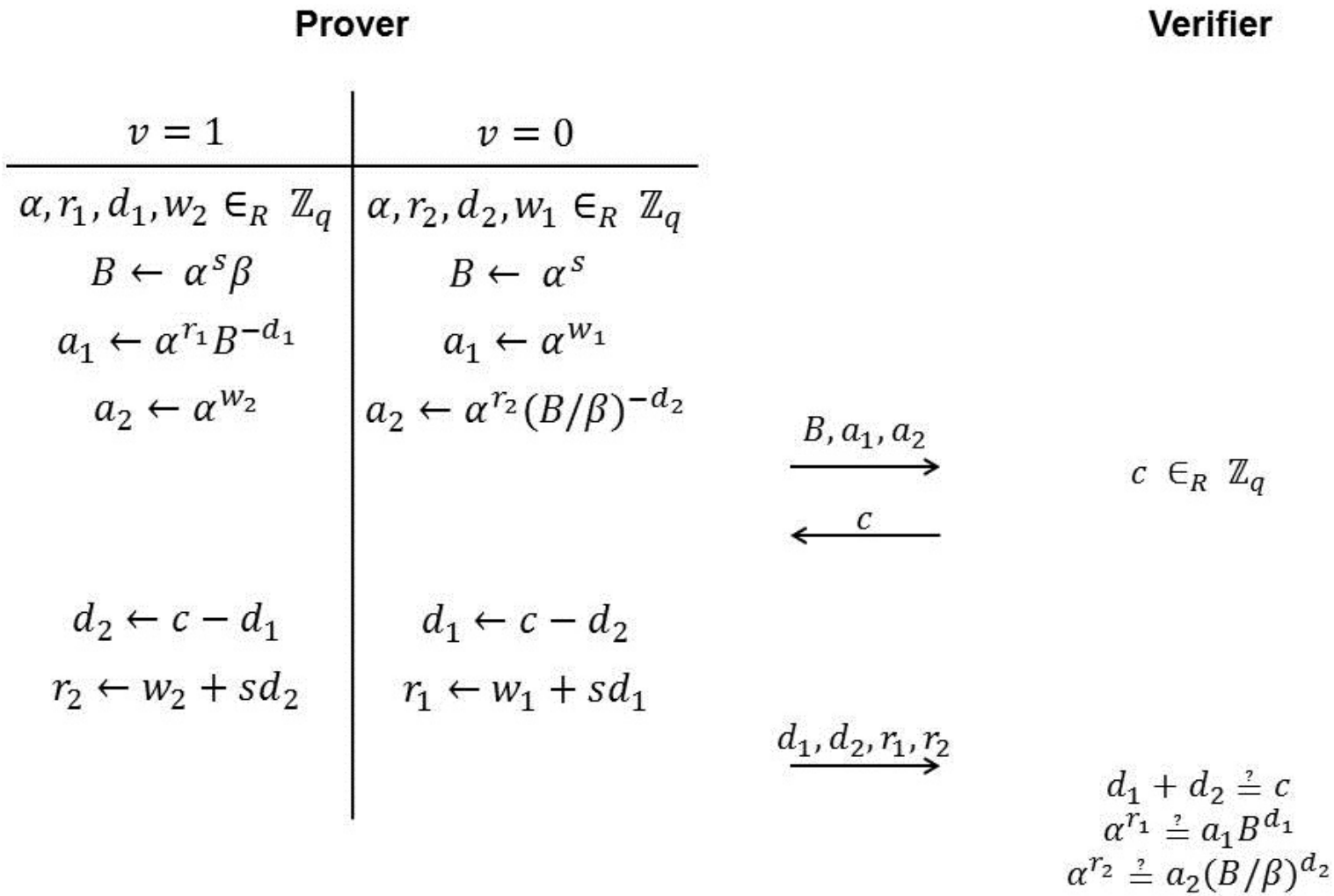
- For $c \in \{0, 1\}$: ZeroKnowledge

(2) Proof of knowledge of a Discrete Log

	Alice		**Election Authority**
0	$\theta = \alpha^s$	$\longrightarrow\longrightarrow^{\theta}\longrightarrow\longrightarrow$	
1	$\theta' = \alpha^{s'}$	$\longrightarrow\longrightarrow^{\theta'}\longrightarrow\longrightarrow$	
2		$\longleftarrow\longleftarrow^{c}\longleftarrow\longleftarrow$	c is a random challenge
3	$\theta'' = g^{cs+s'}$	$\longrightarrow\longrightarrow^{\theta''}\longrightarrow\longrightarrow$	$\theta'' \stackrel{?}{=} \theta^c\theta'$

- For $c \in \{0, 1\}$: ZeroKnowledge
- For $c \in \{1, \ldots, p-1\}$: Schnorr

(1) all values t_i are 0 or 1



Consistent use of s and t

This can be proven using a standard cut-and-choose protocol:

(i) Choose $\overline{s}$ uniformly random and compute $\mu = Enc(t, \overline{s}, r, r')$

(ii) Receive challenge bit

(iii) Either send $\overline{s}$ or send $s + \overline{s}$

(iv) V verifies either whether μ was constructed correctly or whether the u and v components of $Enc(t, s, r, r') * \mu \stackrel{?}{=} Enc(t, s + \overline{s}, r, r')$

Assumptions

We make the following assumptions:

- The Discrete Log problem is hard.
- The Paillier encryption is semantically secure.
- The Key Trustees are not conspiring

Protocol properties

Correctness vote count

The election outcome is correct, provided the discrete log of β with respect to α cannot be computed before the election result is made public. This statement remains true even if the Helios server and the Key Trustees conspire.

Unconditional privacy

For each voter i, the mutual information between the voter's choice, and the public view (receipts, other data on bulletin board) is zero. This statement is true as long as a sufficient number of Key Trustees is honest.

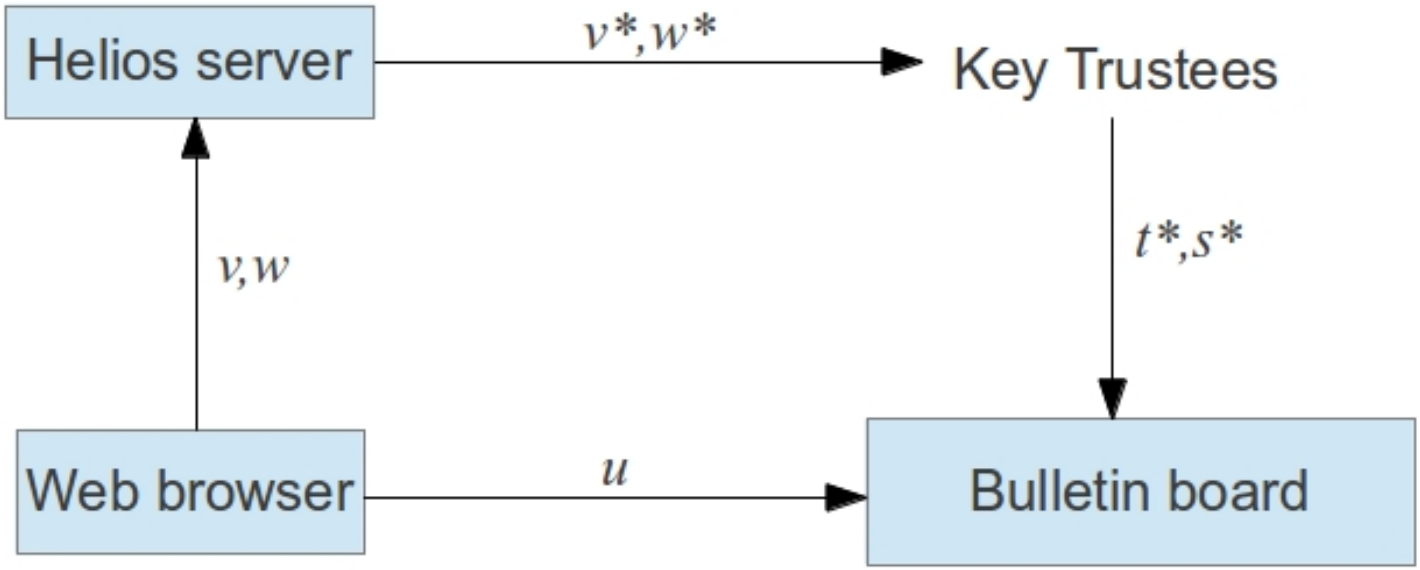
Individual Voter Verifiability

Each voter can verify that his vote is included in the tally.

Universal Verifiability

Any observer can verify that the tally was calculated correctly.

The position of an adversary



Comparison to other work

CGS	Internet voting, computational assumptions.
CFSY	Internet voting, voter needs secret sharing to many authorities.
MoranNaor	Unconditional privacy but not for internet voting; some techniques used.
PAV, PS, Merging	Unconditional privacy but not for internet voting.
NIDC	Internet voting, inefficient BCs, † for voting (at least for now)

Possible generalizations

- The construction is generic, meaning that any voting protocol using homomorphic encryption can be modified
- Similar ideas can be used implement *mix networks* with everlasting privacy to the public.

Acknowledgement

- Thank you Dagstuhl / CASED / TU Darmstadt / JBuchmann !!