Attacking the Asokan-Ginzboorg Protocol for Key Distribution in an Ad-Hoc Bluetooth Network Using CORAL

Attacking the Asokan–Ginzboorg Protocol for Key Distribution in an Ad-Hoc Bluetooth Network Using CORAL

Graham Steel, Alan Bundy, and Monika Maidl

Division of Informatics,
University of Edinburgh,
Edinburgh, EH8 9LE, Scotland,
e-mail: g.j.steel@ed.ac.uk,bundy,monika@inf.ed.ac.uk
http://dream.dai.ed.ac.uk/graham

Abstract. We describe CORAL, a counterexample finder for incorrect inductive conjectures. By devising a first-order version of Paulson's formalism for cryptographic protocol analysis, [10], we are able to use CORAL to attack protocols which may have an unbounded number of principals involved in a single run. We show two new attacks CORAL has found on the Asokan–Ginzboorg protocol for establishing a group key in an ad-hoc network of Bluetooth devices, [1].
Keywords: Wireless and mobile communication protocols, cryptographic protocols, refutation of inductive conjectures.

1 Introduction

An abundance of tools and techniques have appeared in the last few years to check security properties of two and three party cryptographic protocols. However, very few of these are able to analyse protocols for group key agreement, where an unbounded number of parties may be involved in a single round, [8, 10]. The only attacks on such protocols to have appeared in the literature have been discovered by hand, [11]. It seems flexible sized group protocols are not easy to attack using model checking type approaches. In this paper, we show how our counterexample finder for inductive conjectures, CORAL, has been used to automatically discover two new attacks on such a protocol: the Asokan–Ginzboorg protocol for key establishment in an ad-hoc Bluetooth network, [1].

In the rest of the paper, we first describe the CORAL system and our first-order model for the protocol problem. We describe the Asokan–Ginzboorg protocol, and then outline how CORAL models its operation directly, without having to restrict to a small concrete instance involving a fixed number of participants. We give the specification for the intruder's capabilities that we used, and explain how we found the new attacks. Finally, we discuss related work and further work, and give some conclusions.

2 The CORAL System

The refutation of incorrect inductive conjectures is a problem of general interest in formal methods and verification. Inductive theorem provers are frequently employed in

the verification of programs, algorithms and protocols. Programs and algorithms often contain bugs, and protocols may be flawed, causing the proof attempt to fail. However, it can be hard to interpret a failed proof attempt; it may be that some additional lemmas need to be proved or a generalisation made. In this situation, a tool which can not only detect an incorrect conjecture, but also supply a counterexample in order to allow the user to identify the bug or flaw, is potentially very valuable.

CORAL is such a tool, and is based on a previously under-exploited feature of the *proof by consistency* technique. Proof by consistency is a technique for automating inductive proofs in first-order logic. Originally developed to prove correct theorems, this technique has the property of being refutation complete, i.e. it is able to refute in finite time conjectures which are inconsistent with the set of hypotheses. Recently, Comon and Nieuwenhuis have drawn together and extended previous research to show how it may be more generally applied, [3]. CORAL is the first full implementation of this technique, built on the theorem prover SPASS, [14].

The Comon–Nieuwenhuis method relies on a number of theoretical results, but informally, a proof attempt involves two parts. In one, we pursue a *fair induction derivation*. This is a restricted kind of saturation, [2], where we need only consider overlaps between axioms and conjectures, and produce inferences from an adapted superposition rule. In the second part, every clause in the induction derivation is checked for consistency against an *I-Axiomatisation*. This is usually a set of clauses that are sufficient for deciding inequality of ground terms. If any consistency check fails, then the conjecture is incorrect. If they all succeed, and the induction derivation procedure terminates, the theorem is proved. Comon and Nieuwenhuis have shown refutation completeness for this system, i.e. any incorrect conjecture will be refuted in finite time, even if the search for an induction derivation is non-terminating. More details can be found in [3].

In the case where a refutation is found, we are able to extract a counterexample by means of a well-known method first proposed by Green, [6]. When CORAL refutes a security conjecture of the form $\forall \mathrm{trace}.P(\mathrm{trace})$, it has proved in its superposition calculus that $\exists \mathrm{trace}.\neg P(\mathrm{trace})$. We track the instantiations made to the trace variable using an *answer literal*, following Green's method. Green has shown that this will always yield a correct constructive answer for these types of proofs. We show how new attacks are discovered as counterexamples in section 5.

2.1 Modelling Protocols

Paulson's inductive approach has been used to verify properties of several protocols, including group key protocols, [10]. Note however that Paulson's approach has no automated support for finding attacks on faulty protocols. Protocols are formalised in typed higher-order logic. However, no fundamentally higher-order concepts are used – in particular there is no unification of functional objects. Objects have types, and sets and lists are used. All this we model in first-order logic. Like Paulson's, our model allows an indeterminate and unbounded number of agents to participate, playing any role, and using an arbitrary number of fresh nonces and keys.

A protocol is modelled as the set of all possible traces, i.e. all possible sequences of messages sent by any number of honest users under the specification of the protocol and, additionally, faked messages sent by the intruder. A trace of messages is modelled

as a list. For a specific protocol, we generally require one rule per protocol message, each one having the informal interpretation, 'if xt is a trace containing message n addressed to agent xa, then the trace may be extended by xa responding with message $n + 1$'.

The intruder has the usual capabilities specified by the Dolev-Yao model, [4]. We specify intruder knowledge in terms of sets. Given a trace of messages exchanged, xt, we define $analz(xt)$ to be the least set including xt closed under projection and decryption by known keys. This is accomplished by using exactly the same rules as the Paulson model, [10, p. 12]. Then we can define the messages the intruder may send, given a trace xt, as being members of the set $synth(analz(xt))$, where $synth(x)$ is the least set containing x, including agent names and closed under pairing and encryption by known keys. The agent may send anything he can provided it matches the template of the messages in the protocol. This last feature is an optimisation to make searching for attacks more efficient, since the spy gains nothing from sending a message that no honest agent will respond to.

More details of our model can be found in our earlier paper, [12]. Using this model, CORAL has rediscovered several known attacks on two and three party security protocols, including Needham-Schroeder, Neumann-Stubblebine and Otway-Rees. This latter is significant since it requires an honest agent to generate two fresh nonces and to play the role of both the initiator and the responder, things which previous first-order models have not allowed, [13]. CORAL has also discovered the new attacks described below.

3 The Asokan–Ginzboorg Protocol

The Asokan–Ginzboorg protocol is an application level protocol for use with Bluetooth devices. The scenario under consideration is this: a group of people are in a meeting room and want to set up a secure session amongst their laptops. They know and trust each other, but their computers have no shared prior knowledge and there is no trusted third party or public key infrastructure available. The protocol proceeds by assuming a short group password is chosen and displayed, e.g. on a whiteboard. The password is assumed to be susceptible to a *dictionary attack*, but the participants in the meeting then use the password to establish a secure secret key.

Asokan and Ginzboorg describe two protocols for establishing such a key in their paper, [1]. We have analysed the first of these. Completing analysis of the second protocol using CORAL would require some further work (see section 8). Here is a description of the first Asokan–Ginzboorg protocol (which we will hereafter refer to as simply 'the Asokan–Ginzboorg protocol'). The members of the group are written as M_i, with M_n acting as group leader.

$$
\begin{array}{lll}
1.\ M_n \rightarrow \mathrm{ALL} : M_n, \{\!|\ E |\!\}_P & & \\
2.\ M_i \rightarrow M_n \ : M_i, \{\!|\ R_i, S_i |\!\}_E & & i = 1, \ldots, n-1 \\
3.\ M_n \rightarrow M_i \ : \{\!|\ \{Sj, j = 1, \ldots, n\} |\!\}_{R_i} & & i = 1, \ldots, n-1 \\
4.\ M_i \rightarrow M_n \ : M_i, \{\!|\ S_i, h(S_1, \ldots, S_n) |\!\}_K & & \text{some } i
\end{array}
$$

Informally, what is happening is this:

1. M_n broadcasts a message containing a fresh public key, E, encrypted under the password, P, written on the whiteboard.
2. Each other participant M_i, for $i = 1, \ldots, n-1$, sends M_n a contribution to the final key, S_i, and a fresh symmetric key, R_i.
3. Once M_n has a response from everyone, she collects together the S_i in a package along with a contribution of her own (S_n) and sends out one message to each participant, containing this package and encrypted under the respective symmetric key R_i.
4. One participant responds to M_n with the package she just sent passed through a one way hash function and encrypted under the new group key $K = f(S_1, \ldots, S_n)$, with f a commonly known function.

Asokan and Ginzboorg argue that it is sufficient for each group member to receive confirmation that one other member knows the key: everyone except M_n receives this confirmation in step 3. M_n gets confirmation from a random member of the group in step 4.

3.1 Modelling the Protocol

The advantage of our approach is that we can model flexible group protocols without having to restrict ourselves to a small concrete instance with a fixed number of participants. This means that we do not have to guess how many players are needed to achieve an attack, CORAL will search for attacks involving any number of participants. Of course, it is more likely to find attacks with small number of participants first, since this will involve reasoning with smaller clauses.

For messages 1, 2 and 4, our model is similar to that for a two party protocol, as described briefly above (section 2.1). The main challenge was to cope with message 3. For a group of n participants, $n-1$ message 3s will be sent out, each encrypted under a different key. Moreover, the group leader for the run must check that she has received $n-1$ message 2s. In order to model this, we need to accommodate the possibility of different numbers of messages being added to the trace in one instant. This is solved by the use of a logic programming style approach. We define a new predicate which checks a trace to see if all message 2s have been sent for a particular group leader and number of participants. It returns a new trace containing the response prescribed by the protocol. It works in all modes of instantiation, allowing us to use it to construct protocol runs for various sized groups while the refutation search is going on. For more details, see `http://homepages.inf.ed.ac.uk/s9808756/asokan-ginzboorg-model/`.

4 Modelling Spies in a Wireless Network

The Dolev-Yao model is widely agreed upon as a realistic model of a spy in a fixed-network. However, some of their assumptions seem inappropriate in a single-hop wireless situation. Although an attacker may be able to totally disrupt communications by jamming the radio signal, it seems unlikely that he would be able to selectively intercept some messages and remove them from the airwaves, while allowing other messages to

pass. Asokan and Ginzboorg mention that they intend the protocol to be tolerant to disruption attacks by an attacker who can add fake messages, but not block or delay messages, so we assume these abilities for our spy.

There is also the question of whether the spy should be accepted as an honest participant by the other players, as he generally is in fixed-network protocol models[1]. An honest participant will be present in the room and able to see the password on the whiteboard, whereas the protocol seems mostly intended to protect against attacks by a spy in an adjacent room, who can interfere with communications but who does not know the password. However, the protocol is explicitly designed to protect a participant against conspiracy by a group of other participants trying to restrict the agreed key to a prechosen range. Some possible dishonesty from participants in the room must therefore be considered.

We decided to test the protocol against two attackers: one inside the room, and one outside. Different outcomes are considered successful for the different spies.

Spy 1 - Outside the Room

This spy cannot see the whiteboard, so does not know the password. His objective is to effect a *disruption attack*, i.e. to make honest participants accept keys which are not mutually shared.

Spy 2 - In the Room

This spy's capabilities differ from the first in that he knows the passwords written on the whiteboard, and is accepted as an honest agent. His objective is to gain control of communication in the room, i.e. by making all participants agree on different keys that only he knows.

5 Discovering Attacks

To check for disruption attacks, we gave CORAL a conjecture stating that any complete run of the protocol (i.e. one finished by a valid message 4) must contain a correct message 2 for every message 3. This is because principals make their contribution to the key in message 2, then receive the whole key under their symmetric key R_i in message 3. So, if every message 3 corresponds to a message 2 then everyone can read the key. If this conjecture is incorrect, and there is a trace violating this property, this will constitute an attack. In clausal normal form, the conjecture says, 'there is no valid trace containing a message 3 with an unmatched message 2', and looks like this:

% some honest xi has replied with a message 4
member(sent(xi,mn,pair(principal(xi),encr(pair(nonce(si),h(package)),f(package))))
,xtrace)=true ∧

[1] In some models he is assumed not to be an honest participant, but instead to control the long term keys of an honest participant, though this amounts to the same thing.

% a message 3 (to someone) is in the trace
member(sent(x,xj,encr(package,nonce(rj))),xtrace)=true ∧
% but there's no corresponding message 2, so the message 3 is wrong or faked
member(sent(xj,mn,pair(principal(xj),encr(pair(nonce(rj),nonce(sj)),key(e))))
,xtrace)=false →

CORAL found the following counterexample:[2]

```
cons(sent(s(a),a,pair(principal(s(a)),encr(pair(nonce(U),
  h(cons(nonce(U),cons(nonce(Y),nil)))),
  f(cons(nonce(U),cons(nonce(Y)),nil))))),
cons(sent(spy,s(a),
  encr(cons(nonce(U),cons(nonce(Y),nil)),nonce(V))),
cons(sent(s(a),a,
  encr(cons(nonce(U),cons(nonce(Y),nil)),nonce(V))),
cons(sent(spy,s(a)),pair(principal(a),
  encr(pair(nonce(V),nonce(U)),key(W)))),
cons(sent(s(a),a,pair(principal(s(a)),
  encr(pair(nonce(V),nonce(U)),key(W)))),
cons(sent(spy,all,pair(principal(a),
  encr(key(W),key(X)))),
cons(sent(s(a),all,pair(principal(s(a)),
  encr(key(W),key(X)))),nil))))))
```

this corresponds to the following disruption attack on a group of size 2:[3]

$$
\begin{array}{lll}
1. & B \rightarrow \mathrm{ALL} & : B, \{\!|\, E_1 \,|\!\}_P \\
1'. & spy_A \rightarrow \mathrm{ALL} & : A, \{\!|\, E_1 \,|\!\}_P \\
2'. & B \rightarrow A & : B, \{\!|\, R_B, S_B \,|\!\}_{E_1} \\
2. & spy_A \rightarrow B & : A, \{\!|\, R_B, S_B \,|\!\}_{E_1} \\
3. & B \rightarrow A & : \{\!|\, S'_B, S_B \,|\!\}_{R_B} \\
3'. & spy_A \rightarrow B & : \{\!|\, S'_B, S_B \,|\!\}_{R_B} \\
4.' & B \rightarrow A & : B, \{\!|\, S_B, h(S'_B, S_B) \,|\!\}_{f(S'_B, S_B)}
\end{array}
$$

At the end of the run, B now accepts the key $f(S'_B, S_B)$ as a valid group key, but it contains numbers known only to B. The spy could perform a mirror image of this attack on A to make her accept another key unknown to B – provided A tries to initiate a run herself as group leader when B's attempt fails. Similar replay attacks could be made on groups of a larger size – a spy can just repeat the first message 2 he sees, with the agent identifier at the front swapped for that of one of the other agents in the group. He would not even have to start an overlapping session, as in the attack above. To change the protocol to protect against these attacks is quite straightforward (see section 6).

[2] Note that CORAL presents the trace in reverse.

[3] We have renamed the variables and the names of the participants to make the attack easier to follow.

The attack above requires that the spy manages to send message 2 before the honest agent A can send her reply. In a single hop network, there is a certain amount of luck involved. It might require A to invoke the command to establish a key just after B does. In a multi hop network, the intruder might be able to effect the attack quite easily.

5.1 Attack by a Spy Inside the Room

To model the spy being in the room, we made him an accepted agent and added all whiteboard passwords to his knowledge set. We gave CORAL a conjecture stating that at the end of a run, principals have agreed on the same key (note this is different to the conjecture above, where we just required that principals had received a key they could read). The conjecture looks like this:

% if trace contains two message 2s in response to the same message 1
member(sent(xi,mn,pair(principal(xi),encr(pair(nonce(ri),nonce(si)),key(e))))
,xtrace)=true $\wedge$
member(sent(xj,mn,pair(principal(xj),encr(pair(nonce(rj),nonce(sj)),key(e))))
,xtrace)=true $\wedge$
% and two message 3s (from someone) responding to those
member(sent(x,xi,encr(package1,nonce(ri))),xtrace)=true$\wedge$
member(sent(y,xj,encr(package2,nonce(rj))),xtrace)=true$\wedge$
% then the keys received must be the same
$\rightarrow$ *eq(package1,package2)=true*

CORAL found the counterexample shown below. The result of this attack is that A has a key she believes is the group key but is in fact only shared by her and the spy, and B has a key he thinks is the group key but is only known to him and the spy. This means that the spy can see everything sent, decide what to pass on and possibly doctor material as it is passed on:

1. $spy \rightarrow \mathrm{ALL} : spy, \{\!|\ E_1 |\!\}_P$
2. $A \rightarrow spy : A, \{\!|\ R_A, S_A |\!\}_{E_1}$
2. $B \rightarrow spy : B, \{\!|\ R_B, S_B |\!\}_{E_1}$
3. $spy \rightarrow A : \{\!|\ S_{spy}, S_A, S'_{spy} |\!\}_{R_A}$
3. $spy \rightarrow B : \{\!|\ S_{spy}, S_A, S_B |\!\}_{R_B}$
4. $A \rightarrow spy : A, \{\!|\ S_A, h(S_{spy}, S_A, S'_{spy}) |\!\}_{f(S_{spy}, S_A, S'_{spy})}$

This is just a standard protocol run for three participants, except that in the first message 3, the spy switches in a number of his own (S'_{spy} in the place of S_B). This means that A accepts the key as $f(S_{spy}, S_A, S'_{spy})$, whereas B accepts $f(S_{spy}, S_A, S_B)$, and both of the keys are known to the spy.

Note that only the initiator of the run can effect this attack - the use of public key cryptography in message 2 prevents another participant from being able to carry out a similar deception. The attack can be easily adapted to a group of any size.

6 Protecting Against the Attacks

To protect against the disruption attack, we can include the agent identifier inside the encrypted package in message 1. This will prevent the faked message 1'. We could also demand an implementation that protects against replays, but this is generally frowned upon in protocol analysis circles. The agent identifier in message 2 should also be encrypted, to prevent replays against groups of size 3 or more.

Protecting against the second attack is a more subtle problem: one way to do it would be to have message 4 broadcast to all participants. This would make sense in a normal run as it would allow other participants to see that someone had sent message 4, and that the run was finished. Other participants could open the encrypted package and check that the hashed number matches the hash of the key they received in message 3. If anyone is unhappy, they could cry foul and get everyone to do another run. Two conspiring agents could subvert this defence by sending a suitably doctored message 4, but there is no advantage for them to gain by having a separate key for themselves. The spy could fake a message 4 from one of the agents, but since the message is broadcast to everyone, the agent concerned could recognise that he is being impersonated and call a halt to proceedings.
Here is the revised protocol:

1. $M_n \rightarrow \mathrm{ALL} : \{\!|\ M_n, E|\!\}_P$
2. $M_i \rightarrow M_n \ : \{\!|\ M_i, R_i, S_i|\!\}_E, i = 1, \ldots, n-1$
3. $M_n \rightarrow M_i \ : \{\!|\ \{Sj, j = 1, \ldots, n\}|\!\}_{R_i}, i = 1, \ldots, n-1$
4. $M_i \rightarrow ALL : M_i, \{\!|\ S_i, h(S_1, \ldots, S_n)|\!\}_K$, some i.

7 Related Work

The closest related work is probably that of Meadows, [8]. Here, NPA, a state exploration based tool for analysing cryptographic protocols, was extended to deal with the CLIQUE group protocols. However, NPA could not rediscover the attacks found by Pereira and Quisquater, [11], whose analysis was carried out by hand. The attacks they found were quite subtle, involving properties of the Diffie-Hellman exponentiation operation widely used in the CLIQUE suite. They also involved the spy doing some quite imaginitive things, like joining the group, leaving, and then forcing the remaining members to accept a compromised key, which NPA could not account for. It would be interesting to see whether these kinds of attacks could be found automatically by CORAL. We hope that the very flexible inductive model used might mean that these attacks are within CORAL's scope.

Recently, Millen and Shmatikov looked at Pereira and Quisquater's attacks as part of their work on integrating products and Diffie-Hellman exponentiation into symbolic protocol analysis, [9]. They sketched how one of their attacks could be rediscovered in their setting, though this has not been automated. Finding the attacks could potentially be very efficient. However, their approach requires the user to decide how many participants playing what role should be considered in the search for an attack, which our approach does not.

8 Further Work

Our work with CORAL is ongoing. We intend to analyse further protocols in the future, focusing on the kind of flexible or group protocols that are not easy to attack using a model checking approach. We do not intend to compete for speed of attack finding on a large corpus of standard protocols, since other approaches are better suited to this.

As CORAL is built on SPASS, a theorem prover capable of equational reasoning, we should be able to reason about some simple algebraic properties of the cryptosystems underlying protocols, such as Diffie-Hellman type operations. This would allow us to analyse the second Asokan–Ginzboorg protocol, which is quite different to the first and seems not to be susceptible to the same attacks, and also the CLIQUES protocols mentioned above. The main task here would be to devise a way of modelling the low-level cryptographic operations in such a way that the essential properties are captured, but without going into too fine a degree of detail which would make automated analysis unfeasible. Another option might be to farm out algebraic work to a connected computer algebra package.

In the longer term we intend to try to exploit the flexibility of our system as a general tool for inductive counterexample finding, and apply it to some other security problems. One idea is to use the system to model security problems at a higher level. We could model a company's computer network as a system of local networks and servers, firewalls etc. all with formally defined behaviour, and examine how interactions in the presence of intruders might lead to exploitable vulnerabilities. To deal with larger problems like this, we might need to enhance SPASS to exploit domain knowledge a little more. A user defined strategy that can vary as the proof proceeds, and a critics mechanism, [7], to suggest pruning lemmas are two possible ideas we intend to explore.

In theory, CORAL can also show security properties of protocols to be correct when there are no attacks to be found. However, to make this work in practice would require some considerable work. The formulae to be proved are significantly larger than the kinds of examples that have been proved by proof by consistency in the past. The key factor would be the generation of relevant lemmas, which could be done in similar way to the NPA tool.

9 Conclusions

We have presented CORAL, our system for refuting incorrect inductive conjectures, and shown two new attacks on the Asokan–Ginzboorg protocol that it found. An advantage of looking for attacks on group protocols with CORAL is that we can model the protocol generally and so look for attacks on any size of group. A model checking type approach would have required us to fix the size of the group in advance. Additionally, the flexibility of a theorem proving approach allows us to look for unconventional attacks, e.g. where two members of the group share different keys with the intruder which they both believe to be the group key.

In future we intend to attack more group protocols, particularly wireless group protocols, and then try to model other security problems inductively and look for more vulnerabilities.

References

1. N. Asokan and P. Ginzboorg. Key-agreement in ad-hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.
2. L. Bachmair and H. Ganzinger. Completion of First-order clauses with equality by strict superposition (extended abstract). In *Proceedings 2nd International CTRS Workshop*, pages 162–180, Montreal, Canada, 1990.
3. H. Comon and R. Nieuwenhuis. Induction = I-Axiomatization + First-Order Consistency. *Information and Computation*, 159(1-2):151–186, May/June 2000.
4. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions in Information Theory*, 2(29):198–208, March 1983.
5. H. Ganzinger, editor. *Automated Deduction – CADE-16, 16th International Conference on Automated Deduction*, LNAI 1632, Trento, Italy, July 1999. Springer-Verlag.
6. C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4. Edinburgh University Press, 1969.
7. A. Ireland. Productive use of failure in inductive proof. *Journal of Automated Reasoning*, 16(1-2):79–111, 1996.
8. C. Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In P. Degano, editor, *Proceedings of the First Workshop on Issues in the Theory of Security*, pages 87–92, Geneva, Switzerland, July 2000.
9. J. Millen and V. Shmatikov. Symbolic protocol analysis with products and diffie-hellman exponentiation. In *IEEE Computer Security Foundations Workshop*, 2003.
10. L.C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85–128, 1998.
11. O. Pereira and J.-J. Quisquater. Security analysis of the cliques protocols suites: 1st results. In *In Proceedings of IFIP Sec'01*, pages 151–166, June 2001.
12. G. Steel, A. Bundy, and E. Denney. Finding counterexamples to inductive conjectures and discovering security protocol attacks. In *Proceedings of the Foundations of Computer Security Workshop*, 2002. Appeared in Proceedings of The Verify'02 Workshop as well. Also available as Informatics Research Report EDI-INF-RR-0141.
13. C. Weidenbach. Towards an automatic analysis of security protocols in First-order logic. In Ganzinger [5], pages 314–328.
14. C. Weidenbach et al. System description: SPASS version 1.0.0. In Ganzinger [5], pages 378–382.