Type-Based Verification of Electronic Voting Protocols

Véronique Cortier, LORIA - CNRS, Nancy

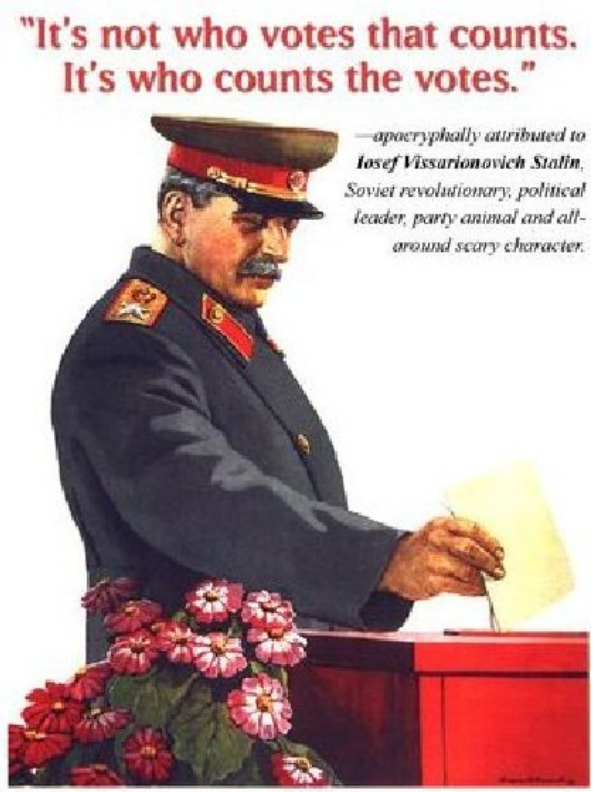
Joint EasyCrypt-F*-CryptoVerif School 2014

Joint work with Fabienne Eigner Steve Kremer, Matteo Maffei, Cyrille Wiedling

Electronic voting

Electronic voting promises

- Convenient, efficient and secure facility for recording and tallying votes (Computers compute better than humans)
- for a variety of types of elections : from small committees or on-line communities through to full-scale national elections

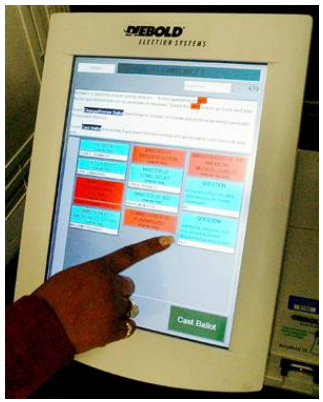


Already used e.g. in Estonia, Norway, USA, France, Australia.
Banned in Germany, Ireland, UK.

Two main families for e-voting

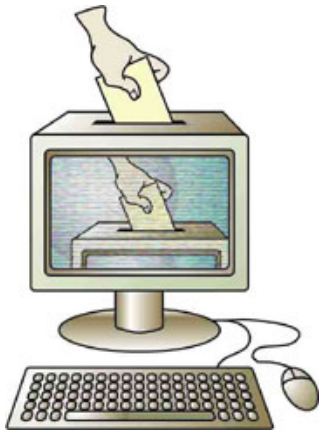
Voting machines

- Voters have to go to a voting station
- External authentication system (e.g. ID card)



Internet voting

- Voters vote from home
- From their own computers
- Systems in use : Norwegian protocol, Estonian protocol, Helios, ...



Confidentiality of the votes

Vote privacy

"No one should know how I voted"

Confidentiality of the votes

Vote privacy

"No one should know how I voted"

Better : Receipt-freeness / Coercion-resistance

"No one should know how I voted,
even if I am willing to tell my vote ! "

Confidentiality of the votes

Vote privacy

"No one should know how I voted"

Better : Receipt-freeness / Coercion-resistance

"No one should know how I voted,
even if I am willing to tell my vote ! "

- vote buying
- coercion

Confidentiality of the votes

Vote privacy

"No one should know how I voted"

Better : Receipt-freeness / Coercion-resistance

"No one should know how I voted,
even if I am willing to tell my vote ! "

- vote buying
- coercion

ebay

Silk Road
anonymous marketplace

Everlasting privacy : no one should know my vote, even when the cryptographic keys will be eventually broken.

Verifiability

End-to-end Verifiability : the result corresponds to the votes intended by the voters, and nothing else.

- Individual Verifiability : Each voter can check that his/her ballot is in the ballot box.
- Universal Verifiability : Everyone can check that the result corresponds to the content of the ballot box.

Verifiability

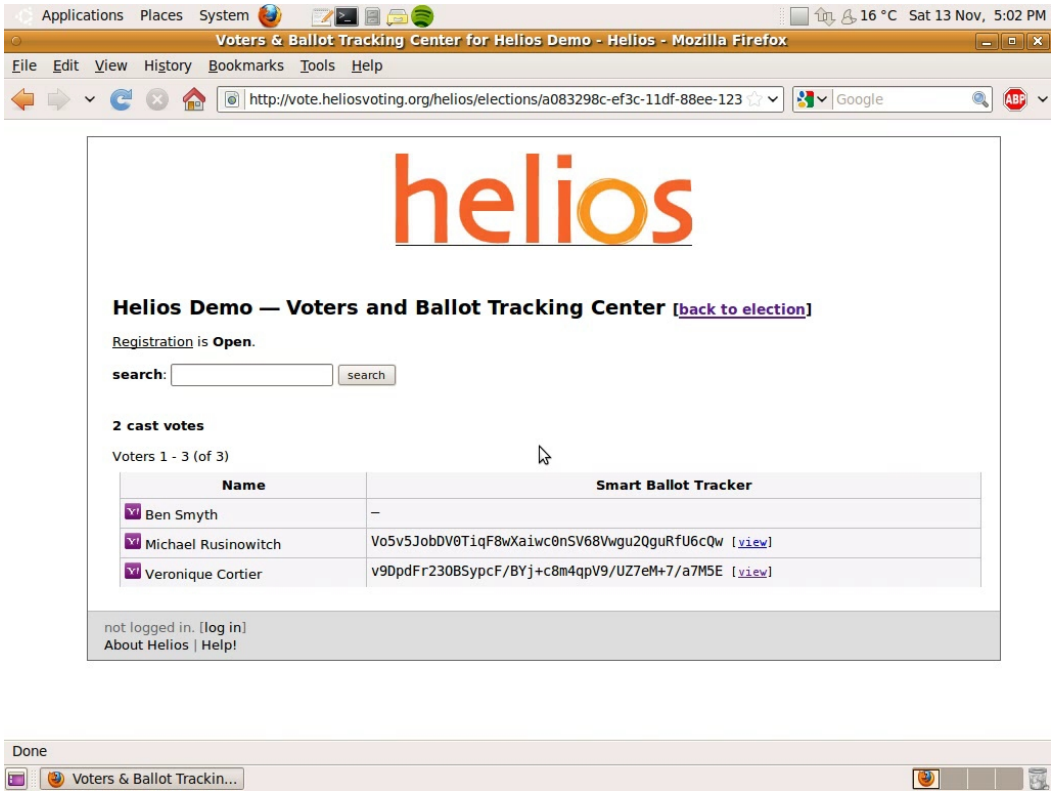
End-to-end Verifiability : the result corresponds to the votes intended by the voters, and nothing else.

- Individual Verifiability : Each voter can check that his/her ballot is in the ballot box.
- Universal Verifiability : Everyone can check that the result corresponds to the content of the ballot box.

You should verify the election, not the system.

A e-voting system : Helios

http ://heliosvoting.org/



Developed by B. Adida *et al*, already in use :

- Election at Louvain University Princeton
- Election of the IACR board (major association in Cryptography)

Behavior of Helios (simplified)

Phase 1 : voting

Bulletin Board

Alice	$\{v_A\}_{\mathsf{pk}(E)}$	$v_A = 0$ or 1
Bob	$\{v_B\}_{\mathsf{pk}(E)}$	$v_B = 0$ or 1
Chris	$\{v_C\}_{\mathsf{pk}(E)}$	$v_C = 0$ or 1

$\mathsf{pk}(E)$: public key, the private key being shared among trustees.

Behavior of Helios (simplified)

Phase 1 : voting

$\{v_D\}_{\mathsf{pk}(E)}$ →

Bulletin Board

Alice	$\{v_A\}_{\mathsf{pk}(E)}$	$v_A = 0$ or 1
Bob	$\{v_B\}_{\mathsf{pk}(E)}$	$v_B = 0$ or 1
Chris	$\{v_C\}_{\mathsf{pk}(E)}$	$v_C = 0$ or 1

$\mathsf{pk}(E)$: public key, the private key being shared among trustees.

Behavior of Helios (simplified)

Phase 1 : voting

Bulletin Board		
Alice	$\{v_A\}_{\mathrm{pk}(E)}$	$v_A = 0$ or 1
Bob	$\{v_B\}_{\mathrm{pk}(E)}$	$v_B = 0$ or 1
Chris	$\{v_C\}_{\mathrm{pk}(E)}$	$v_C = 0$ or 1
David	$\{v_D\}_{\mathrm{pk}(E)}$	$v_D = 0$ or 1

$\mathrm{pk}(E)$: public key, the private key being shared among trustees.

Behavior of Helios (simplified)

Phase 1 : voting

Bulletin Board		
Alice	$\{v_A\}_{\mathrm{pk}(E)}$	$v_A = 0$ or 1
Bob	$\{v_B\}_{\mathrm{pk}(E)}$	$v_B = 0$ or 1
Chris	$\{v_C\}_{\mathrm{pk}(E)}$	$v_C = 0$ or 1
David	$\{v_D\}_{\mathrm{pk}(E)}$	$v_D = 0$ or 1
...	...	

Phase 2 : Tallying using homomorphic encryption (El Gamal)

$$\prod_{i=1}^{n} \{v_i\}_{\mathrm{pk}(E)} = \{\sum_{i=1}^{n} v_i\}_{\mathrm{pk}(E)}$$

based on $g^a * g^b = g^{a+b}$

$\longrightarrow$ Only the final result needs to be decrypted !

$\mathrm{pk}(E)$: public key, the private key being shared among trustees.

This is oversimplified !

$\{v_D\}_{\mathsf{pk}(E)}$ →

Bulletin Board

Alice	$\{v_A\}_{\mathsf{pk}(E)}$
Bob	$\{v_B\}_{\mathsf{pk}(E)}$
Chris	$\{v_C\}_{\mathsf{pk}(E)}$
David	$\{v_D\}_{\mathsf{pk}(E)}$
...	...

$v_A = 0$ or 1

$v_B = 0$ or 1

$v_C = 0$ or 1

Result : $\{v_A + v_B + v_C + v_D + \cdots\}_{\mathsf{pk}(E)}$

This is oversimplified !

$\{v_D\}_{\mathsf{pk}(E)}$ →

Bulletin Board

Alice	$\{v_A\}_{\mathsf{pk}(E)}$
Bob	$\{v_B\}_{\mathsf{pk}(E)}$
Chris	$\{v_C\}_{\mathsf{pk}(E)}$
David	$\{v_D\}_{\mathsf{pk}(E)}$
...	...

$v_A = 0$ or 1

$v_B = 0$ or 1

$v_C = 0$ or 1

$v_D = 100$

Result : $\{v_A + v_B + v_C + 100 + \cdots\}_{\mathsf{pk}(E)}$

A malicious voter can cheat !

This is oversimplified !

$\{v_D\}_{\mathsf{pk}(E)}$

Bulletin Board

Alice	$\{v_A\}_{\mathsf{pk}(E)}$
Bob	$\{v_B\}_{\mathsf{pk}(E)}$
Chris	$\{v_C\}_{\mathsf{pk}(E)}$
David	$\{v_D\}_{\mathsf{pk}(E)}$
...	...

$v_A = 0$ or 1

$v_B = 0$ or 1

$v_C = 0$ or 1

~~$v_D = 100$~~

Result : $\{v_A + v_B + v_C + v_D + \cdots\}_{\mathsf{pk}(E)}$

~~A malicious voter can cheat !~~

In Helios : use of Zero Knowledge Proof

$$\{v_D\}_{\mathsf{pk}(E)}, \mathsf{ZKP}\{v_D = 0 \text{ or } 1\}$$

Other e-voting protocols

Pure electronic voting protocols

- Civitas (both verifiable and coercion-resistant)
- Belenios (a ballot-stuffing resistant variant of Helios)
- Norwegian protocol (developed by Scytl)
- FOO, ...

Hybrid systems

- Pret à voter
- Scantegrity
- ...

How to model vote privacy ?

How to state formally :

"No one should know my vote (0 or 1)" ?

Idea 1 : An attacker should not learn the value of my vote.

How to model vote privacy ?

How to state formally :

"No one should know my vote (0 or 1)" ?

Idea 1 : An attacker should not learn the value of my vote.
But everyone knows 0 and 1 !

How to model vote privacy ?

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

Idea 2 : An attacker cannot see the difference when voters are different $\text{Voter}(A, 0) \approx \text{Voter}(B, 0)$

How to model vote privacy ?

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

Idea 2 : An attacker cannot see the difference when voters are different $\text{Voter}(A, 0) \approx \text{Voter}(B, 0)$

Who voted might be public (*cf* Helios)

How to model vote privacy ?

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

~~Idea 2 : An attacker cannot see the difference when voters are different~~ $\text{Voter}(A, 0) \approx \text{Voter}(B, 0)$

Idea 3 : An attacker cannot see the difference when I vote 0 or 1.

$$\text{Voter}(A, 0) \approx \text{Voter}(A, 1)$$

How to model vote privacy ?

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

~~Idea 2 : An attacker cannot see the difference when voters are different~~ $\text{Voter}(A, 0) \approx \text{Voter}(B, 0)$

Idea 3 : An attacker cannot see the difference when I vote 0 or 1.

$$\text{Voter}(A, 0) \approx \text{Voter}(A, 1)$$

- The attacker always sees the difference since the tally differs.
- Unanimity does break privacy.

How to model vote privacy ?

How to state formally :

"No one should know my vote (0 or 1)" ?

~~Idea 1 : An attacker should not learn the value of my vote.~~

~~Idea 2 : An attacker cannot see the difference when voters are different~~ $\text{Voter}(A, 0) \approx \text{Voter}(B, 0)$

~~Idea 3 : An attacker cannot see the difference when I vote 0 or 1.~~

$$\text{Voter}(A, 0) \approx \text{Voter}(A, 1)$$

Idea 4 : An attacker cannot see when votes are swapped.

$$\text{Voter}(A, 0) \mid \text{Voter}(B, 1) \approx \text{Voter}(A, 1) \mid \text{Voter}(B, 0)$$

S. Kremer & M. Ryan

How to formalize end-to-end verifiability ?

- For any announced result r
- For all voters that believe their vote has been counted $\mathsf{VoterHappy}(id_1, v_1), \ldots, \mathsf{VoterHappy}(id_n, v_n)$

We have that $r = v_1 + \cdots + v_n + r'$
where r' corresponds to the votes casted by compromised voters.

$\longrightarrow$ Requires to count.

Difficulties when analysing e-voting protocols

Primitives

- homomorphic encryption
- blind signatures
- zero-knowledge proofs
- AC operators
- everything combined (example of the Norwegian protocol)

Properties

- vote privacy : requires equivalence-based properties
- verifiability : requires to count

What formal methods can do ?

Few tools for equivalence

- ProVerif : often needs to be combined with ProSwapper
  $\longrightarrow$ does not support AC properties in practice
- Some more prototypes tools : Akiss, APTE, SPEC
  $\longrightarrow$ limited in the equational theories they can handle in practice

$\longrightarrow$ No tool support for homomorphic encryption !

What formal methods can do ?

Few tools for equivalence

- ProVerif : often needs to be combined with ProSwapper
  $\longrightarrow$ does not support AC properties in practice
- Some more prototypes tools : Akiss, APTE, SPEC
  $\longrightarrow$ limited in the equational theories they can handle in practice

$\longrightarrow$ No tool support for homomorphic encryption !

Proofs by hand [CSF 2011, POST 2012]

- Helios
- Norwegian protocol

$\longrightarrow$ tedious and error-prone

Almost no proofs of verifiability.

Another approach : proof by typing

How to use type systems to prove security of e-voting

The special case of F^* and rF^*.

How to type End2end verifiability

- For any announced result r
- For all voters that believe their vote has been counted $\mathsf{VoterHappy}(id_1, v_1), \ldots, \mathsf{VoterHappy}(id_n, v_n)$

Then $r = v_1 + \cdots + v_n + r'$
where r' corresponds to the votes casted by compromised voters.

We split End2end verifiability into three (stronger) properties

- Individual verifiability
- Universal verifiability

How to type End2end verifiability

- For any announced result r
- For all voters that believe their vote has been counted $\mathsf{VoterHappy}(id_1, v_1), \ldots, \mathsf{VoterHappy}(id_n, v_n)$

Then $r = v_1 + \cdots + v_n + r'$
where r' corresponds to the votes casted by compromised voters.

We split End2end verifiability into three (stronger) properties

- Individual verifiability
- Universal verifiability
- No clash property

Individual Verifiability

$$\begin{aligned}\text{voter}(id, v) := \ & \text{assume Vote}(id, v) \\ & \quad \ldots \text{ let } b = \text{ in } \ldots \\ & \text{assume MyBallot}(id, v, b) \\ & \quad \ldots \text{ send}(\cdots) \ldots \text{ receive}(\cdots) \ldots \text{ check } \ldots \\ & \text{assert VoterHappy}(id, v, b, bb)\end{aligned}$$

Individual Verifiability

$$\begin{aligned}\text{voter}(id, v) := \ & \text{assume Vote}(id, v) \\ & \quad \ldots \text{ let } b = \text{ in } \ldots \\ & \text{assume MyBallot}(id, v, b) \\ & \quad \ldots \text{ send}(\cdots) \ldots \text{ receive}(\cdots) \ldots \text{ check } \ldots \\ & \text{assert VoterHappy}(id, v, b, bb)\end{aligned}$$

$$\text{VoterHappy}(id, v, b, BB) := Vote(id, v) \wedge \exists b \in bb.\text{MyBallot}(id, v, b)$$

Universal Verifiability

"If the judge is happy,
the result corresponds to the ballots on the board"

$$\mathsf{JudgeHappy}(bb, r) :=$$
$$\exists vbb.(\mathsf{GoodSanitization}(bb, vbb) \wedge \mathsf{GoodCounting}(vbb, r))$$

Universal Verifiability

"If the judge is happy,
the result corresponds to the ballots on the board"

$\mathsf{JudgeHappy}(bb, r) :=$
$\exists vbb.(\mathsf{GoodSanitization}(bb, vbb) \wedge \mathsf{GoodCounting}(vbb, r))$

$\mathsf{GoodCounting}(vbb, r) :=$
$vbb =_m \{\mathsf{Wrap}(v_1), \ldots, \mathsf{Wrap}(v_n)\}$
$r = v_1 + \cdots + v_n$

$\mathsf{GoodSanitization}(bb, vbb)$: no honest ballot has been removed.

Universal Verifiability

"If the judge is happy,
the result corresponds to the ballots on the board"

$\mathsf{JudgeHappy}(bb, r) :=$
$\exists vbb.(\mathsf{GoodSanitization}(bb, vbb) \wedge \mathsf{GoodCounting}(vbb, r))$

$\mathsf{GoodCounting}(vbb, r) :=$
$vbb =_m \{\mathsf{Wrap}(v_1), \ldots, \mathsf{Wrap}(v_n)\}$
$r = v_1 + \cdots + v_n$

$\mathsf{GoodSanitization}(bb, vbb)$: no honest ballot has been removed.

Theorem

Individual verifiability, universal verifiability and no clash entail end-to-end verifiability

Privacy

Theorem (POPL'14, instantiated to vote privacy)

Let $P = fun(vA, vB) \rightarrow System[Alice(vA), Bob(vB)]$.

If $\emptyset \vdash P(\{\langle Lv_1, Rv_2\rangle\}, \{\langle Lv_2, Rv_1\rangle\}) \rightsquigarrow I_{eq}$ *then*

$$P(v_1, v_2) \approx P(v_2, v_1)$$

Privacy

Theorem (POPL'14, instantiated to vote privacy)

Let $P = fun(vA, vB) \rightarrow System[Alice(vA), Bob(vB)]$.

If $\emptyset \vdash P(\{\langle Lv_1, Rv_2\rangle\}, \{\langle Lv_2, Rv_1\rangle\}) \rightsquigarrow I_{eq}$ *then*

$$P(v_1, v_2) \approx P(v_2, v_1)$$

Our contribution : Design of a sealed-based library for voting

- homomorphic encryption
- proofs of statements such as $a + b = b + a$ are discharged to Z3

Application to Helios

Several variants of Helios have been analyzed automatically

- homomorphic encryption
- mixnet tallying

for both verifiability and privacy.

Future work

More properties

- eligibility verifiability
- coercion resistance

More primitives

- zero-knowledge proofs
- blind signatures
- theory of the Norwegian protocol

More protocols

- Norwegian protocol
- Civitas