VERIFYING PRIVACY BY LITTLE INTERACTION AND NO PROCESS EQUIVALENCE

Denis Butin* [1] and Giampaolo Bella[2]
[1]*School of Computing, Dublin City University, Ireland*
[2]*Dipartimento di Matematica e Informatica, Università di Catania, Italy*
Software Technology Research Laboratory, De Montfort University, UK
denis.butin@computing.dcu.ie, giamp@dmi.unict.it

Keywords: E-voting, Privacy, Inductive Method

Abstract: While machine-assisted verification of classical security goals such as confidentiality and authentication is well-established, it is less mature for recent ones. Electronic voting protocols claim properties such as voter privacy. The most common modelling involves indistinguishability, and is specified via trace equivalence in cryptographic extensions of process calculi. However, it has shown restrictions. We describe a novel model, based on unlinkability between two pieces of information. Specifying it as an extension to the Inductive Method allows us to establish voter privacy without the need for approximation or session bounding. The two models and their latest specifications are contrasted.

1 INTRODUCTION

Formal analysis of security protocols is even more relevant as electronic voting (e-voting) starts being used for official elections across the world. The novel properties claimed by such protocols require new models and tools, especially when computer-aided analysis is desired. A key objective of e-voting protocols, called voter privacy or ballot secrecy, states that the way a particular voter voted is not known to the adversary. While the vote itself obviously becomes public in the final stage, it is the link between voter and ballot that must remain confidential. Other goals than voter privacy are also commonly studied for e-voting protocols, but we focus on voter privacy in this work.

The most notable efforts in the area, which are rather recent, are based upon a widely-accepted model, process equivalence. These have advanced computer-aided analysis of e-voting protocols significantly, albeit all existing tools face some limitations. The general idea of using equivalences in process calculi was presented as early as in 1996 (Schneider and Sidiropoulos, 1996), concurrently with the Inductive Method's inception (Paulson, 1998). This position paper makes the point that also the latter can be used profitably to verify e-voting protocols without inherent limitations about size and features, hence offering a valuable parallel means of analysis. The analyst gets strong assistance in his task — for example in keeping a clear picture of the various scenarios that the execution of the protocol may entail, over which the goals must be assessed.

However, interaction is only significant during proof development, that is, when a line of reasoning is being investigated for the first time. With automatic tools, this effort usually results in the implementation of extended analysis routines that can later be invoked and run fully automatically. Similarly, although theorem proving remains interactive, vast portions of the proof scripts can be re-used with minor adaptations over new case studies, when these require the same line of reasoning. For example, once confidentiality proofs were developed (Paulson, 1998), the corresponding scripts received negligible modifications throughout subsequent applications. We argue that this feature of interactive tools has not been emphasised properly so far. Therefore, the effort of developing a proof for an innovative property could be comparable to that of expanding an automatic tool for the same reason. Proof reuse could be the right balance between automation and interaction.

After recalling the main existing efforts, our debate starts by comparing them with the mechanization of our model in the Inductive Method.

*Research supported by the SFI grant 08/RFP/CMS1347.

2 EXISTING MODEL AND TOOLS FOR PRIVACY ANALYSIS

Most existing papers on formal privacy modelling in the context of e-voting are based on the following indistinguishability criterion: it must not be possible to distinguish between a situation in which the honest voter V_a voted x and the honest voter V_b voted y, and one in which the converse is true. A cryptographic extension of a process calculus is then used, often the applied pi calculus. Two main tools build upon this model.

ProVerif The automatic ProVerif tool (Blanchet, 1998) uses an algorithm based on Horn clauses. It is able to efficiently check an under-approximation of (i.e., a stronger equivalence than) observational equivalence between processes in the applied pi calculus for an unbounded number of sessions (Blanchet et al., 2008). Because of the approximation, it lacks in precision and may produce false negatives — spurious attacks may be found. Supplementary equational theories for broader cryptographic primitives support are easy to add, but this may lead to non-termination in some cases. ProVerif was used to analyse — for a fixed number of voters — voter privacy in the Fujioka, Okamoto and Ohta (FOO) (Fujioka et al., 1993) protocol in (Delaune et al., 2009). FOO involves two election officials and uses blind signatures as well as bit commitments. The bisimulation reasoning was done by hand.

AKiSs More recently, the focus in the process equivalence approach has shifted from trying to prove observational equivalence to checking a weaker kind of equivalence: trace equivalence ($\approx_t$). In (Chadha et al., 2012), a new cryptographic process calculus is introduced alongside a novel procedure for checking equivalence. Specifically, under- and over-approximations of $\approx_t$ are introduced, the fine-grained trace equivalence $\approx_{ft}$ and the coarse trace equivalence $\approx_{ct}$. Over-approximation can prove protocols correct, and under-approximation can rule out flawed protocols. The calculus supports a broad variety of cryptographic primitives: all those that can be modelled in an *optimally reducing convergent rewrite system.* The procedure is limited to a bounded number of protocol sessions. For a class of processes called *determinate*, there is a coincidence between $\approx_{ct}$, $\approx_t$ and observational equivalence. Automation is provided by the AKiSs tool. The work also includes an analysis of voter privacy in FOO. However, since the model of FOO does not yield a determinate process, observational equivalence cannot be proved by this approach and the approximation $\approx_{ft}$ must be checked instead.

3 DISCUSSION

Precision Since ProVerif systematically uses under-approximations, it is not precise in general. While it will not deem a flawed protocol correct, it may fail to validate a correct protocol due to the detection of a false attack. For AKiSs, precision depends on the class of the process modelling the protocol. Those which can be modelled using determinate processes can be checked directly for observational equivalence and $\approx_t$ because of their coincidence with $\approx_{ct}$ in that class. On the other hand, some e-voting protocols, particularly those using phases like FOO, must be under-approximated by $\approx_{ft}$ because they do not lead to determinate processes. In that case, as for ProVerif, the risk of spurious attack detection subsists. Unlinkability in the Inductive Method is treated by inductive theorem proving, so there can be no false positives. When unlinkability cannot be established, attacks are not given explicitly but examination of the remaining subgoal allows the user to trace back the flaw to the problematic protocol step. This interactivity gives greater insight into the protocol's intricacies but requires more effort than the aforementioned tools. Another aspect of precision is session bounding. While the Inductive Method and ProVerif both support unlimited protocol sessions, the computational cost of AKiSs blows up exponentially when more sessions are interleaved.

Automation vs. interaction AKiSs is fully automated. Voter privacy in ProVerif was checked by hand in (Kremer and Ryan, 2005). An automatic verification in ProVerif was presented in (Delaune et al., 2008), but a translation algorithm is involved and its correctness is not formally proven. In (Delaune et al., 2009), a ProVerif privacy proof for a fixed number of voters is partially automated. The Inductive Method is mechanized in Isabelle/HOL, an interactive theorem-prover. As noted above, this requires user interaction at proof development time, whereas ProVerif and AKiSs require it at tool development time.

Termination Termination is guaranteed neither in ProVerif nor AKiSs. While ProVerif features a resolution method that ensures termination for a specific syntactic transformation of protocol models, this method is limited to secrecy and authentication properties(Blanchet and Podelski, 2005). Since the Inductive Method is based on induction, there is no termination issue.

Supported cryptographic primitives Associative/commutative (A/C) operators are currently supported by none of the tools, despite their usefulness for the modelling of protocols using exponentiation or XOR. Blind signatures, which are commonly used in e-voting protocols, are supported by all three tools compared in this paper. AKiSs supports a larger variety of cryptographic primitives than ProVerif and the current version of the Inductive Method. In (Chadha et al., 2012), Chadha et al. conjecture that all those which can be modelled in a rewrite system with a specific convergence property are supported. Notably, trapdoor commitments can be modelled. By contrast, supporting new cryptographic primitives in the Inductive Method requires modelling them through additional rules using existing primitives or changing the underlying framework. The latter option requires more work since specific assumptions about the interaction of operators are made in the theory *Message*, which is always imported. On the other hand, major modifications have been performed successfully, for instance in (Martina and Paulson, 2011), demonstrating the flexibility of the approach. New cryptographic primitives can be added easily to the applied pi calculus by devising new equational theories, but the resulting model may be beyond the scope of the tool's automatic analysis (Delaune et al., 2010).

Efficiency The Authors of (Chadha et al., 2012) state that the automated analysis of privacy for FOO requires a few minutes on a modern laptop. Loading and verifying all FOO theories in Isabelle/HOL takes a similar time. However, our FOO theory also establishes other useful security guarantees about the protocol. Another metric is the respective human effort necessary to formalise protocols and their goals before the actual analysis. This aspect remains to be quantified.

Synthesis Table 1 summarises the comparison but does not aim for exhaustiveness. The "large set" of cryptographic primitives supported by AKiSs is the set of those that can be modelled in an optimally reducing convergent rewrite system.

4 OUTLINE OF VOTER PRIVACY IN THE INDUCTIVE METHOD

The Inductive Method The Inductive Method (Bella, 2007; Paulson, 1998), specified in the interactive theorem-prover Isabelle/HOL, has never been applied to e-voting protocols until now. It models protocol steps and a Dolev-Yao adversary's actions as inductive rules. At its core are inductively defined message operators. Initially supported message components are agent names, guessable integers (numbers), nonces, hashes and ciphertexts. The *parts* operator takes as input a message and returns the set of all its components, including encrypted ones for which the key is not available. It is normally used to denote all elements appearing in network traffic. *analz* bears close resemblance to *parts*, but takes into account cryptography. Only ciphertext for which the corresponding decryption key is available yields the embedded plaintext. As opposed to *parts* and *analz*, the *synth* operator specifies message building rather than message deconstruction. Taking into account available cryptographic keys, it inductively defines the set of messages that can be built from an initial message set. Ciphertexts and message hashes can be generated from available messages. Agent names and guessable numbers can be synthesised ex nihilo.

Cryptographic keys in Isabelle have the type *key*. In the case of asymmetric keys, for encryption, key pairs are modelled by *priEK* and *pubEK*. For signatures, *priSK* and *pubSK*. The *invKey* function turns an asymmetric key into its associated reciprocal key, e.g. *invKey(priEK A) = pubEK A*.

Not only is the number of agents unbounded: every agent is also able to interleave any number of protocol sessions. Since proofs are carried out inductively, susceptibility to replay attacks is taken into account. Agents can be either the Server (a trusted agent which knows all shared keys and is never compromised), a friendly agent or the Spy, who embodies the threat model.

Three message events are available. The *Says* event models the sending of a message between agents. For instance, $Says\ A\ B\ \{Nonce\ Na\}$ means that agent *A* sends agent *B* a message consisting solely of the nonce *Na*. It is realistic to assume that not all messages reach their destination, hence message sending does not imply message reception in general. However, a sent message can be delivered as intended; this is modelled by the *Gets* events, which precisely represents the reception of a message by an agent. The *Notes* event models the addition of a message to an agent's knowledge, for later use. It can be seen as a private recording of information.

	ProVerif	AKiSs	Inductive Method
Precision	Precise for determinate processes, else under-approximation	Under-approximation	Precise for generalised association synthesis
Operation	Partial automation	Full automation	Interactive proving
Unbounded sessions	Yes	No	Yes
Systematic termination	No	No	Yes
Supported cryptographic primitives	Usual + blind signatures, bit commitments, proxy re-encryption	Large set conjectured	Usual + blind signatures, bit commitments

Table 1: Synthesis of characteristics of mechanized FOO privacy analyses

Every protocol step is modelled as an inductive rule, with preconditions and a postcondition. The protocol model is the set of all admissible traces built from those steps. The empty trace is also allowed, and modelled by the *Nil* event. Furthermore, a *Fake* event accounts for message forgery by the Spy, who builds and sends everything she can from components extracted from network traffic:

```
| Fake:
  ⟦evsf ∈ bankerb_gets;
   X ∈ synth (analz (knows Spy evsf))⟧
   ⟹ Says Spy B X # evsf ∈ bankerb_gets
```

Extensions for e-voting protocols The following extensions are protocol independent. Blind signatures can be modelled as an inductive rule using symmetric cryptography and agent knowledge:

```
| 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
```

New operators are added to the Inductive Method to make the treatment of voter privacy possible. One is *analzplus*: by building upon the traditional message set analyzer *analz*, it is empowered with an external message set from which extra decryption keys can be derived. This helps to define *aanalz*, whereby the attacker extracts all meaningful associations from protocol events and traces:

```
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 B} ∪ {Agent A'} ∪
                (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)
```

For instance, considering the event *Says A B X* where *A* is uncompromised and *evs* is a generic protocol trace,

```
aanalz (Spy Says A B X # evs) = insert
(Agent A ∪ Agent B ∪ analzplus X (analz (knows Spy evs)))
(aanalz Spy evs)
```

The association synthesiser *asynth* builds new associations out of existing ones that share a common element. The common element should not be one of the election officials' names since those appear in all protocol runs and are therefore linked to all voter identities. It is defined as follows:

```
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
```

Associations arising from a generic trace of the protocol model are examined using these operators. In particular, the set *asynth*(*aanalz Spy evs*) formalises all possible associations that the attacker can build out of active observation of the trace *evs*. Unlinkability of two pieces of information can then be specified by stating that they cannot belong to any association in this set at the same time.

Verifying privacy for FOO We now focus on the case study of the FOO protocol. The main privacy theorem is stated as follows: whenever the first step of the protocol was performed by a regular honest voter, any association containing her vote cannot also contain her name.

theorem *foo_V_privacy_asynth*:
$[\![$*Says V Adm* $\{\!|$*Agent V*,
Crypt (*priSK V*) (*Crypt b* (*Crypt c* (*Nonce Nv*)))$|\!\}$ $\in$ *set evs;*
$a \in$ (*asynth* (*aanalz Spy evs*));
Nonce Nv $\in a$; $V \notin$ *bad*; $V \neq$ *Adm*; $V \neq$ *Col*; *evs* $\in$ *foo*$]\!]$
$\Longrightarrow$ *Agent V* $\notin a$

Although this statement is protocol-dependent, it is simple enough be adapted to any e-voting protocol — precisely as the current literature shows with the confidentiality goal. No honesty assumptions are made about the election officials Administrator and Collector.

The proof involves case-splits about possible associations arising from each protocol step, and a number of lemmas specifying possible associations, such as the following one:

lemma *aanalz_PR*:
$[\![a \in$ *aanalz Spy evs*; *Crypt P R* $\in a$; *evs* $\in$ *foo*$]\!] \Longrightarrow$
(*Agent Col* $\notin a$ $\vee$
(*Agent V* $\in a \longrightarrow V \in$ *bad* $\vee$ $V =$ *Col*) $\vee$
(*Nonce Nv* $\notin$ *parts* $\{R\}$)) $\wedge$
((*Nonce Nv* $\notin a$) $\vee$
(*Key* (*invKey P*) $\in$ *analz* (*spies evs*) $\wedge$
Agent V $\notin$ *parts* $\{R\}$))

This lemma establishes properties of association sets from the protocol model that contain at least one ciphertext. It states that if a nonce appears as an atomic component of the body of the ciphertext, then either the name of the election official Collector does not appear in the association, or all agent names appearing in it are either dishonest agents or the Collector.

Specifying and verifying the unlinkability model involved substantial effort. About 20 subsidiary lemmas had to be proved beforehand. Many of them, like the one just mentioned, *analz_PR*, establish properties of association sets. Remarkably, the generic nature of the proof has great potential for reuse over other protocols. Also, its redundant parts may be programmed as ML tactics for greater automation. In cases where unlinkability does not hold, the subgoal that cannot be proven indicates which protocol step can lead to an attack.

5 CONCLUSION

We shed a different light on voter privacy by seeing it as unlinkability property between two pieces of information. The feasibility of this model is shown by a specification in the Inductive Method and a successful privacy proof for a classic e-voting protocol. While the process equivalence model is showing steady progress through new trace equivalence approximations, our machine-assisted analysis is precise for an unbounded number of sessions, and supports proof reuse. Its interactive nature also provides broader protocol understanding to the analyst. Since process equivalence-based methods are not currently able to deal with all protocols, it seems worthwhile to investigate alternative approaches like this one.

A submission of our Isabelle theories to the online Archive of Formal Proofs (Klein et al., 2012) is being prepared.

Future Work Demonstrating the flexibility and reusability of our specification through additional examples is our main next task. The proofs must be adapted to a generalised association synthesiser. We then intend to analyse protocols that cannot be handled in current implementations of the indistinguishability model, such as e-passport protocols, and investigate the specification of associative/commutative operators. Finally, we plan to model other privacy-type properties, such as coercion resistance.

REFERENCES

Bella, G. (2007). *Formal Correctness of Security Protocols*. Information Security and Cryptography. Springer.

Blanchet, B. (1998). An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. of the 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Press.

Blanchet, B., Abadi, M., and Fournet, C. (2008). Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51.

Blanchet, B. and Podelski, A. (2005). Verification of cryptographic protocols: tagging enforces termination. *Theoretical Computer Science*, 333(1-2):67–90. Special issue FoSSaCS'03.

Chadha, R., Ciobâcă, Ş., and Kremer, S. (2012). Automated verification of equivalence properties of cryptographic protocols. In *Programming Languages and Systems —Proceedings of the 21th European Symposium on Programming (ESOP'12)*, Lecture Notes in Computer Science. Springer. To appear.

Delaune, S., Kremer, S., and Ryan, M. (2009). Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487.

Delaune, S., Kremer, S., and Ryan, M. D. (2010). Verifying privacy-type properties of electronic voting protocols: A taster. In *Towards Trustworthy Elections – New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 289–309. Springer.

Delaune, S., Ryan, M., and Smyth, B. (2008). Automatic verification of privacy properties in the applied pi calculus. *Syntax*, 263/2008:263278.

Fujioka, A., Okamoto, T., and Ohta, K. (1993). A practical secret voting scheme for large scale elections. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, ASIACRYPT 'a92, pages 244–251. Springer-Verlag.

Klein, G., Nipkow, T., and Paulson, L. (2012). In *The Archive of Formal Proofs*. `http://afp.sf.net`.

Kremer, S. and Ryan, M. (2005). Analysis of an electronic voting protocol in the applied pi calculus. In *In Proc. 14th European Symposium On Programming (ESOP05), volume 3444 of LNCS*, pages 186–200. Springer.

Martina, J. E. and Paulson, L. C. (2011). Verifying multicast-based security protocols using the inductive method. In *Workshop on Formal Methods and Cryptography (CryptoForma 2011)*.

Paulson, L. C. (1998). The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128.

Schneider, S. and Sidiropoulos, A. (1996). CSP and anonymity. In *In European Symposium on Research in Computer Security*, pages 198–218. Springer-Verlag.