Engineering Secure Software Systems

skipped in Winter 2020/21: Voting Protocols

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

Part I: Crypto Protocols

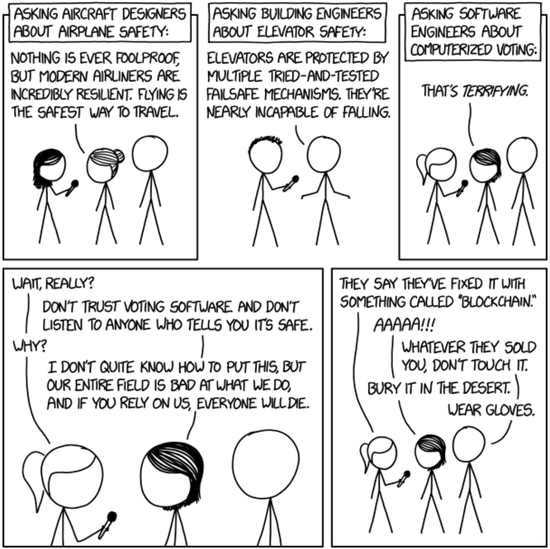
Overview

Electronic Voting

state of the field

- active research area
- interesting protocols and crypto primitives
- application / development of techniques for protocols
- real-life applications?

XKCD 2030: Voting Software



There are lots of very smart people doing fascinating work on cryptographic voting protocols. We should be funding and encouraging them, and doing all our elections with paper ballots until everyone currently working in that field has retired.

But …

academic application

- Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. “BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme”. In: ACM Conference on Computer and Communications Security. ACM, 2016, pp. 1614–1625
- Véronique Cortier, Constantin Catalin Dragan, François Dupressoir, and Bogdan Warinschi. “Machine-Checked Proofs for Electronic Voting: Privacy and Verifiability for Belenios”. In: 31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018. IEEE Computer Society, 2018, pp. 298–312. ISBN: 978-1-5386-6680-7. DOI: 10.1109/CSF.2018.00029. URL: https://doi.org/10.1109/CSF.2018.00029

Voting in Computer Science

here: security

count votes with correctness, privacy

- insecure network
- active adversary
- tradeoff verifiability/privacy
- manipulation: interference with counting process
- ...

not an exhaustive list!

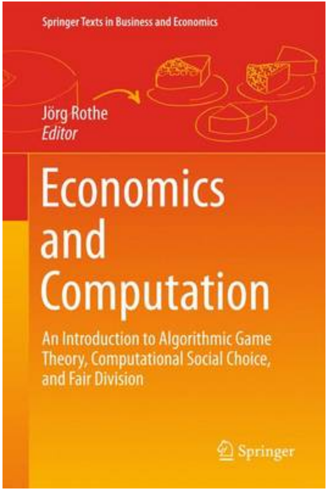
AI: computational social choice

given votes, determine election result

- what is a “fair” way to choose a winner?
- how do we stop people from voting “insincerely”?
A	10 votes
B	10 votes
C	3 votes

  preference $C > B$, vote “honestly”?
- impossibility results
- computational complexity as “solution”

Jörg Rothe, ed. Economics and Computation, An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division. Springer, 2016. ISBN: 978-3-662-47903-2. DOI: 10.1007/978-3-662-47904-9. URL: https://doi.org/10.1007/978-3-662-47904-9



Off-Topic: Voting Machines I

Brazilian voting machine by Diebold *source: Wikipedia*

Off-Topic: Voting Machines II

easy manipulation

"easy" exchange of software (60 sec)

bugs

uncounted votes

democratic control?

verify election integrity?

source: Spiegel Online

Analysis of “Realistic” Protocols

case studies

analysis of two voting protocol

motivation

- complex protocol
- complex security goals
- non-standard cryptographic primitives
- non-trivial modeling: no “push button” solution
- limits of automatic analysis

references

- Véronique Cortier. “Electronic Voting: How Logic Can Help”. In: IJCAR. 2014, pp. 16–25
- Ralf Küsters and Johannes Müller. “Cryptographic Security Analysis of E-voting Systems: Achievements, Misconceptions, and Limitations”. In: Electronic Voting - Second International Joint Conference. Ed. by Robert Krimmer, Melanie Volkamer, Nadja Braun Binder, Norbert Kersting, Olivier Pereira, and Carsten Schürmann. Vol. 10615. Lecture Notes in Computer Science. Springer, 2017, pp. 21–41. ISBN: 978-3-319-68686-8. DOI: 10.1007/978-3-319-68687-5. URL: https://doi.org/10.1007/978-3-319-68687-5

Voting Protocols in Lecture: Two Examples

Norwegian voting protocol

- practically relevant
- complex, modern protocol
- non-standard primitives
- shows limitations of model, analysis techniques

FOO92 protocol

- academic protocol
- simpler (and older)
- automatic analysis possible (to a point)

Voting Protocols

parties

chair performs, oversees election

voter votes exactly once

candidate not active in election

protocol structure (typical)

1. voter registration
2. transfer of (encrypted) votes
3. evaluation (counting) of votes
4. announcement of election result

trust?

- election chair honest?
- other voters honest?
- “external” attacker?
- corrupted voter’s hardware?
- minimal honesty assumption?

Voting Protocols Security Goals

$\approx$ authentication

- only "allowed" voters (Alice, Bob, ...) may vote — eligibility
- every voter may vote at most once — eligibility

$\approx$ (strong) secrecy

- Alice's vote remains secret — privacy
- Bob does not learn anything about result before he votes — fairness

verification

- Alice can check that her vote was counted correctly — verifiability

privacy "against" voter

- Alice cannot prove to Charlie how she voted — receipt-freeness

Voting Protocols in Politics

Norway example

- voting protocol used in nationwide elections
- 28 000 voters in 2011
- 70 000 voters in 2013
- use suspended in 2014: *Voters' fears about their votes becoming public could undermine democratic processes [CNN14].*

reference (presentation, images, analysis)

- Véronique Cortier and Cyrille Wiedling. "A formal analysis of the Norwegian E-voting protocol". In: Journal of Computer Security 25.1 (2017), pp. 21–57. DOI: 10.3233/JCS-15777. URL: https://doi.org/10.3233/JCS-15777
- Kristian Gjøsteen. "Analysis of an internet voting protocol". In: IACR Cryptology ePrint Archive 2010 (2010), p. 380. URL: http://eprint.iacr.org/2010/380

Protocol Participants

overview

- V voter
- P voter's computer
- R receipt generator
- D decryption service
- A auditor

interactions

- V interacts with P, verifies out-of-band feedback
- P communicates with B
- B receives ballots from P, sends them to R and D
- R sends receipts to V (out-of-band and via B/P)
- A receives information from all infrastructure players

Norway Protocol: Voting Process

voter (V)

- has list of receipt codes $d_V(f(o)^{s_V})$ for each candidate o
- instructs computer to vote for candidate o and sign ballot
- receives
  - acceptance message from computer
  - out-of-band message from authority
- checks matching
- ensures "cast-as-intended" property

voter's computer (P)

- encrypts ballot with public ElGamal key
- adds proof that vote is for "valid candidate"
- sends message to ballot box
- waits for confirmation (signed hash of ballot)
- notifies user

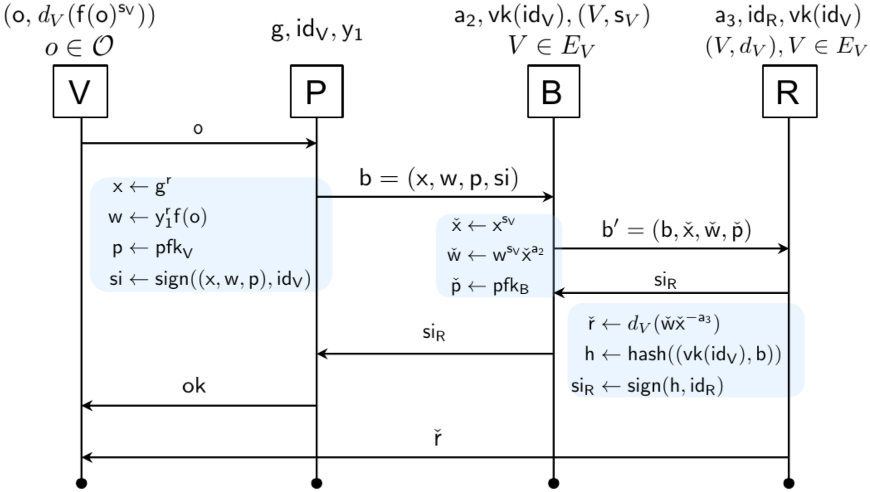
Norway Protocol: Voting Process

ballot box (B)

- waits for encrypted, signed ballot from P
- checks signatures, proofs
- re-encrypts and blinds ballot
- generates proof for computation correctness
- sends message to receipt generator (R)
- waits for signed message from R, forward to P

receipt generator (R)

- receives and verifies encrypted ballot from B
- generates receipt:
  - receipt code r using voter-specific function d_V, sent out-of-band to voter
  - signature of hash of encrypted ballot, sent to P (via B)


$(o, d_V(f(o)^{s_V}))$
$o \in \mathcal{O}$
g, id_V, y_1
$a_2, vk(id_V), (V, s_V)$
$V \in E_V$
$a_3, id_R, vk(id_V)$
$(V, d_V), V \in E_V$
V
P
B
R
o
$x \leftarrow g^r$
$w \leftarrow y_1^r f(o)$
$p \leftarrow pfk_V$
$si \leftarrow sign((x, w, p), id_V)$
$b = (x, w, p, si)$
$\check{x} \leftarrow x^{s_V}$
$\check{w} \leftarrow w^{s_V}\check{x}^{a_2}$
$\check{p} \leftarrow pfk_B$
$b' = (b, \check{x}, \check{w}, \check{p})$
si_R
$\check{r} \leftarrow d_V(\check{w}\check{x}^{-a_3})$
$h \leftarrow hash((vk(id_V), b))$
$si_R \leftarrow sign(h, id_R)$
si_R
ok
$\check{r}$

key aspects

- verifiability: voter can check that her vote has been counted correctly
    - trace-property: reachability in model checking game
    - conceptually similar to secrecy, authentication
    - use **event**, be careful
    - issues?
- privacy: voter's vote must remain secret
    - derivability not enough
    - strong secrecy required
    - indistinguishability proof
    - more involved modeling, algorithms
    - see concrete analysis later

Norway Protocol: Modeling in ProVerif

rewriting system

- ElGamal
- re-encryption
  - keys: $k_3 = k_1 + k_2$
- blinding
- signatures
- zero knowledge proofs
- associativity, commutativity

proofs

- ballot secrecy
- two corruption scenarios:
  1. authorities honest, all but two voters corrupt
  2. ballot box corrupt, all but two voters corrupt
- manual proof in symbolic model

complexity

- primitives outside of our term model
- cannot be handled by ProVerif or other state-of-the-art tools

FOO92 protocol

- academic protocol
- uses simpler cryptographic primivites
- allows automatic formal analysis in ProVerif

reference

Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. "A Practical Secret Voting Scheme for Large Scale Elections.". In: AUSCRYPT. 1992, pp. 244–251

Cryptographic Primitives

standard

- signatures: sskey, pskey, sign, checksign
- anonymous channels: standard in ProVerif, difficult in practice (Tor, etc.)

commitment

type comkey.

fun commit(comkey, bitstring): bitstring.

reduc forall m:bitstring, k:comkey; open(commit(m,k),k)=m.

blind signature scheme

type blikey.

fun blind(blikey, bitstring): bitstring.

reduc forall unblind(kb, blind(m,kb))=m.

reduc forall unblind(kb, sign(ks, blind(kb, m)))=sign(ks,m)

note: no subterm theory!

FOO92 Protocol Overview

phases

1. legitimization
   - votes "registered" by election authority
   - legitimizes **votes**, not voters
2. voting
   - voters send votes to "collector" instance
3. result announcement
   - chair publishes election result

crucial

- phase i starts only after phase $i - 1$ completed
- ProVerif: **phase** command

FOO92 Phase 1: legitimization

voter A		chair E
vote v		
random key k_{com}		
$x = \text{commit}(k_{com}, v)$		
random key k_{bli}		
$e = \text{blind}(k_{bli}, x)$		
	$\xrightarrow{\text{sig}_{k_A}(e)}$	
		checks that A may vote, has not voted yet
	$\xleftarrow{\text{sig}_{k_E}(e)}$	

$$
\begin{aligned}
y &= \text{unblind}(k_{bli}, \text{sig}_{k_E}(e)) \\
  &= \text{sig}_{k_E}(\text{commit}(k_{com}, v))
\end{aligned}
$$

result phase 1

A has “secret” vote signed by E, vote can only be “opened” with Alice’s secret key k_{com}

A cannot change vote later (commitment)

FOO92 Phase 2: Voting

voter A	chair E

$$y = \mathrm{sig}_{k_E}\left(\mathrm{commit}(k_{com}, v)\right)$$
$$\xrightarrow{\hspace{8em}}$$

add $(\mathrm{commit}(k_{com}, v), y)$ to list L

result phase 2

- chair has list of commitments
- every commitment signed by E
- list L may be published only after phase 2 completed

FOO92 Phase 3: Announcement

voter A		chair E
		publishes list L
finds own commitment at position l in L		
	$\xrightarrow{(l, k_{com})}$	
		opens entry l publishes vote

Alice's last two messages: anynomous!

“New” Primitives in FOO92

how does the protocol use

- commitment?
- blinding?

ProVerif modeling

identical for symmetric encryption and commitment

note from real life

commitment has different properties than encryption

- (realistic) encryption
  - $\mathsf{enc}^s_k(t) = \mathsf{enc}^s_{k'}(t')$ possible for $(k,t) \neq (k',t')$
  - can be avoided by checksums
- commitment: value $\mathsf{commit}(k_{com}, v)$ must **uniquely** fix v
- $\mathsf{commit}(k_{com}, t) = k_{com} \oplus t$ completely insecure as commitment
- more generally: possibilistically (or information-theoretically) secure encryption cannot be used

Blind Signatures

application

- Alice wants "certificate" that she sent m to Bob (e.g., before deadline)
- Bon must not learn m
  - possibility: let Bob sign $\textbf{hash}(m)$
  - issue? when Bob learns m later, he can determine m came from Alice
  - Alice wants to **anonymously** reveal m later

specification blind signatures

reduc forall
unblind(kb, blind(m,kb))=m.

reduc forall
unblind(kb, sign(ks, blind(kb, m)))=sign(ks,m)

steps

- Alice sends $\textbf{sig}_{k_A}\left(\overbrace{\textbf{blind}(k_{blind}, m)}^{=x}\right)$ to Bob
- Bob sends $y = \textbf{sig}_{k_B}(x)$ to Alice
- Alice reveals (anonymously) $\textbf{sig}_{k_B}(m) = \textbf{unblind}(k_{blind}, \underbrace{\textbf{sig}_{k_B}(\textbf{blind}(k_{blind}, m))}_{=y})$

Exercise

Task (blind signatures)

As noted in the lecture, the given equational theory for blind signatures is no subterm theory, which leads to difficulties in automatic protocol analysis. Is there a way to rewrite protocols using blind signatures using only standard cryptographic primitives? Does such a rewrite give the same security guarantees as intended by the primitive?

Voting Protocols Security Properties

central goal: privacy

- information about Alice's vote must remain secret
- but: election result is published

privacy impossible if …

- one party receives all / no votes
- attacker knows all votes except for Alice's
  - attacker learns how dishonest voters voted
  - assume at least two "honest" voters

approach

- attacker does not learn more about Alice's vote than what can be deduced from election result
- attacker cannot distinguish:
  1. actual election
  2. election obtained by permutation of honest votes

Modeling Privacy in Voting Protocols

assumption

- honest voters Alice and Bob, votes must remain secret
- attacker learns about votes via election result

definition

voting protocol satisfies **privacy**, if indistinguishable:

1. real election
2. election with Alice's and Bob's votes swapped

Alice's / Bob's **vote**: candidate (s)he votes for

indistinguishability

- term level: static equivalence via tests
- extension to processes
- ProVerif: **choice**

Exercise

Task (voting protocols)

In the lecture, several security properties for voting protocols were discussed. Design a simple voting protocol that satisfies at least one of these properties. Use ProVerif to show that the property is in fact satisfied.

Modeling the FOO92 Protocol in ProVerif

reference

Steve Kremer and Mark Ryan. "Analysis of an Electronic Voting Protocol in the Applied Pi Calculus". In: ESOP 2005. Ed. by Shmuel Sagiv. Vol. 3444. Lecture Notes in Computer Science. Springer, 2005, pp. 186–200. ISBN: 3-540-25435-8

processes

1. voter
2. election chair (administrator)
3. election chair (collector)
4. main process (environment)

FOO92 Voter Process

sequence

- generate keys for blinding and commitment
- vote *v*: not as nonce (bitstring)
  - “weak” secret, since small set of possible values
  - defined in main process
- follows protocol specification
- synchronisation using **phase** commands

```
let processVoter =
    new blinder: blikey.
    new r: comkey.
    let blindedcommitedvote = blind(commit(v,r), blinder) in
        out (ch, (hostv, sign(blindedcommitedvote, skv)));
        in (ch, m2 : bitstring);
        let blindedcommitedvote0 = checksign(m2,pka) in
            if blindedcommitedvote0 = blindedcommitedvote then
                let signedcommitedvote=unblind(m2, blinder) in
                    phase 1;
                    out (ch, signedcommitedvote)
                    in (ch, l:bitstring, =signedcommitedvote)
                    phase 2;
                    out (ch, (l,r)).
```

FOO92 Election Chair Process A

separation election chair

- administration (A)
- collector

task: check voters, sign votes

- receives voter's public keys over private channel
- receives voter's vote, checks signature
- (blindly) sign votes

FOO92 Election Chair Process A

```
let processA =
    in (privCh, pubkv);
    in (ch, m1:bitstring);
    let (hv, sig)=m1 in
        let pubkeyv=getpk(hv) in
            if pubkeyv=pubkv then
                out (ch,sign(checksign(sig,pubkeyv), ska)).
```

FOO92 Collector Process

separation election chair

- administration (A)
- collector

tasks

- “book-keeping:” collects successfully received votes
- for every vote: new index in global list
- expects key to open commitment
- does not have a private key: verifiable!

FOO92 Collector Process

```
let processCollector =
    phase 1;
    in (c,m3:bitstring);
    new l:bitstring;
    out ch, (l,m3);
    phase 2;
    in (ch,(=l, rand))
    let voteV=open(checksign(m3,pka), rand) in
        out (ch,voteV).
```

note

- collector does not wait for every voter's commitment
- must be done in real implementation

FOO92 Main Process

tasks

- key generation and distribution
  - secret keys for voters
    - simplification: all honest voters share private/public key! issue?
  - secret key for election chair
  - send public keys on public channel
  - honest voters' public key to admin process
    - models voter registration
  - register adversary-controlled voters
- starting processes
  - unbounded number of "server process" instances

observe

main process $\neq$ election chair

- voters' private keys remain secret

FOO92 Main Process

```
process
    new ska, skv: sskey.
    new ch: channel.
    new privCh: channel [private].
    let pka=pk(ska) in
        let hosta=host(pka) in
            let hostv=host(pkv) in
                out (ch, pka);
                out (ch, hosta);
                out (ch, pkv);
                out (ch, hostv);
                out (privCh, pkv);
                out (privCh, pk(ski));
                (!processV) | (!processA) | (!processC);
```

FOO92 Modeling Summary

protocol modeling

- relatively straightforward
- what did we leave out?

examples

- PKI
  - as always, we abstract this away
- identities: only single voter identity
  - simplifies authentication, possibly verifiability
- verifiability: Alice waits for “indexed element”
  - assumes secure transmission, what if somebody “fakes” Alice’s view?
- ...

Specification of Security Properties

analyzed

1. fairness
2. eligibility (somewhat)
    - all honest voters share a key
3. privacy
    - manual steps required

summary

- possible in theory
- difficulties in practice
  - trusted hardware needed
  - security issues
  - social/legal aspects

verification difficulties

- complex protocols and security goals
- non-standard cryptographic primitives
- equational theories currently out of scope for automatic analysis
- verification of older protocols (FOO92) possible
- modeling in ProVerif intricate

further issues

- relationship between properties (privacy, coercion resistance)
- trade-off simplicity / security
- new attacks: “clash attacks”

References i

Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. "BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme". In: ACM Conference on Computer and Communications Security. ACM, 2016, pp. 1614–1625.

CNN. E-voting experiments end in Norway amid security fears. June 2014. URL: http://www.bbc.com/news/technology-28055678.

Véronique Cortier, Constantin Catalin Dragan, François Dupressoir, and Bogdan Warinschi. "Machine-Checked Proofs for Electronic Voting: Privacy and Verifiability for Belenios". In: 31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018. IEEE Computer Society, 2018, pp. 298–312. ISBN: 978-1-5386-6680-7. DOI: 10.1109/CSF.2018.00029. URL: https://doi.org/10.1109/CSF.2018.00029.

Véronique Cortier. "Electronic Voting: How Logic Can Help". In: IJCAR. 2014, pp. 16–25.

- Véronique Cortier and Cyrille Wiedling. “A formal analysis of the Norwegian E-voting protocol”. In: Journal of Computer Security 25.1 (2017), pp. 21–57. DOI: 10.3233/JCS-15777. URL: https://doi.org/10.3233/JCS-15777.
- Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. “A Practical Secret Voting Scheme for Large Scale Elections.”. In: AUSCRYPT. 1992, pp. 244–251.
- Kristian Gjøsteen. “Analysis of an internet voting protocol”. In: IACR Cryptology ePrint Archive 2010 (2010), p. 380. URL: http://eprint.iacr.org/2010/380.

- Ralf Küsters and Johannes Müller. “Cryptographic Security Analysis of E-voting Systems: Achievements, Misconceptions, and Limitations”. In: Electronic Voting - Second International Joint Conference. Ed. by Robert Krimmer, Melanie Volkamer, Nadja Braun Binder, Norbert Kersting, Olivier Pereira, and Carsten Schürmann. Vol. 10615. Lecture Notes in Computer Science. Springer, 2017, pp. 21–41. ISBN: 978-3-319-68686-8. DOI: 10.1007/978-3-319-68687-5. URL: https://doi.org/10.1007/978-3-319-68687-5.
- Steve Kremer and Mark Ryan. “Analysis of an Electronic Voting Protocol in the Applied Pi Calculus”. In: ESOP 2005. Ed. by Shmuel Sagiv. Vol. 3444. Lecture Notes in Computer Science. Springer, 2005, pp. 186–200. ISBN: 3-540-25435-8.

Jörg Rothe, ed. Economics and Computation, An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division. Springer, 2016. ISBN: 978-3-662-47903-2. DOI: 10.1007/978-3-662-47904-9. URL: https://doi.org/10.1007/978-3-662-47904-9.