Towards a Publicly-Verifiable Mixing Based Voting System Providing Everlasting Privacy

Denise Demirel, Jeroen van de Graaf, Roberto Araújo

Johannes Buchmann

Authors

Denise Demirel, Cryptography and Computer Algebra Group, Technische Universität Darmstadt, Germany

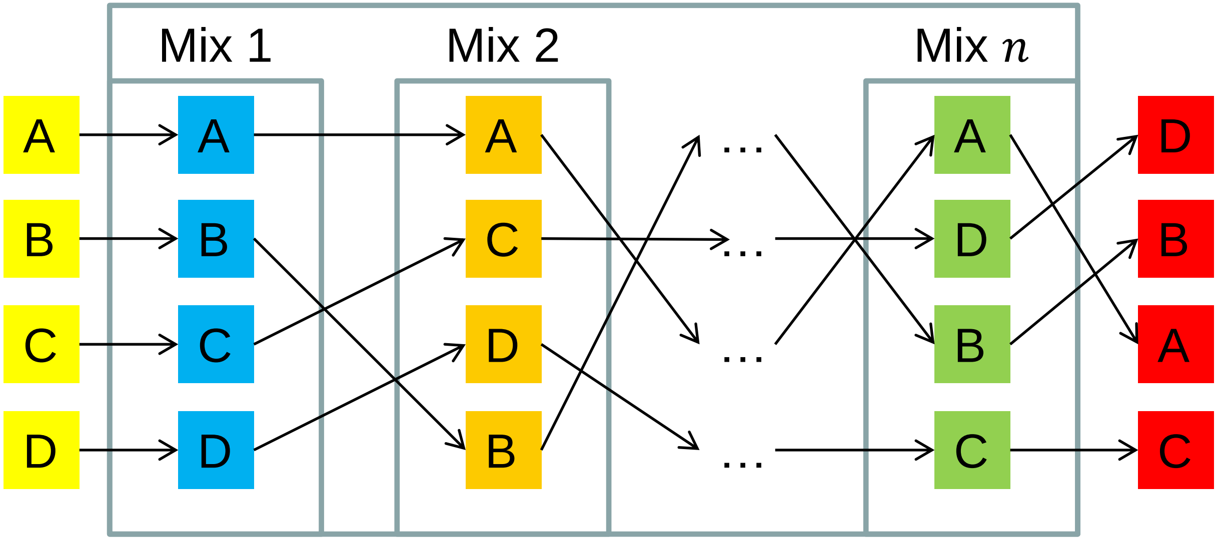
Jeroen van de Graaf, Departamento de Ciência de Computação, Universidade Federal de Minas Gerias, Brazil

Roberto Araújo, Faculdade de Computação, Universidade Federal do Pará, Brazil.

Johannes Buchmann, Cryptography and Computer Algebra Group, Technische Universität Darmstadt, Germany

Introduction

- Mix-nets were introduced by David Chaum in 1981
- Reencryption mix-nets [1993, Park et al.] allow third parties to verify the correctness of the shuffling procedure



- Prêt à Voter, Helios, Civitas…

Motivation

- Votes are encrypted using public key cryptography. Voter gets a receipt.
- Voter verifies that the encrypted vote is contained in the tally and that the ciphertext is unmodified.
- Mix-nets are used to make these votes anonymous before decrypting.
- Verification of its correct function by publishing additional information.

Public-Key Encryption

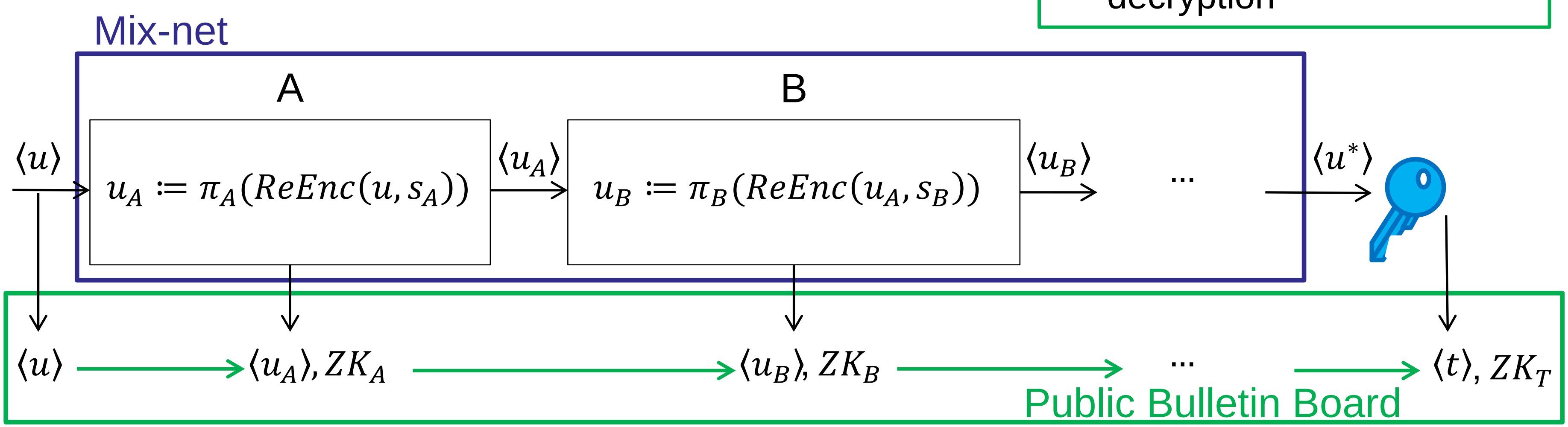
Probabilistic public-key encryption algorithm (Gen, Enc, Dec)

- Semantical (CCA) security
- Homomorphic such that
$$\forall m, m' \in G, \forall r, r' \in H: Enc(m, r) \cdot Enc(m', r') = Enc(m +_G m', r +_H r')$$
It follows that messages can be "reencrypted"
$$ReEnc(u, r') = Enc(m, r) \cdot Enc(0_G, r') = Enc(m, r + r')$$
- (k, n)-threshold decryption

Reencryption Mix-net

- Input $u = Enc(t, s)$, with message t and randomness s
- $ReEnc(u, s') = Enc(t, s) \cdot Enc(0_G, s') = Enc(t, s + s')$

Public Verification Process
- ZK-Proof of correct shuffing
- ZK-Proof of correct decryption



Computational Privacy

- Homomorphic public-key cryptography, e.g., Paillier, Elgamal
- Computational assumptions
- Current implementations have an expiration date
- Violates principle of free and secret suffrage
- Possible solution: using an unconditional hiding commitment scheme to encode the published audit information

Commitment Scheme

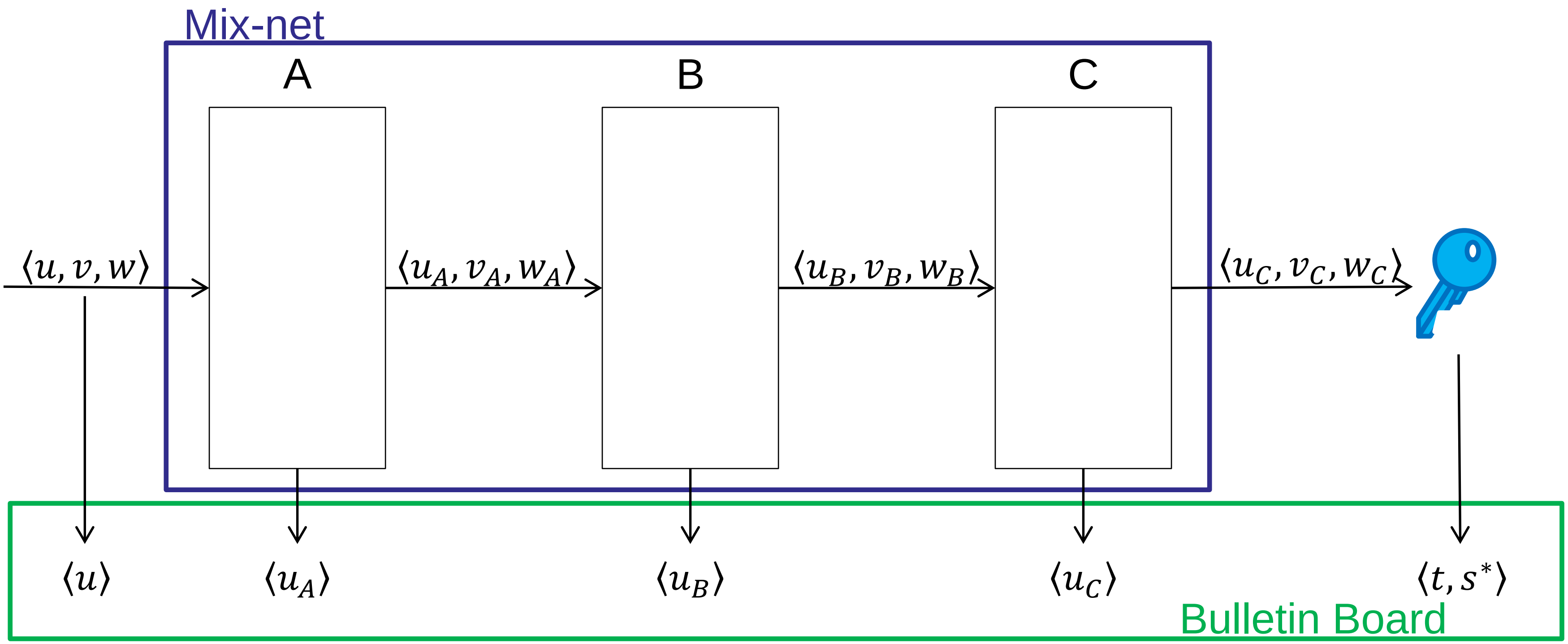
A (non-interactive and unconditional hiding) commitment scheme $(GenCom, Com, Unv)$

- $GenCom(1^{\kappa})$ defines message space and randomization space for security parameter κ.
- $c = Com(t, s) \in C$ generates commitment $c \in C$ to $t \in M$ and $s \in R$.
- $Unv(c, t, s)$ returns t if $c = Com(t, s)$ and $\perp$ if not.

Mix-net Providing Everlasting Privacy Towards the Public

Input

$u = Com(t, s)$, commitment to message t with decommitment s
$v = Enc_M(t), w = Enc_R(s)$ opening values encrypted with public-key cryptography

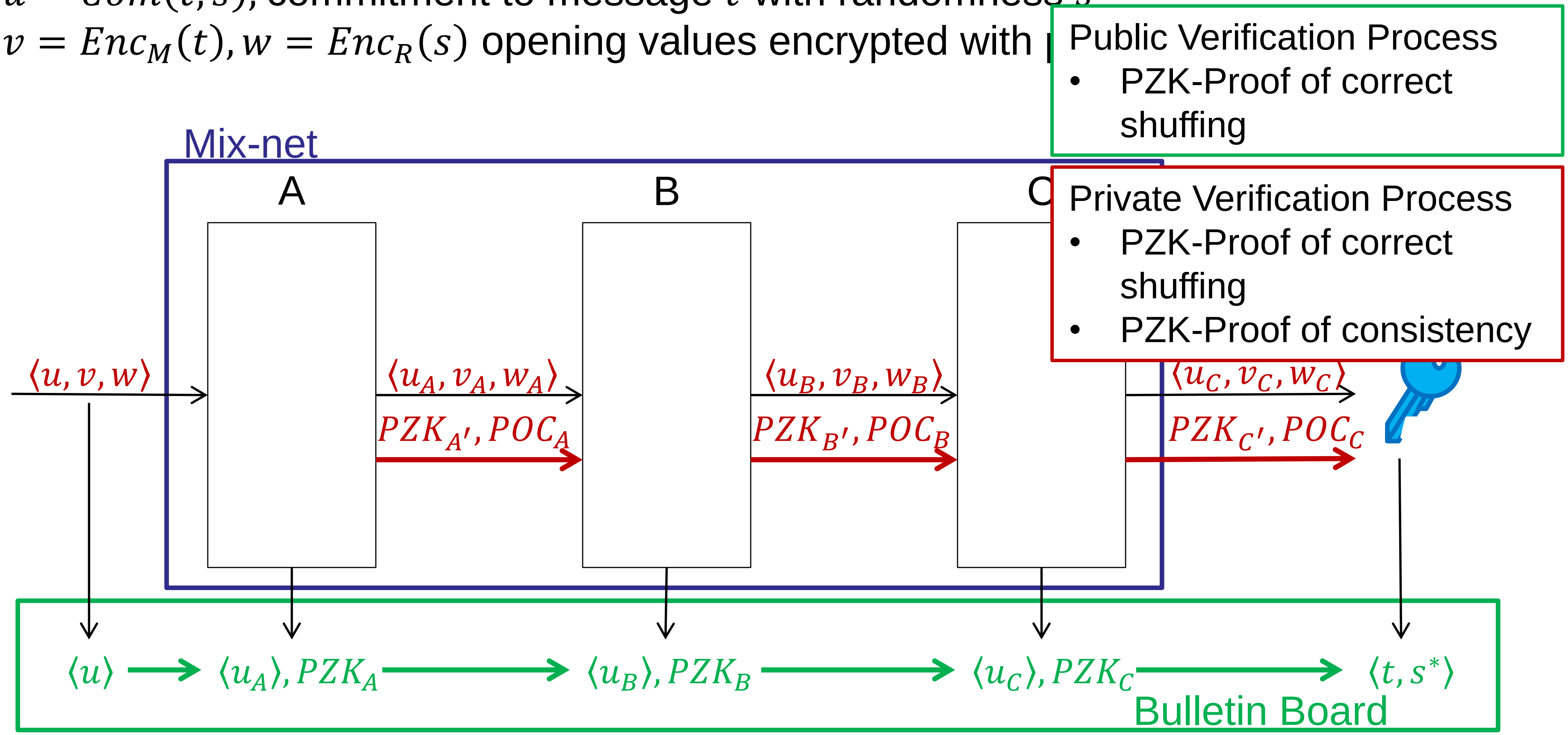


Mix-net Providing Everlasting Privacy Towards the Public

Input

$u = Com(t, s)$, commitment to message t with randomness s

$v = Enc_M(t), w = Enc_R(s)$ opening values encrypted with p



Assumptions

Correctness:

- The random challenge bits are unpredictable
- The authorities cannot break the computational binding property of the cryptographic primitives for the parameters chosen before the whole process is completed

Robustness:

- "(k, n) -threshold"- decryption at least k out of n key holders participate in the decryption process
- **The authorities carry out the private verification process**

Assumptions (2)

Privacy:

- The authorities cannot break the computational assumption of the encryption scheme.
- At least one mix is honest and keeps the association between its input and output values secret.
- Using "(k, n) -threshold"- decryption at least $(n - k + 1)$ out of n key holders keep their key portion secret.
- **Private channels can be used to send the encrypted opening values.**

Properties

Correctness: Changes on the messages will be detected with overwhelming probability even if all authorities collaborate.

Robustness: The protocol always terminates successfully. If one authority cheats, it will get caught with overwhelming probability.

Privacy: During the process, the messages remains secret as long as a minimum number of authorities act honest.

Everlasting privacy towards observers: All published data do not reveal any information about the messages.

Improving 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

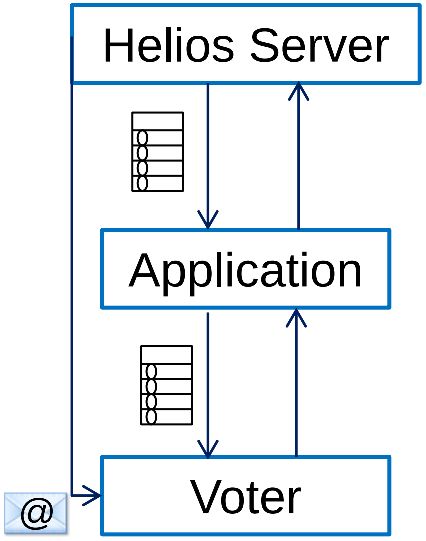
Helios Election Server

Helios Election Process (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

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.

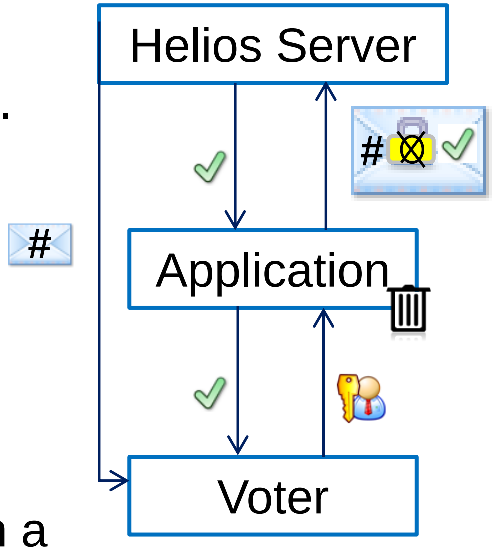
d) Hash of encrypted vote is shown to the voter.

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.



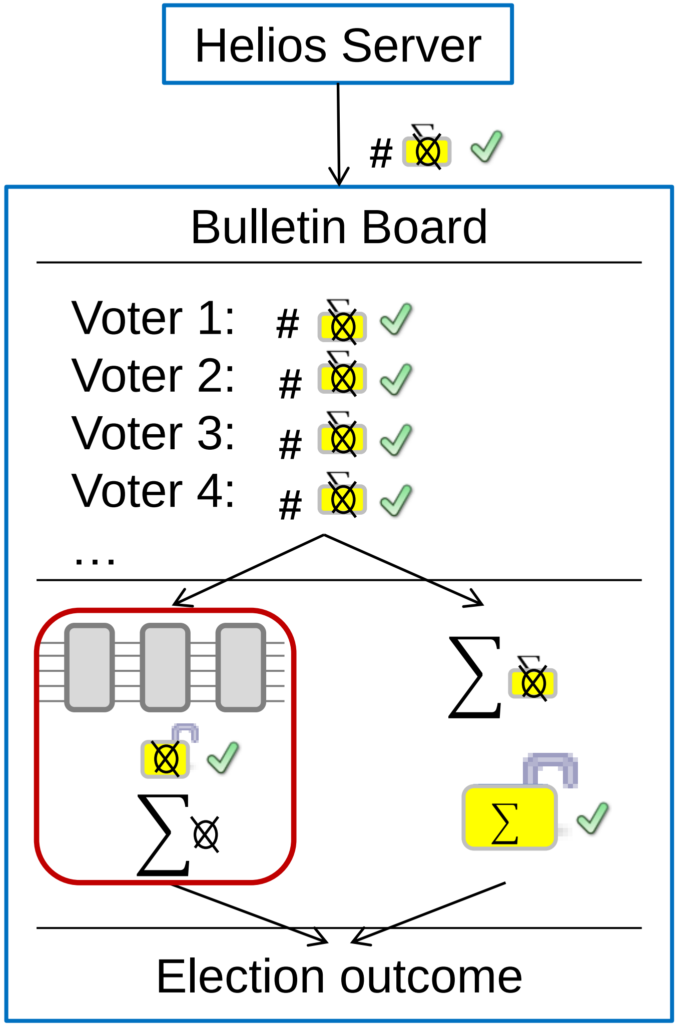
Helios Election Process (3)

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.

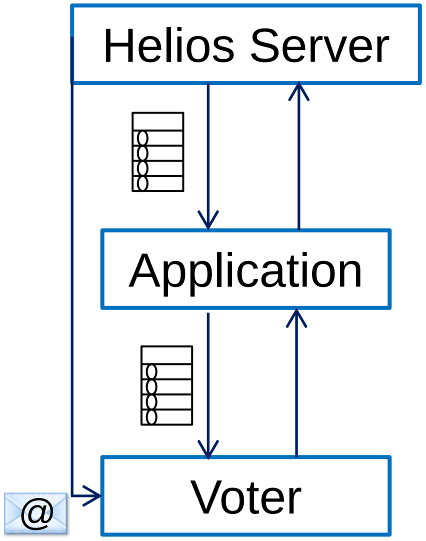
- Mixing + decryption + tallying
- Homomorphic tallying + decryption



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 commitment scheme).

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 a commitment scheme.

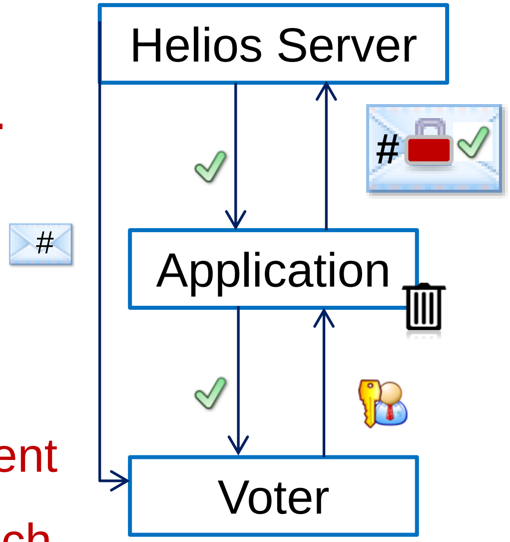
d) The Hash of the commitment is shown to the voter.

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.



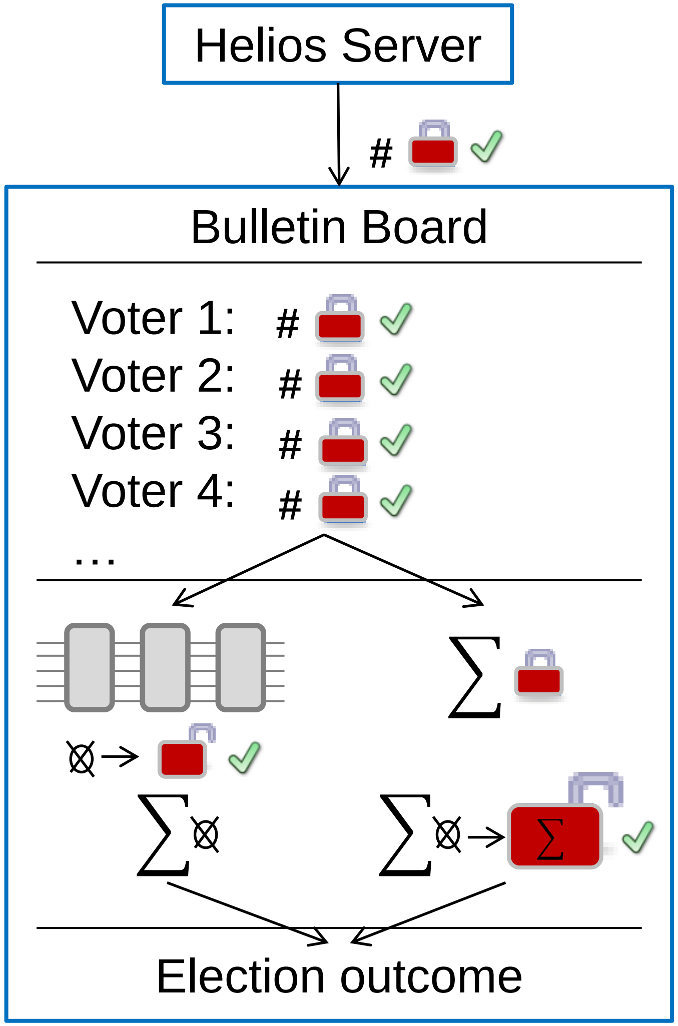
Helios Election Process (3)

3. Tallying and publishing of votes

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

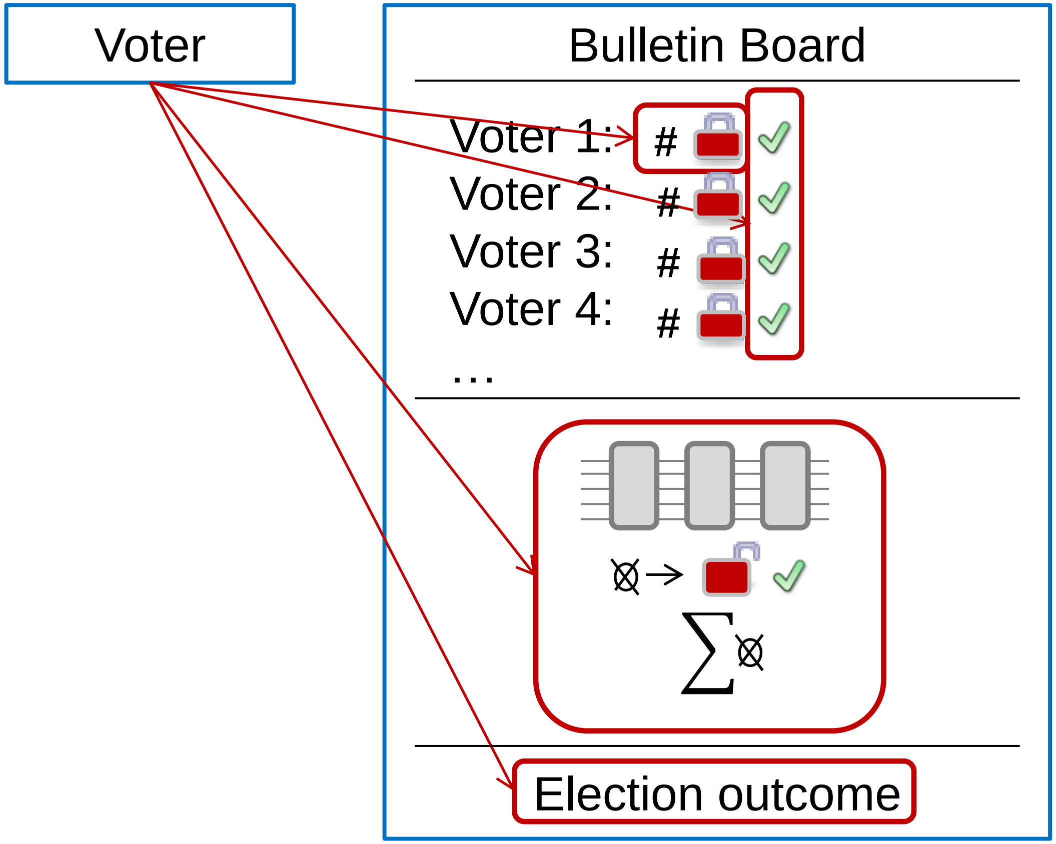
b) The Helios server computes the election outcome.

- Mixing + decryption + tallying
- Homomorphic tallying + decryption



Properties

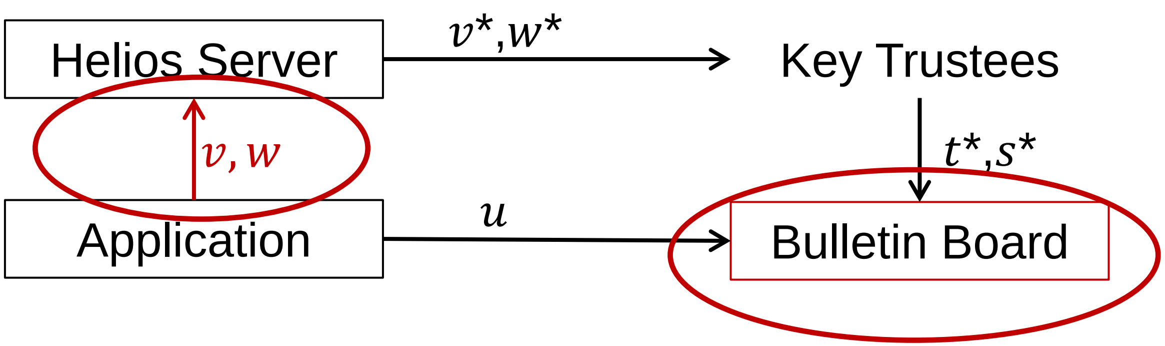
- Individual Verifiability
- Universal Verifiability
- Correctness



Everlasting Privacy Towards the Public

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

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



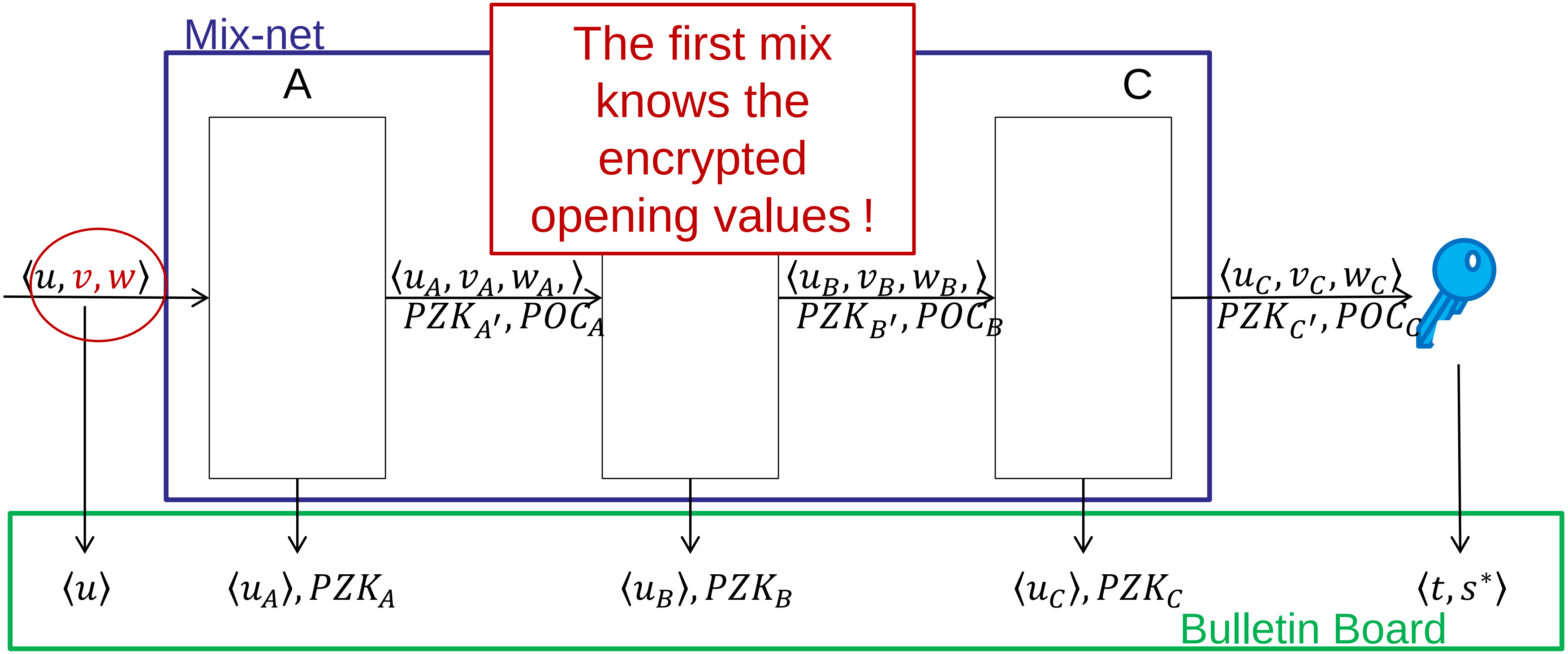
Everlasting Privacy Towards the Authorities

TECHNISCHE UNIVERSITÄT DARMSTADT

Input

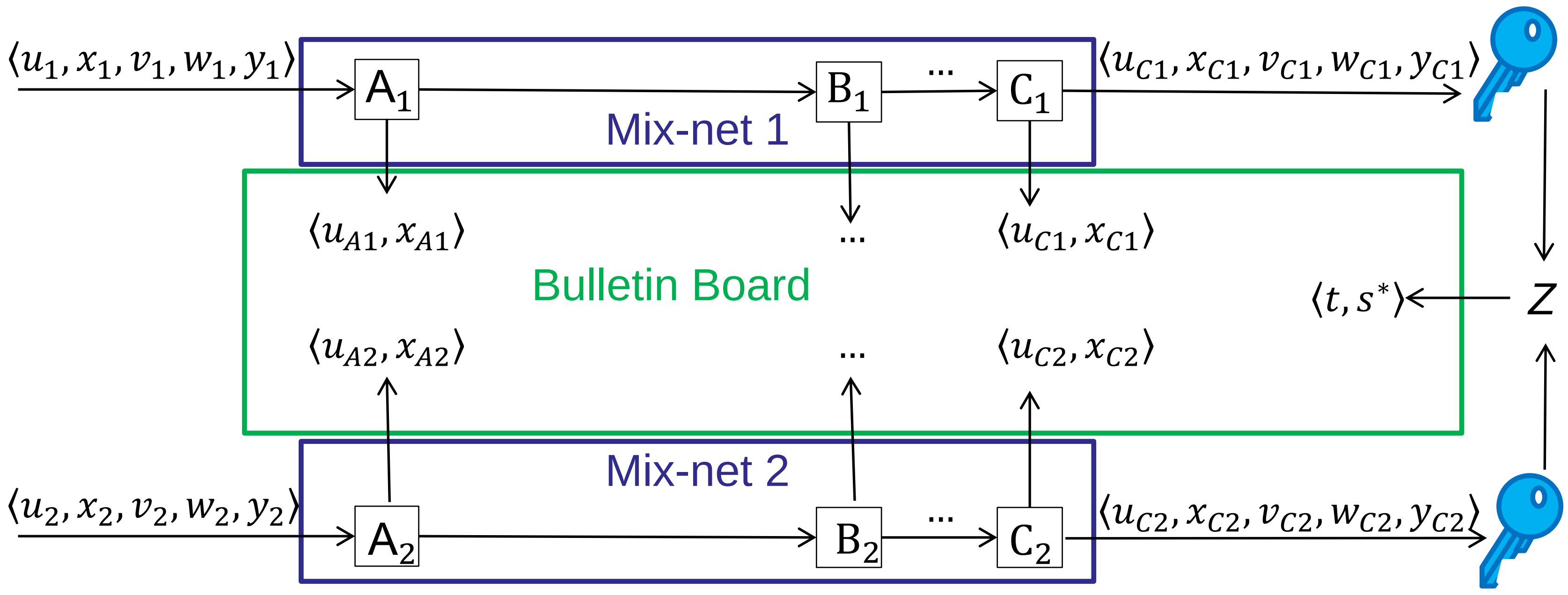
$u = Com(t, s)$, commitment to message t with randomness s

$v = Enc_M(t), w = Enc_R(s)$ opening values encrypted with public-key cryptography



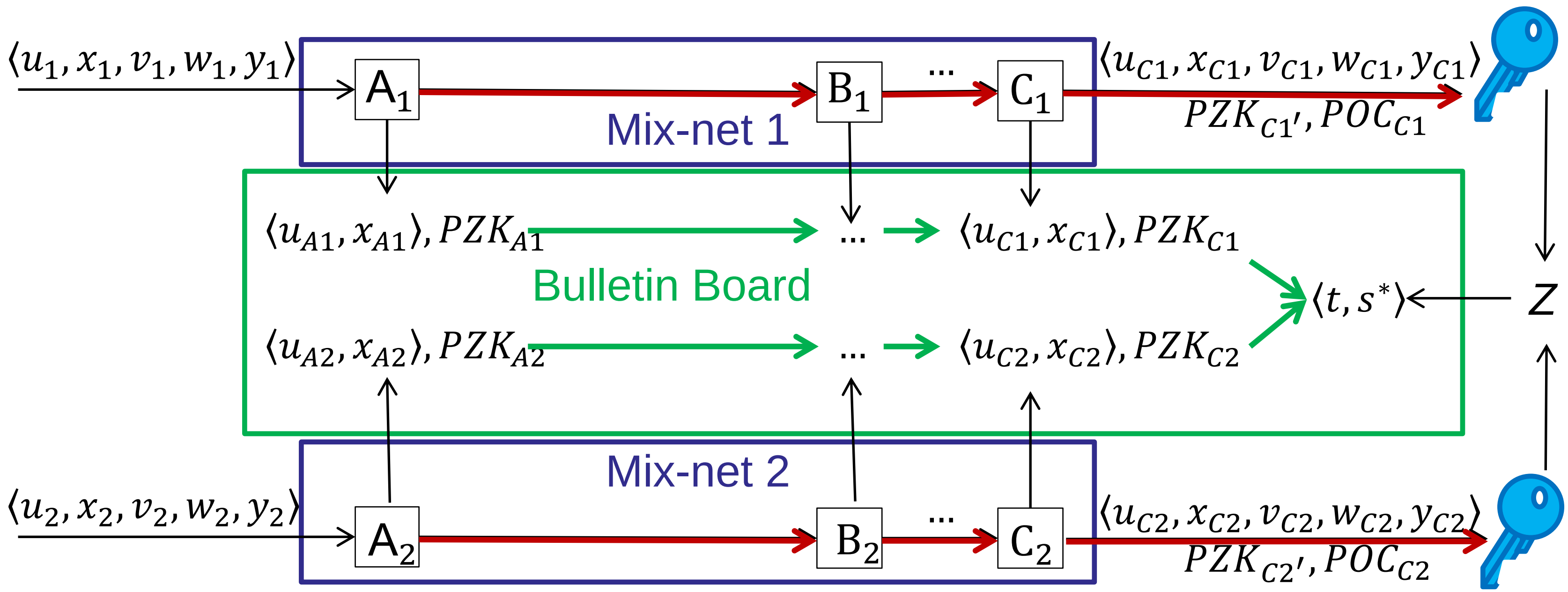
Secret Sharing

- Split message t: $t_1 + t_2 = t$
- Choose ID
- Submit $u_1, x_1, v_1, x_1 = Com(ID, r_1)$, $y_1 = Enc_R(r_1)$ to Mix-net 1
 and $u_2, x_2, v_2, x_2 = Com(ID, r_2)$, $y_2 = Enc_R(r_2)$ to Mix-net 2

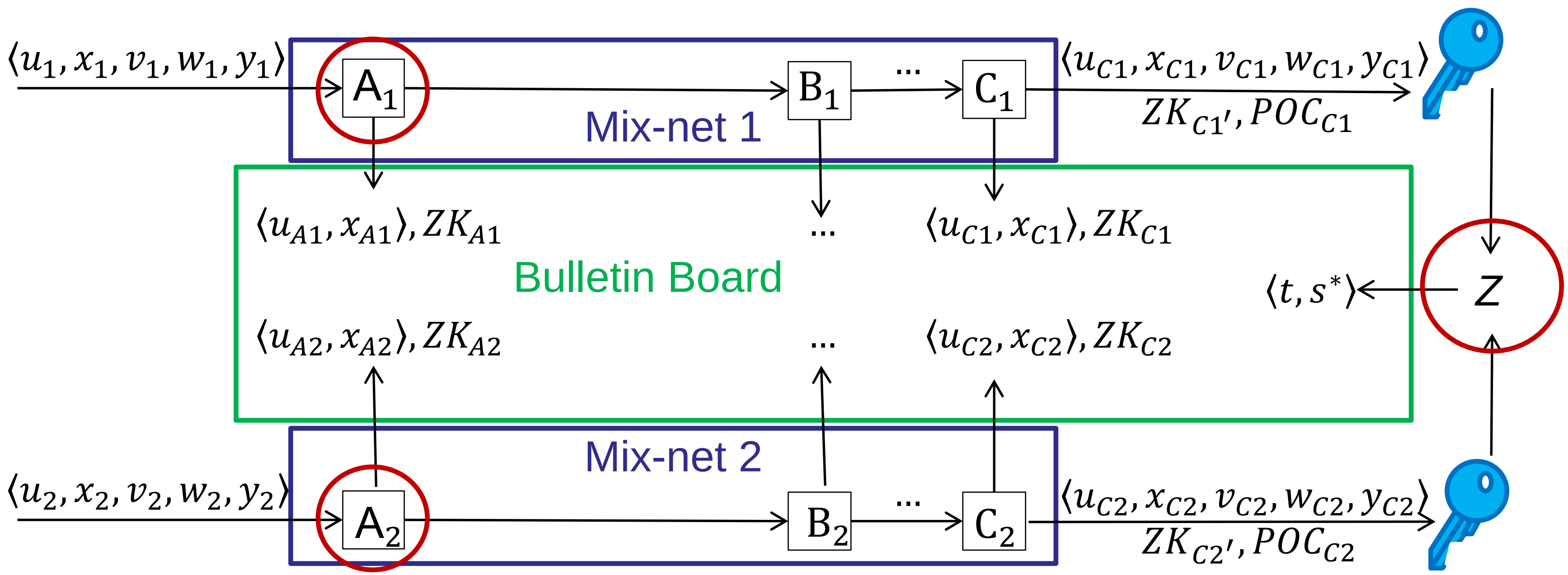


Secret Sharing

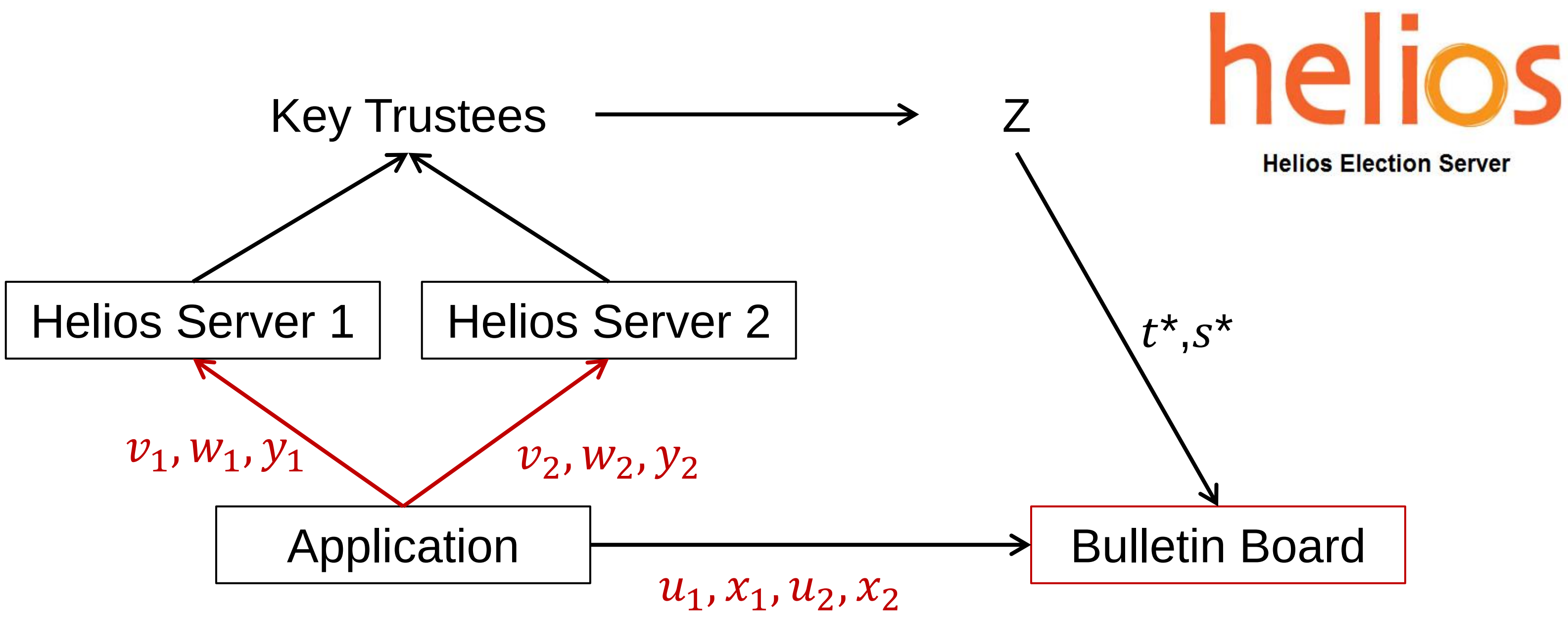
- Split message t: $t_1 + t_2 = t$
- Choose ID
- Submit $u_1, x_1, v_1, x_1 = Com(ID, r_1)$, $y_1 = Enc_R(r_1)$ to Mix-net 1
  and $u_2, x_2, v_2, x_2 = Com(ID, r_2)$, $y_2 = Enc_R(r_2)$ to Mix-net 2



No Single Point of Failure



Improving Helios



+ Everlasting Privacy towards the authorities.

– Robustness

Commitment Scheme - Requirements

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Correctness**: For any $t \in M, s \in R: Unv(Com(t,s),t,s) = t$
- **Non-Interactive**
- **Computationally Binding:** Given $c = Com(t,s)$, for any $\mathrm{PPT\ A}$ the probability to find (t', s') with $t \neq t'$ such that $Com(t,s) = Com(t', s')$ is negligible in κ.
- **Unconditionally Hiding:** Distribution of $Com(t,s)$ and $Com(t', s')$ must be identical when $s, s' \in R$ are chosen uniformly at random.
- **Homomorphic:** For all $t, t' \in M$ and $s, s' \in R$

$$Com(t,s) \cdot_C \; Com(t', s') = Com(t +_M t', s +_R s')$$

It follows that messages can be "rerandomize"

$$ReRand(c, s') = Com(t,s) \cdot_C \; Com(0_M, s') = Com(t, s +_R s')$$

Cryptographic Primitives

1. Commitment scheme (homorphic, unconditionally hiding)

2. Two instances of the encryption scheme

1. Enc_M which is homomorphic over message space M

2. Enc_R which is homomorphic over randomization space R

- Paillier encryption and adapted Pedersen Commitments [MN07]
- EC groups with asymmetric pairing (E.Cuvelier,O.Pereira,T.Peters)

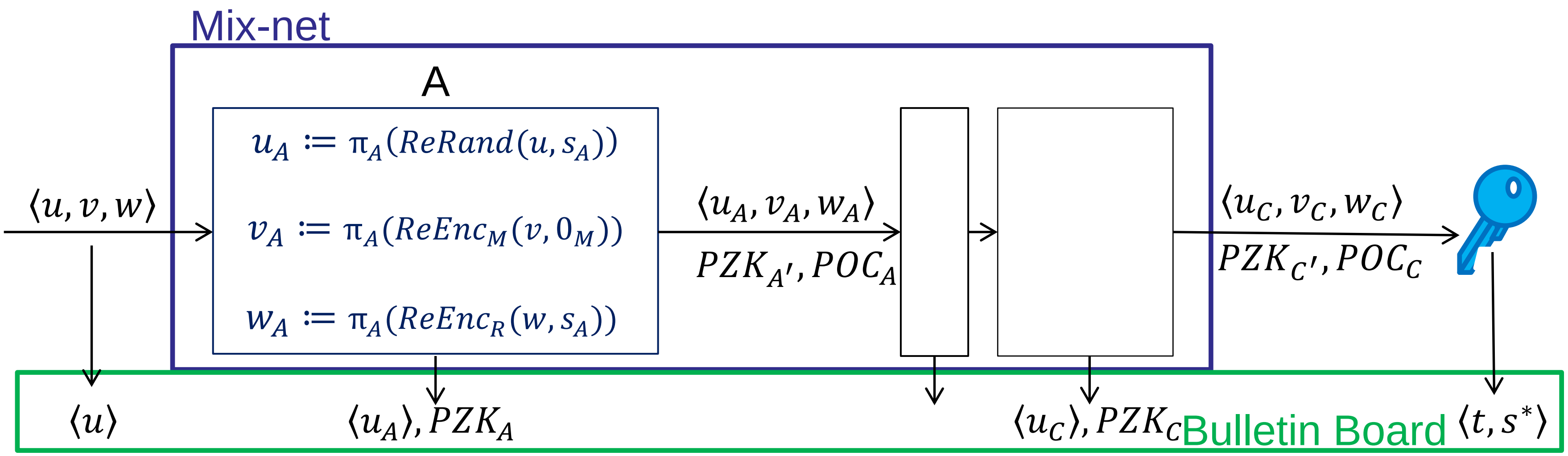
3. Perfect zero-knowledge proof of correct shuffling and consistency

- Non-interactive zero-knowledge shuffle argument, e.g., [Groth2010] or [LZ12].
- Cut-and-choose based shuffling proof [SK95]

Mixing Process

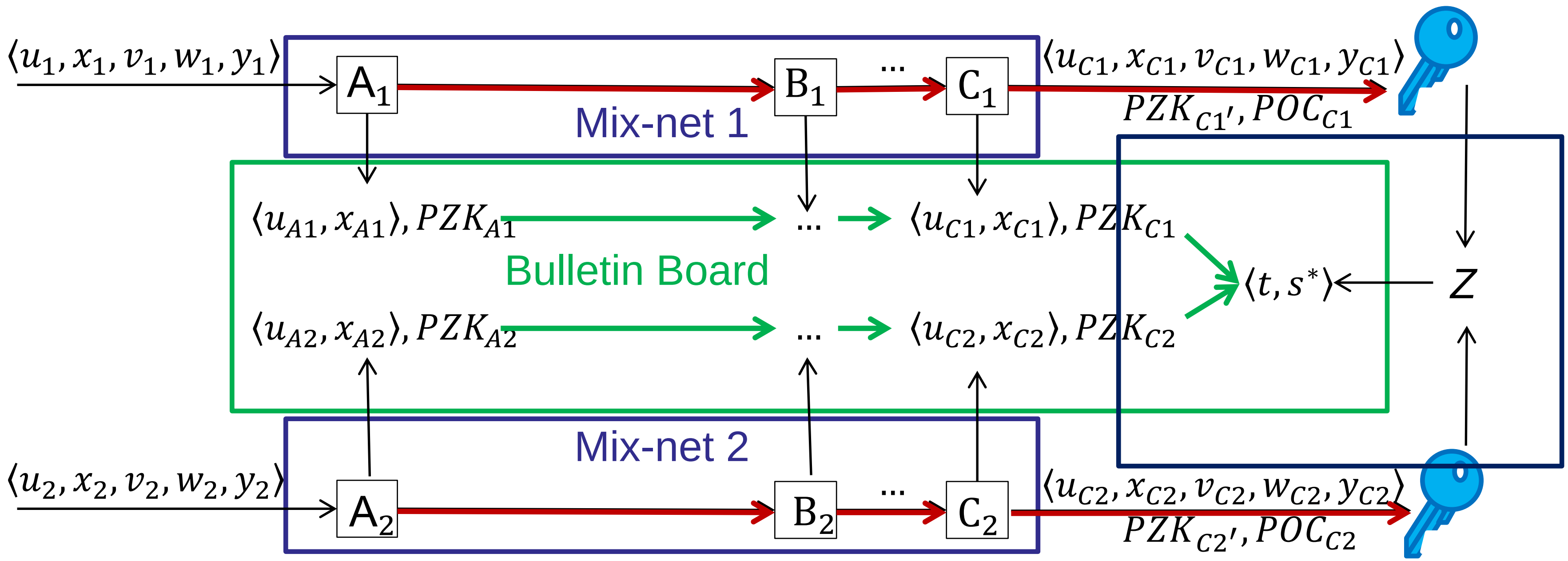
Anonymisation

- Rerandomize public output $u'_A = u \cdot Com(\mathbf{0_M}, \boldsymbol{s_A}) = Com(t, s + \boldsymbol{s_A})$
- Reencrypt message $v'_A = v \cdot Enc_M(\mathbf{0_M}) = Enc_M(t)$
- Adapt decommitment value $w'_A = w_i \cdot Enc_R(\boldsymbol{s_A}) = Enc_R(s + \boldsymbol{s_A})$
- Permutation $\langle u_A, v_A, w_A \rangle = \langle \boldsymbol{\pi_A}(u'_A), \boldsymbol{\pi_A}(v'_A), \boldsymbol{\pi_A}(w'_A) \rangle$



Secret Sharing

- Split message t: $t_1 + t_2 = t$
- Choose ID
- Submit $u_1, x_1, v_1, x_1 = Com(ID, r_1)$, $y_1 = Enc_R(r_1)$ to Mix-net 1
  and $u_2, x_2, v_2, x_2 = Com(ID, r_2)$, $y_2 = Enc_R(r_2)$ to Mix-net 2



Matching Shares

Matching

- Reveal decommitment value: $Dec_R(y_{Ci}) = s_i'^*$,
- Reveal association: $ID^* = x_{Ci} \cdot Com(0_M, -s_i'^*) = Com(ID, 0_R)$
- Match shares with same ID^* and publish opening values

Proof of Correct Matching

- x_{C1}: $Com(ID, s_1'^*)$ and x_{C2}: $Com(ID, s_2'^*)$,
- $Com(ID, s_1'^*) = Com(ID, s_2'^*) \cdot Com(0_M, s_1'^* - s_2'^*))$
- Proof of correct matching by showing knowledge of "rerandomization" value $(s_1'^* - s_2'^*)$, e.g., by cut-and-choose.

Consistency Proof

TECHNISCHE UNIVERSITÄT DARMSTADT

$$u = Com(t,s), v = Enc_M(t,r), w = Enc_R(s,r')$$

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

1. Choose t', s', r'', r''' and generate second tripl
2. Challenge: 0 or 1
3. If 0 publish t', s', r'', r''', if 1 publish $t + t', s +$
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.

$$u' \stackrel{?}{=} u \cdot Com(t', s')$$

$$u = Com(t,s)$$

$$v = Enc_M(t,r)$$

$$w = Enc_R(s,r')$$

$$u' \stackrel{?}{=} Com(t + t', s + s')$$

$$v' \stackrel{?}{=} Enc_M(t + t', r \cdot r'')$$

$$w' \stackrel{?}{=} Enc_R(s + s', r' \cdot r''')$$

$$v' \stackrel{?}{=} v \cdot Enc_M(t', r''), w' \stackrel{?}{=} w \cdot Enc_R(s', r''')$$

Public Verification of Correct Shuffling – Cut-and-choose

$u_A = \boldsymbol{\pi_A}(u \cdot Com(0_M, \boldsymbol{s_A}))$

Input u → Output u_A

Show knowledge of $\boldsymbol{\pi_A}$ and $\boldsymbol{s_A}$

1. Choose π_L and s_L at random
2. Generate and publish intermediate batch $u'_A = \pi_L(ReRand\langle u, s_L\rangle)$
3. Challenge: “left” or “right”
4. If “left” publish π_L and s_L ,if “right” publish π_R und s_R
5. Check and
6. Repeat

u —(π_L , s_L)→ $u'_A = \pi_L(u \cdot Com(0_M, s_L))$ —(π_R , s_R)→ $u_A = \pi_R(u'_A \cdot Com(0_M, s_R))$

$s_A = s_L + s_R$ and $\pi_L \circ \pi_R = \pi_A$

Private Verification of Correct Shuffling – Cut-and-choose

$u_A = \boldsymbol{\pi_A}(u \cdot Com(0_M, \boldsymbol{s_A}))$

Input u	Output u_A
Input v, w	Output v_A, w_A

$v_A = \boldsymbol{\pi_A}(v \cdot Enc_M(0_M, \boldsymbol{r_A})), w_A = \boldsymbol{\pi_A}(w \cdot Enc_R(\boldsymbol{s_A}, \boldsymbol{r'_A}))$

Show knowledge of π_A, s_A, r_A and r'_A

π_L, s_L $\qquad$ π_R, s_R

u	$u'_A = \pi_L(u \cdot Com(0_M, s_L))$	$u_A = \pi_R(u'_A \cdot Com(0_M, s_R))$
v w	$v'_A = \pi_L(v \cdot Enc_M(0_M, r_L))$ $w'_A = \pi_L(w \cdot Enc_R(s_L, r'_L))$	$v_A = \pi_R(v \cdot Enc_M(0_M, r_R))$ $w_A = \pi_R(w \cdot Enc_R(s_A, r'_R))$

π_L, s_L, r_L, r'_L $\qquad$ π_R, s_R, r_R, r'_R

$\pi_A = \pi_L \circ \pi_R, s_A = s_L + s_R, r_A = r_L + r_R$ and $r'_A = r'_L + r'_R$

Summary

Simple mix-net providing everlasting privacy towards the public

Complex mix-net (using secret sharing) providing everlasting privacy towards the authorities

Future work

- Finding more efficient primitives
- Quantum resistant commitment and encryption scheme

Thank you

Questions?