

# SATMC: a SAT-based Model Checker for Security Protocols<sup>\*</sup>

Alessandro Armando and Luca Compagna

AI-Lab, DIST – Università di Genova  
Viale Causa 13, 16145 Genova, Italy.  
{armando,compa}@dist.unige.it

## 1 Introduction

We present SATMC (SAT-based Model Checker), an open and flexible platform for SAT-based bounded model checking [8] of security protocols. Under the standard assumptions of *perfect cryptography* and of *strong typing*, SATMC performs a bounded analysis of the problem by considering scenarios with a finite number of sessions whereby messages are exchanged on a channel controlled by the most general intruder based on the Dolev-Yao model [12].

Given a positive integer  $k$  and a protocol description written in a rewrite-based formalism, SATMC automatically generates a propositional formula by using sophisticated encoding techniques developed for planning (see [13] for a survey); state-of-the-art SAT-solvers taken off-the-shelf are then used to check the propositional formula for satisfiability and any model found by the solver is turned into a partially ordered set of transitions of depth  $k$  whose linearizations correspond to attacks on the protocol. If the formula is found to be unsatisfiable, then  $k$  is incremented and the whole procedure is iterated until either a termination condition is met or a given upper bound for  $k$  is reached.

Experimental results indicate that the approach is very effective: SATMC takes a few seconds to analyze and detect flaws on 22 protocols of the Clark&Jacob library [10].

## 2 Bounded Model Checking of Security Protocols

A security problem associated to a given security protocol consists of determining whether some undesirable states can be reached starting from some initial states by using the legal protocol steps and the admissible actions of the intruder.

Like other SAT-based model checkers, SATMC tackles the (bounded) reachability problem by generating SAT formulae of the form:  $\Phi_H^k = I_0 \wedge \bigwedge_{i=0}^{k-1} T_i^{i+1} \wedge G_k$  where  $I_0$ ,  $T_i^{i+1}$ , and  $G_k$  are the boolean formulae representing the initial state, the transition relation, and the goal states, respectively. The main difference

---

<sup>\*</sup> This work was partially funded by the FET Open EC Project “AVISPA: Automated Validation of Internet Security Protocols and Applications” (IST-2001-39252) and by the FIRB Project no. RBAU01P5SS.

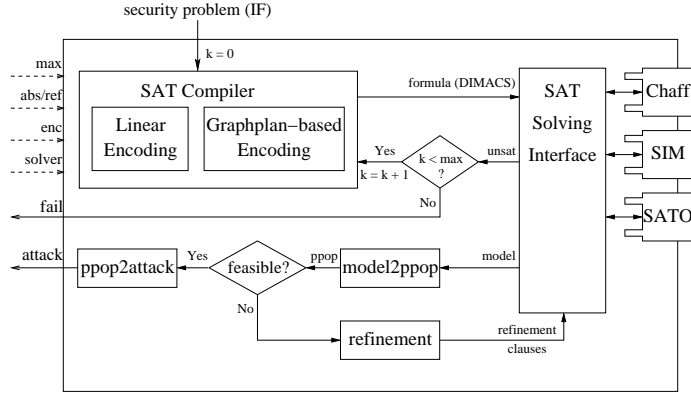


Fig. 1. Architecture of SATMC

between the encoding techniques used by SATMC and those employed in other bounded model checkers (e.g. NuSMV [9]) is in the way the formula that encodes the transition relation, i.e.  $T_i^{i+1}$ , is generated. Two encoding techniques are currently implemented in SATMC: the first belongs to the family of so-called *linear encodings* [3], the second is the more sophisticated *graphplan-based encoding* [6].

### 3 SATMC

The architecture of SATMC is shown in Figure 1. SATMC takes as input a security problem expressed in a rewrite-based formalism called IF [2] and the following parameters: `max`, an integer to be used as upper-bound for iterative deepening; `abs/ref`, a boolean parameter used to enable the abstraction/refinement strategy introduced in [4]; `enc`, the selected SAT reduction encoding technique (*linear* and *graphplan* are currently supported); `solver` (Chaff, SIM, and SATO are currently supported). SATMC returns either an attack or reports that no attack has been found on the input security problem by using up to `max` unfoldings of the transition relation.

The SAT Compiler first translates the IF specification into a SATE specification<sup>1</sup> and then applies the selected encoding technique to the SATE specification for increasing values of  $k$ . The formula generated at each step is fed to the selected state-of-the-art SAT solver through the SAT Solving Interface. As soon as a satisfiable formula is found, the corresponding model is translated back (by the `model2ppop` module) into a partially ordered set of transitions of depth  $k$  (also called pseudo partial-order plans) potentially representing a set of attacks on the protocol. The tool then checks whether there exists a linearization of the

<sup>1</sup> SATE is the native STRIPS-like language used in SATMC.

**Table 1.** Experimental data on the Clark-Jacob's library [10]

Protocol	K	A	CL	ET	ST	Protocol	K	A	CL	ET	ST
<i>Andrew</i>	9	442	1,365	0.14	0.01	<i>KLS rep.</i>	7	1,634	23,190	3.89	0.03
<i>EKE</i>	5	394	1,337	0.12	0.00	<i>NSCK</i>	9	435	1,406	0.12	0.00
<i>ISO-CCF-1 U</i>	4	102	295	0.00	0.00	<i>NSPK</i>	7	411	1,279	0.09	0.00
<i>ISO-CCF-2 M</i>	4	115	311	0.02	0.00	<i>NSPK-server</i>	8	847	2,702	0.23	0.00
<i>ISO-PK-1 U</i>	4	149	418	0.03	0.00	<i>SPLICE</i>	9	951	3,168	0.32	0.00
<i>ISO-PK-2 M</i>	4	129	363	0.02	0.00	<i>Swick 1</i>	5	192	554	0.06	0.00
<i>ISO-SK-1 U</i>	4	93	265	0.01	0.00	<i>Swick 2</i>	6	257	838	0.08	0.00
<i>ISO-SK-2 M</i>	4	117	314	0.02	0.00	<i>Swick 3</i>	4	171	498	0.05	0.01
<i>KaoChow 1</i>	7	426	1,781	0.18	0.01	<i>Swick 4</i>	5	215	634	0.04	0.00
<i>KaoChow 2</i>	9	726	3,393	0.32	0.00	<i>Stubblebine rep</i>	3	146	478	0.04	0.00
<i>KaoChow 3</i>	9	990	6,118	0.66	0.03	<i>Woo-Lam M</i>	6	481	1,539	0.19	0.00

partially ordered set of transitions.<sup>2</sup> If no linearization exists, then the associated attack is computed and returned, otherwise the refinement module refines the encoding by adding new clauses and the whole procedure is iterated. As soon as a linearizable partial-order sequence of transitions is found, the `ppop2attack` module translates it into an `attack` which is reported to the user.

Thanks to a recent extension [7], SATMC now also compiles protocol security problems into logic programs which are in turn fed into a state-of-the-art solver for logic programs.

## 4 Experiments

We have thoroughly tested SATMC against a large number of security protocols of the Clark&Jacob library [10]. Table 1 reports the results of our experiments.<sup>3</sup> For each protocol we give the smallest value of  $k$  at which the attack is found (**K**), the number of propositional variables (**A**) and clauses (**CL**) in the SAT formula, the time spent to generate the SAT formulae (**ET**), and the total time spent by the SAT solver (**ST**). Notice that the solving times are negligible for all problems and that the encoding time is less than 1 second for all problems but one. These results confirm the effectiveness of approach and pave the way to the application of SATMC to protocols of industrial complexity.<sup>4</sup>

The SATMC tool and all the IF specifications of the security problems used in our experiments can be found at <http://www.ai.dist.unige.it/satmc>.

<sup>2</sup> If the abstraction/refinement strategy is disabled (i.e. `abs/ref=false`), the linearization is guaranteed to exist and the check is avoided. See [4] for the details.

<sup>3</sup> Experiments have been carried out on a PC with a 1.4 GHz CPU, 1 GB of RAM, and with `max=10`, `abs/ref=false`, `enc=graphplan`, and `solver=chaff`.

<sup>4</sup> SATMC is one of the back-ends of the AVISPA Tool (<http://www.avispa-project.org>), a platform for the development of large-scale Internet security protocols.

## 5 Related Work

A large number of techniques for the analysis security protocols are available. While some techniques (e.g., [16, 15]) are tailored to the analysis of security protocols, others reduce the analysis of security protocols to some general purpose formalism such as CSP [14], rewriting logic [11], or logic programming [1]. SATMC, by performing a reduction to propositional logic, clearly belongs to the second camp and similarly to all techniques in the this camp, it can be easily adapted in response to changes or extensions to the underlying model (e.g. as when properties of cryptographic operators or different models of the intruder are to be taken into account—see, e.g., [5]). Additionally—thanks to the efficiency of state-of-the-art SAT solvers—SATMC performance comparable to that of techniques in the first camp [2].

## References

1. L. C. Aiello and F. Massacci. Verifying security protocols as planning in logic programming. *ACM Trans. on Computational Logic*, 2(4):542–580, Oct. 2001.
2. A. Armando, D. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The AVISS Security Protocol Analysis Tool. In *Proc. of CAV'02*, LNCS 2404. Springer-Verlag, 2002.
3. A. Armando and L. Compagna. Automatic SAT-Compilation of Protocol Insecurity Problems via Reduction to Planning. In *Proc. of FORTE 2002*. 2002.
4. A. Armando and L. Compagna. Abstraction-driven SAT-based Analysis of Security Protocols. In *Proc. of SAT 2003*, LNCS 2919. Springer-Verlag, 2003.
5. A. Armando and L. Compagna. An Optimized Intruder Model for SAT-based Model-Checking of Security Protocols. In *Proc. of ARSPA Workshop*. 2004.
6. A. Armando, L. Compagna, and P. Ganty. SAT-based Model-Checking of Security Protocols using Planning Graph Analysis. In *Proc. of FME'2003*. 2003.
7. A. Armando, L. Compagna, and Y. Lierler. Automatic Compilation of Protocol Insecurity Problems into Logic Programming. In this volume, 2004.
8. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *Proc. of TACAS'99*, LNCS 1579. Springer-Verlag, 1999.
9. A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A New Symbolic Model Verifier. *LNCS 1633*, 1999.
10. J. Clark and J. Jacob. A Survey of Authentication Protocol Literature: Version 1.0, 17.Nov.1997. URL: [www.cs.york.ac.uk/~jac/papers/drareview.ps.gz](http://www.cs.york.ac.uk/~jac/papers/drareview.ps.gz).
11. G. Denker, J. Meseguer, and C. Talcott. Protocol specification and analysis in Maude. In *Proc. of the Workshop on Formal Methods and Security Protocols*. 1998.
12. D. Dolev and A. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.
13. H. Kautz, H. McAllester, and B. Selman. Encoding Plans in Propositional Logic. In *Proc. of KR'96*, 1996.
14. G. Lowe. Casper: a Compiler for the Analysis of Security Protocols. *Journal of Computer Security*, 6(1):53–84, 1998.
15. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. of the CCS'01*, pages 166–175, 2001.
16. D. Song. Athena: A new efficient automatic checker for security protocol analysis. In *Proc. of 12th Computer Security Foundation Workshop*, pages 192–202, 1999.