Electronic Voting Protocol Analysis with the Inductive Method

Denis Butin (DCU) and Giampalo Bella (UniCT)

Introduction

- E-voting use is spreading quickly in the EU and elsewhere
- Sensitive, need for formal guarantees
- Inductive Method: protocol verification through theorem proving + mathematical induction
- Toolbox being built with FOO as dummy protocol
- Goal: make all properties rigourously verifiable

Motivation

- Analysis of e-voting dominated by ProVerif automatic verifier
- Powerful, but sometimes limited
- Motivation to fill in the gaps with complementary, alternative approach

E-voting protocols: primitives

- Often require modelling new crypto primitives
- Blind signatures
- Bit commitment
- Proxy re-encryption . . .

Related Work

- Ryan / Kremer / Delaune: applied pi calculus, partially mechanized through ProVerif
- Observational equivalence: traces in which two voters swap their votes are equivalent in a sense
- Parts of the proof done by hand

Method: the Inductive approach

- Mathematical induction on protocol steps
- Dolev-Yao threat model
- Tool support: Isabelle/HOL interactive theorem prover

FOO'92

- "A practical secret voting scheme for large scale elections", AUSCRYPT 1992
- By Fujioka, Okamoto and Ohta
- Claims four classical properties

Properties of FOO

- Fairness: Partial results confidential as long as the voting phase is ongoing
- Eligibility: Only registered voters can vote, and only once
- Individual verifiability: Voters can check their vote was counted
- Privacy: How a particular voter voted is not known to anyone

Properties of FOO – in practice

- Fairness – tally confidentiality before deadline
- Eligibility – authentication + uniqueness check
- Individual verifiability – Event implication check
- Privacy – LINKABILITY concept (hard), our focus here

Steps of FOO (1/2)

In a nutshell:

- Voter picks a vote and sends a signed, blinded commitment of it with its identity to Administrator
- Administrator checks this and returns it with own (blind) signature if approved
- V unblinds this and sends it to Collector anonymously
- Collector checks what he receives and records it on a list if correct
- Deadline

Steps of FOO (2/2)

- Once the deadline is reached, Collector publishes list of commitments
- If V's commitment is in list, V discloses secret commitment key anonymously
- Collector opens V's ballot and publishes it

Steps of FOO – modelled

```
 | Unblinding:
   "⟦evsb ∈ foo; Crypt (priSK V) BSBody ∈ analz (spies evsb); BSBody = Crypt b (Crypt c (Nonce N));
    b ∈ symKeys; Key b ∈ analz (spies evsb)⟧  ⟹ Notes Spy (Crypt (priSK V) (Crypt c (Nonce N))) # evsb ∈ foo"

(* Preparation *)
 | EV1:
   "⟦evs1 ∈ foo; V ≠ Adm; V ≠ Col; c ∈ symKeys; Key c ∉ used evs1; b ∈ symKeys; Key b ∉ used evs1; b ≠c; Nonce Nv ∉ used evs1⟧
    ⟹ Says V Adm ⦃Agent V, Crypt (priSK V) (Crypt b (Crypt c (Nonce Nv)))⦄  # Notes V (Key c) # Notes V (Key b) # evs1 ∈ foo"

(* Administration *)
| EV2:
   "⟦evs2 ∈ foo; V ≠ Adm; V ≠ Col; Gets Adm ⦃Agent V, Crypt (priSK V) BSBody⦄ ∈ set evs2;
    BSBody = Crypt P R; ∀ X Y. MPair X Y ∉ parts{BSBody}; (* Only accept ciphertext of specified length *)
    Notes Adm (Agent V) ∉ set evs2⟧ ⟹ Says Adm V (Crypt (priSK Adm) BSBody) # Notes Adm (Agent V) # evs2 ∈ foo"

(* Voting -- anonymous channel *)
 | EV3:
   "⟦evs3 ∈ foo; Says V Adm ⦃Agent V, Crypt (priSK V) (Crypt b (Crypt c (Nonce Nv)))⦄ ∈ set evs3;
    Gets V (Crypt (priSK Adm) (Crypt b (Crypt c (Nonce Nv)))) ∈ set evs3⟧
    ⟹ Anms V Col (Crypt (priSK Adm) (Crypt c (Nonce Nv))) # evs3 ∈ foo"

(* Collecting *)
| EV4:
   "⟦evs4 ∈ foo; V ≠ Adm; V ≠ Col; Gets Col ⦃Number anms, Crypt (priSK Adm) CX⦄ ∈ set evs4;
    CX = Crypt P R; ∀ X Y. MPair X Y ∉ parts{CX}; Says Col Col CX ∉ set evs4⟧ ⟹ Says Col Col CX # evs4 ∈ foo"

(* Opening -- anonymous channel *)
 | EV5:
   "⟦evs5 ∈ foo;
    Says V Adm ⦃Agent V, Crypt (priSK V) (Crypt b (Crypt c (Nonce Nv)))⦄ ∈ set evs5; Gets Col (Crypt c (Nonce Nv)) ∈ set evs5;
    Key c ∈ analz (knows V evs5); c ∉ range shrK; c ∈ symKeys⟧ ⟹ Anms V Col (Key c) # evs5 ∈ foo"

(* Counting *)
| EV6:
   "⟦evs6 ∈ foo; Gets Col ⦃Number anms, Key c⦄ ∈ set evs6; Gets Col (Crypt c (Nonce Nv)) ∈ set evs6; Says Col Col (Nonce Nv) ∉ set evs6⟧ ⟹
   Says Col Col (Nonce Nv) (* Counter annonces results *) # evs6 ∈ foo"
```

What is privacy in e-voting?

- Crucial point: privacy is NOT confidentiality of vote ...
- ... But unlinkability of voter and vote
- In Pro-Verif, done with observational equivalence between swapped votes

Privacy in the Inductive Method: aanalz

```
primrec aanalz :: "agent => event list => msg set set"
where
  aanalz_Nil:   "aanalz A [] = {}"
| aanalz_Cons:
"aanalz A (ev # evs) =
 (if A = Spy then
  (case ev of
    Says A' B X ⇒
     (if A' ∈ bad  then aanalz Spy evs
      else if isAnms X
           then insert              ({Agent B} ∪ (analzplus {X} (analz(knows Spy evs)))) (aanalz Spy evs)
           else insert ({Agent A'} Un {Agent B} ∪ (analzplus {X} (analz(knows Spy evs)))) (aanalz Spy evs)
 )
  | Gets A' X ⇒ aanalz Spy evs
  | Notes A' X ⇒ aanalz Spy evs)
  else aanalz A evs)"
```

Extract associations from honest agent's messages

Privacy in the Inductive Method: asynth

```
inductive_set
 asynth :: "msg set set ⇒ msg set set"
 for as :: "msg set set"
 where
  asynth_Build [intro]: "⟦a1 ∈ as; a2 ∈ as; m ∈ a1; m ∈ a2; m ≠ Agent Adm; m ≠ Agent Col⟧ ⟹
                         a1 ∪ a2 ∈ asynth as"
```

Build up association sets from associations with common elements

Privacy in the Inductive Method: theorem statement

```
theorem foo_V_privacy_asynth:
"⟦Says V Adm {|Agent V, Crypt (priSK V) (Crypt b (Crypt c (Nonce Nv)))|} ∈ set evs;
  a ∈ (asynth (aanalz Spy evs));
  Nonce Nv ∈ a; V ∉ bad; V ≠ Adm; V ≠ Col; evs ∈ foo⟧
  ⟹ Agent V ∉ a"
```

If a normal voter started the protocol, the corresponding vote & identity cannot be associated

Summary

- E-voting protocol analysis field active, yet room for improvement
- Inductive Method's flexiblity allows new e-voting analysis
- Privacy: toughest part – crucial choices, ongoing

Future Work

- Complete verification toolbox with missing property formalisations
- Model & analyse real-world e-voting protocols
- Derive general design guidelines

Principles of the inductive method

- Number of agents is unbounded, session interleaving is allowed: replay attack weakness detected
- Cryptographic keys: type *key*, different subtypes for private / public / encryption / signature
- Events: *Says* (models sending), *Gets* (reception), *Notes* (knowledge)
- Trace: history of network events. Inductive reasoning over traces.
- Focus is *not* security of *algorithms*: treated as black boxes in Isabelle

Message set operators

- Fundamental operators, constantly used in security statements
- `parts`: decompose into atomic message components, even ciphertext for which decrypting key unavailable
- `analz`: like parts, but leaving undecryptable ciphertext untouched
- `synth`: build up messages from message components. Includes encryption if encrypting key available

Formal protocol model

- Every protocol step modeled as inductive rule with pre- and postconditions
- Protocol model is set of all admissible traces under those rules
- Empty trace modeled by *Nil* event
- Threat model (DY) represented by *Fake* event
- Agents' knowledge derived from traces