Improving Helios with Everlasting Privacy Towards the Public

Denise Demirel, Jeroen van de Graaf, Roberto Araújo

TECHNISCHE UNIVERSITÄT DARMSTADT

helios

Helios Election Server

Helios

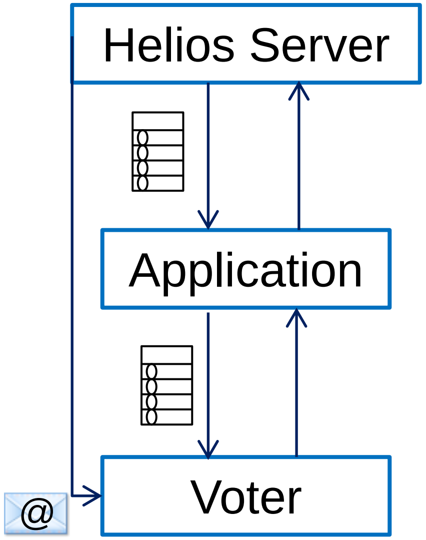
TECHNISCHE UNIVERSITÄT DARMSTADT

- Introduced 2008 by Ben Adida
- Web application for Internet voting
- Online accessible: http://heliosvoting.org
- Easy to use, free of charge, provides end-to-end verifiability
- Tool to support elections for companies, online groups, …
    - President of the Université Catholique de Louvain (2009)
    - Princeton Undergraduate Student Government election (2009)

Helios Election Process (1)

1. System initialization
    a) User creates election by setting parameters and list of eligible voters.
    b) Software generates election templates (e.g. ballot, key pair for threshold decryption).

2. Vote Casting process
    a) Voter receives email containing username, password, URL,...
    b) Single-page JavaScript application starts and downloads parameters and templates.



Helios Election Process (2)

2. Vote Casting process

c) Voter fills out the ballot, which is encrypted by the application.

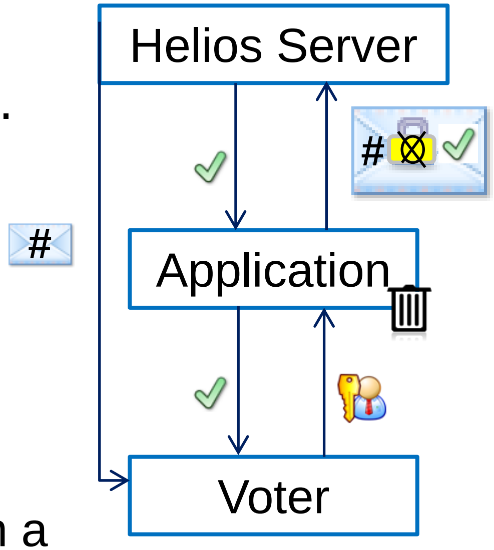
d) Hash of encrypted vote is shown to the voter.

e) The voter has the option to audit.
In this case go back to step 2c).

f) Application clears scope, voter authenticates

g) ID, password, encrypted vote and proofs are sent to the Helios server which responds with a success message.

h) The Helios server sends the voter an email containing the hash of the cast vote.



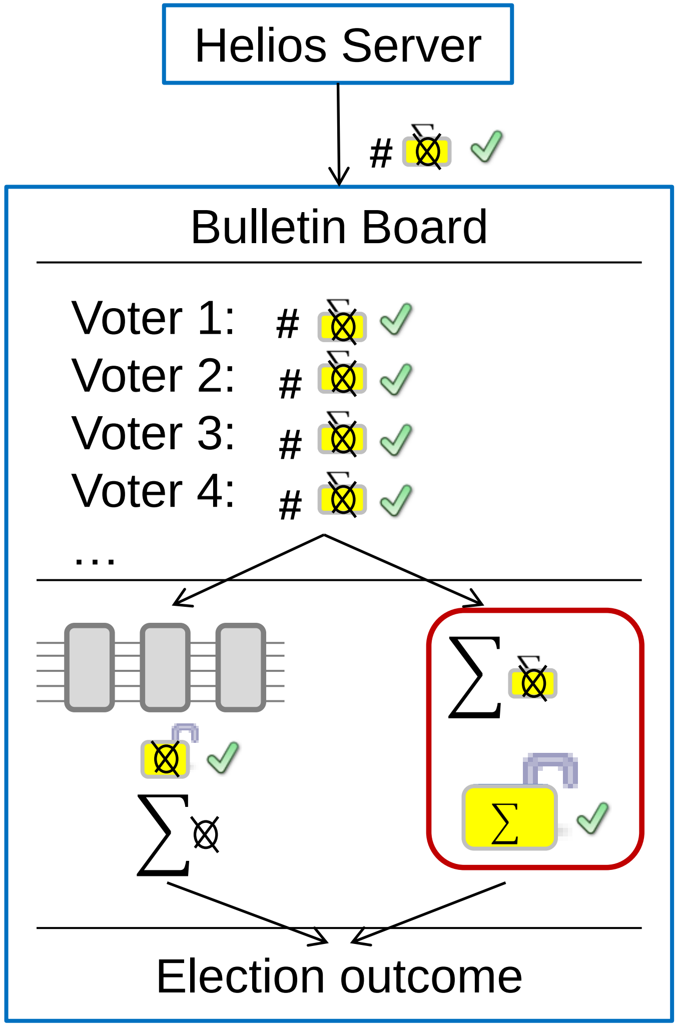
Helios Election Process (3)

3. Tallying and publishing of votes

a) The Helios server publishes the encrypted votes, hashes and proofs on the Bulletin Board.

b) The Helios server computes the election outcome.

- Mixing + decryption + tallying
- Homomorphic tallying + decryption



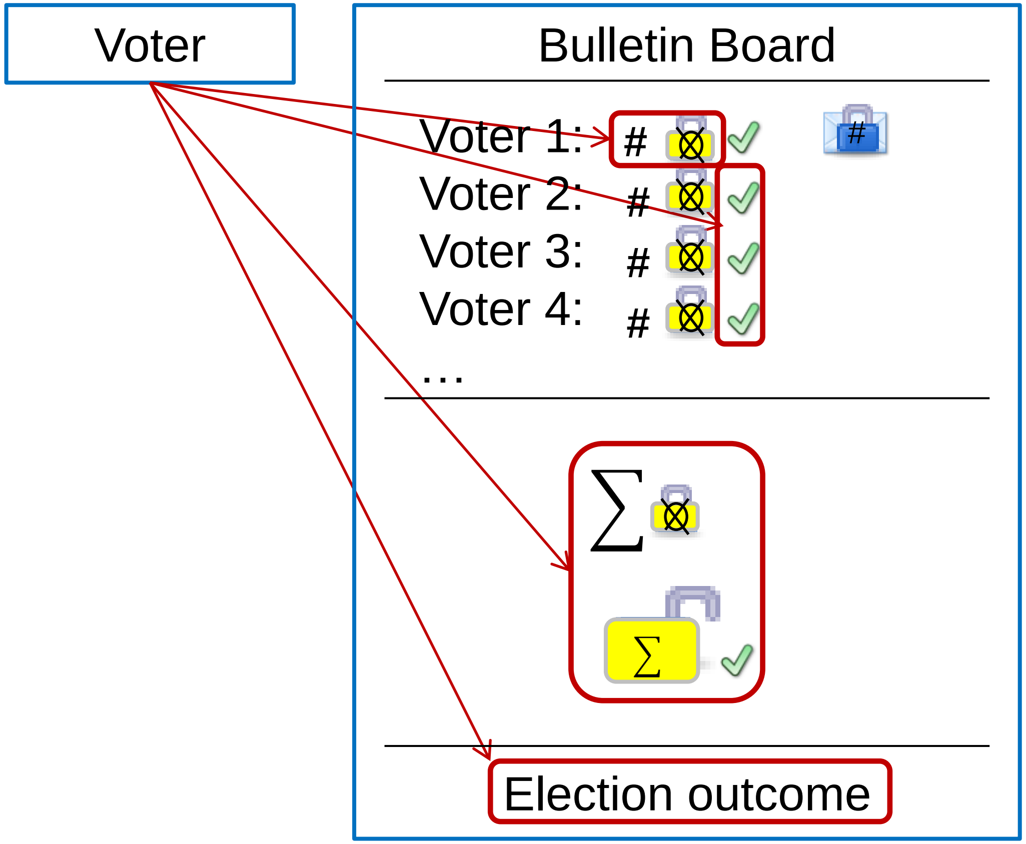
Tallying and Publishing of Votes

Voters	*Candidate 1*	*Candidate 2*	…	*Candidate n*
Voter 1 : Voter V	$Enc(t_1(1))$: $Enc(t_1(V))$	$Enc(t_2(1))$: $Enc(t_2(V))$	… : …	$Enc(t_n(1))$: $Enc(t_n(V))$
Total	$\prod Enc(t_1(j))$	$\prod Enc(t_2(j))$	…	$\prod Enc(t_n(j))$
Equals	$Enc(\sum t_1(j))$	$Enc(\sum t_2(j))$	…	$Enc(\sum t_n(j))$

- Vote is represented as vector $\langle t_1, \ldots, t_l \rangle$.
- Vote for candidate i is vector where $t_i = 1$ and $t_k = 0$ for $k \neq i$.
- Each entry is encrypted individually using exponential ElGamal.
- Votes for each candidate are tallied homomorphically and decrypted.
- Decryption results in δ^{t^*} (need to solve DL, but here values are small).

Properties

- Individual Verifiability
- Universal Verifiability
- Correctness
- Computational Privacy



Computational Privacy

- Homomorphic Public-Key Cryptography e.g. Paillier, ElGamal
- Computational Assumptions
- Brute-Force
- Principle of free suffrage

Helios Providing Everlasting Privacy Towards the Public

Goal: Having all published data (Bulletin Board, receipts), even a computationally unbounded attacker cannot reveal the cast voting decision.

Solution: Encryption of the cast voting decision using a "One-Time-Pad".

Challenge: How can a pair of voting decision and associated key be tallied homomorphically providing Verifiability and Everlasting Privacy regarding the published information?

“Encoding” using Pedersen Commitments

Cyclic group *G*

Random generators *g* and *h* ($\log_g h$ unknown and computationally hard)

Encoding of vote *t* ∈ $\mathbb{Z}_{|G|}$: $C(t,s) = g^s h^t$ with random number *s* ∈ $\mathbb{Z}_{|G|}$.

Properties of Pedersen Commitments

Computational Binding:

Decrypting (opening) to a different value is impossible given that solving Discrete Log is computationally hard.

$$C(t_1, s_1) = g^{s_1}\, h^{t_1} = g^{s_2} h^{t_2} \Leftrightarrow g^{s_1 - s_2} = h^{t_2 - t_1} \Leftrightarrow g = h^{\frac{t_2 - t_1}{s_1 - s_2}}$$

Additive Homomorphic:

$$C(t_1, s_1) * C(t_2, s_2) = g^{s_1} h^{t_1} * g^{s_2} h^{t_2} = C(t_1 + t_2, s_1 + s_2)$$

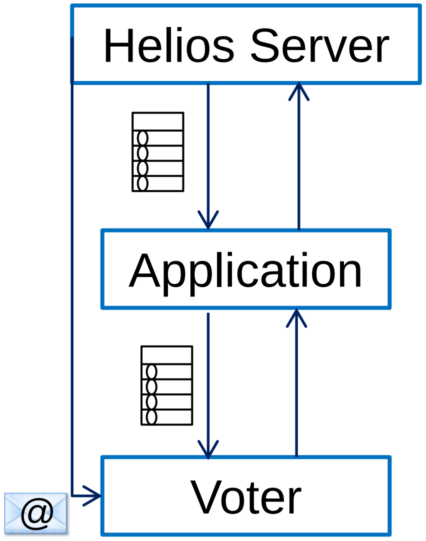
Everlasting Privacy:

The commitment scheme is “unconditionally hiding”.

Modified Election Process (1)

1. System initialization
   a) User creates election by setting parameters and list of eligible voters.
   b) Software generates election templates (e.g. ballot, key pair for threshold decryption, parameters for Pedersen Commitment).

2. Vote Casting process
   a) Voter receives email containing username, password, URL,...
   b) Single-page JavaScript application starts and downloads parameters and templates.



Modified Election Process (2)

2. Vote Casting process

c) Voter marks a choice, which is “encoded” using Pedersen Commitments.

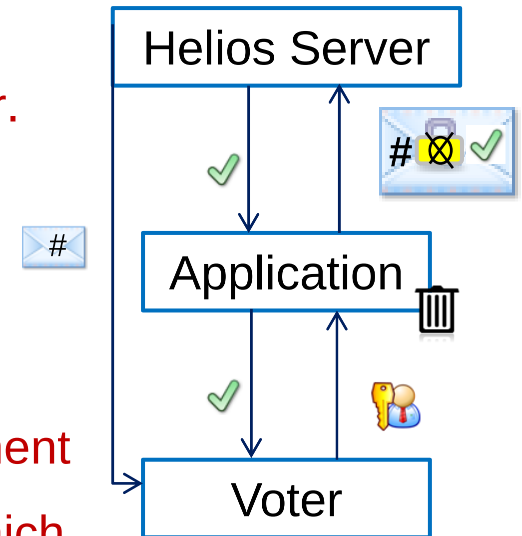
d) The Hash of the commitment is shown to the voter.

e) The voter has the option to audit.
In this case go back to step 2c).

f) Application clears scope, voter authenticates

g) ID, password, commitment, encrypted decommitment values and proofs are sent to the Helios server which responds with a success message.

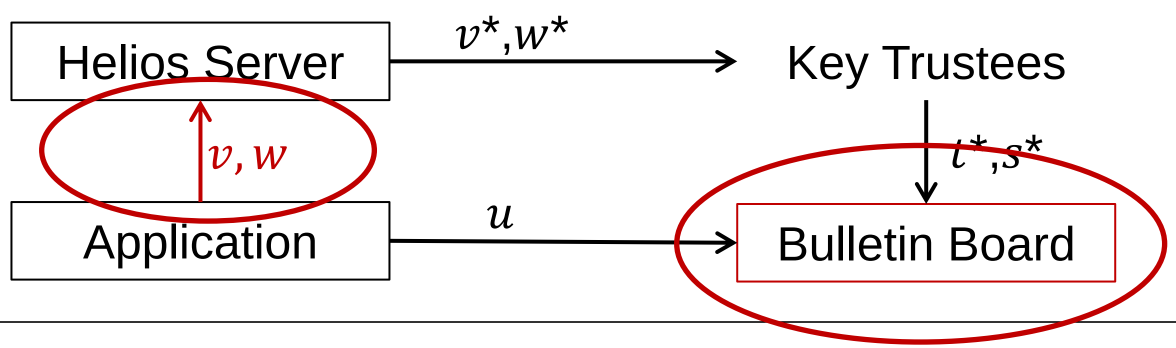
h) The Helios server sends the voter an email containing the hash of the cast vote.



Everlasting Privacy

- Additional Assumption: There exists a private channel between the user‘s browser and the server (e.g. by sending keys per mail).

- Additional Property:
  - Everlasting Privacy towards the public.
  - Attacking this version requires much more work.



Modified Tallying Process

TECHNISCHE UNIVERSITÄT DARMSTADT

1	$u_1(1) = \alpha^{s_1(1)}\beta^{t_1(1)}$	…	$u_l(1) = \alpha^{s_l(1)}\beta^{t_l(1)}$
:	:	:	:
V	$u_1(V) = \alpha^{s_1(j)}\beta^{t_1(j)}$	…	$u_l(V) = \alpha^{s_l(j)}\beta^{t_l(j)}$

$D(v_1)$	…	$D(v_l)$
$s_1{}^* = \sum_j s_1(j)$	…	$s_l{}^* = \sum_j s_l(j)$

$D(w_1)$	…	$D(w_l)$
$t_1{}^* = \sum_j t_1(j)$	…	$t_l{}^* = \sum_j t_l(j)$

$\prod_j u_1(j) = \alpha^{s_1{}^*}\beta^{t_1{}^*}$	…	$\prod_j u_l(j) = \alpha^{s_l{}^*}\beta^{t_l{}^*}$

Public Helios Bulletin Board

1	$v_1(1) = \gamma^{s_1(1)}(r_1(1))^N$	…	$v_l(1) = \gamma^{s_l(1)}(r_l(1))^N$
:	:	:	:
V	$v_1(V) = \gamma^{s_1(V)}(r_1(V))^N$	…	$v_l(V) = \gamma^{s_l(V)}(r_l(V))^N$

1	$w_1(1) = \gamma^{t_1(1)}(r_1'(1))^N$	…	$w_l(1) = \gamma^{t_l(1)}(r_l'(1))^N$
:	:	:	:
V	$w_1(V) = \gamma^{t_1(V)}(r_1'(V))^N$	…	$w_l(V) = \gamma^{t_l(V)}(r_l'(V))^N$

decrypt and publish

$v_1{}^* = \prod_j v_1(j)$	…	$v_l{}^* = \prod_j v_l(j)$
$w_1{}^* = \prod_j w_1(j)$	…	$w_l{}^* = \prod_j w_l(j)$

Helios Server (private Information)

Modified Tallying Process

1	$u_1(1) = \alpha^{s_1(1)}\beta^{t_1(1)}$	…	$u_l(1) = \alpha^{s_l(1)}\beta^{t_l(1)}$
:	:	:	:
V	$u_1(V) = \alpha^{s_1(j)}\beta^{t_1(j)}$	…	$u_l(V) = \alpha^{s_l(j)}\beta^{t_l(j)}$

$D(v_1)$	…	$D(v_l)$
$s_1{}^* = \sum_j s_1(j)$	…	$s_l{}^* = \sum_j s_l(j)$

$D(w_1)$	…	$D(w_l)$
$t_1{}^* = \sum_j t_1(j)$	…	$t_l{}^* = \sum_j t_l(j)$

$\prod_j u_1(j) = \alpha^{s_1{}^*}\beta^{t_1{}^*}$	…	$\prod_j u_l(j) = \alpha^{s_l{}^*}\beta^{t_l{}^*}$

Public Helios Bulletin Board

1	$v_1(1) = \gamma^{s_1(1)}(r_1(1))^N$	…	$v_l(1) = \gamma^{s_l(1)}(r_l(1))^N$
:	:	:	:
V	$v_1(V) = \gamma^{s_1(V)}(r_1(V))^N$	…	$v_l(V) = \gamma^{s_l(V)}(r_l(V))^N$
1	$w_1(1) = \gamma^{t_1(1)}(r_1{}'(1))^N$	…	$w_l(1) = \gamma^{t_l(1)}(r_l{}'(1))^N$
:	:	:	:
V	$w_1(V) = \gamma^{t_1(V)}(r_1{}'(V))^N$	…	$w_l(V) = \gamma^{t_l(V)}(r_l{}'(V))^N$

decrypt and publish

$v_1{}^* = \prod_j v_1(j)$	…	$v_l{}^* = \prod_j v_l(j)$

Note: The Message space of the Commitment and the Encryption Scheme must match.

Helios Server (private information)

Parameters

Need a commitment scheme with a matching semantically secure encryption scheme.

Construction by Moran and Naor (Split Ballot):

Paillier: $\mathrm{N} = p_1 p_2$ where p_1 and p_2 are safe primes.

Pedersen Commitments: Subgroup of Group $\mathbb{Z}^*{}_{4N+1}$ having order N, where $4N + 1$ must be prime.

Proofs

1) Public Proof - Proof of validity of the ballot
    1) Each entry of the vector is either 0 or 1.

       Modification of proof of validity [CFSY96].
    2) Only one entry equals 1.

       Prove knowledge of $\prod_{i=1}^{l} u_i \beta^{-1}$ using Schnorr.

2) Private Proof - Consistency of the commitment and encrypted information

   Cut-and-Choose (Compare to [MN10]).

Proof of Validity

Prover $v = 1$	Prover $v = 0$		Verifier
$\alpha, r_1, d_1, w_2 \in_R \mathbb{Z}_q$	$\alpha, r_2, d_2, w_1 \in_R \mathbb{Z}_q$		
$B \leftarrow \alpha^s \beta$	$B \leftarrow \alpha^s$		
$a_1 \leftarrow \alpha^{r_1} B^{-d_1}$	$a_1 \leftarrow \alpha^{w_1}$		
$a_2 \leftarrow \alpha^{w_2}$	$a_2 \leftarrow \alpha^{r_2} (B/\beta)^{-d_2}$	$\xrightarrow{B, a_1, a_2}$	
$d_2 \leftarrow c - d_1$	$d_1 \leftarrow c - d_2$	$\xleftarrow{c}$	$c \in_R \mathbb{Z}_q$
$r_2 \leftarrow w_2 + s d_2$	$r_1 \leftarrow w_1 + s d_1$	$\xrightarrow{d_1, d_2, r_1, r_2}$	$d_1 + d_2 \stackrel{?}{=} c$ $\alpha^{r_1} \stackrel{?}{=} a_1 B^{d_1}$ $\alpha^{r_2} \stackrel{?}{=} a_2 (B/\beta)^{d_2}$

Consistency Proof

$u = \alpha^s \beta^t, v = \gamma^s r^N, w = \gamma^t r'^N$

Show knowledge of t, s, r, r' by Cut-and-Choose

1. Choose t', s', r'', r''' and generate second triple u', v', w'
2. Challenge: 0 or 1
3. If 0 publish t', s', r'', r''', if 1 publish $\mathrm{t + t', s + s', r * r'', r' * r'''}$.
4. Check and
5. Repeat

$u' \stackrel{?}{=} u\, \alpha^{s'} \beta^{t'}$

$u = \alpha^s \beta^t$

$v = \gamma^s r^N$
$w = \gamma^t r'^N$

$u' \stackrel{?}{=} \alpha^{s+s'} \beta^{t+t'}$

$v' \stackrel{?}{=} \gamma^{s+s'} (r * r'')^N$
$w' \stackrel{?}{=} \gamma^{t+t'} (r' * r''')^N$

$v' \stackrel{?}{=} v\, \gamma^{s'} r''^N, w' \stackrel{?}{=} w\, \gamma^{t'} r'''^N$

Consistency Proof

$u = \alpha^s \beta^t, v = \gamma^s r^N, w = \gamma^t r'^N$

Show knowledge of t, s, r, r' by Cut-and-Choose

1. Choose t', s', r'', r''' and generate second triple u', v', w'
2. Challenge: 0 or 1
3. If 0 publish t', s', r'', r''', if 1 publish $\mathrm{t + t', s + s', r * r'', r' * r'''}$.
4. Check and
5. Repeat

Probability that two different values for s and t will not be detected is

$$\frac{1}{2^b}$$

for b iterations.

$u' \stackrel{?}{=} u\, \alpha^{s'} \beta^{t'}$

$u = \alpha^s \beta^t$

$v = \gamma^s r^N$

$w = \gamma^t r'^N$

$w' \stackrel{?}{=} \gamma^{t+t'} (r' * r''')^N$

$v' \stackrel{?}{=} v\, \gamma^{s'}\, r''^N, w' \stackrel{?}{=} w\, \gamma^{t'}\, r'''^N$

Future Work

- Improvements for booth voting: Principle of free suffrage.
- Everlasting privacy can be implemented also for electronic voting systems like Prêt à Voter, Scratch & Vote and MarkPledge.
- Cuvelier, Peters and Pereira‘s “Efficient commitment scheme with matching encryption scheme based on elliptic curves“ (presented at SecVote 2012).

Thank you

Questions?