Sindh Univ. Res. Jour. (Sci. Ser.) Vol.44 (3) 391-394 (2012)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Heuristic for Directed Model Checking of Security Protocols

Q.U.A.NIZAMANI, H.A. NIZAMANI, F.H. CHANDIO, S.N. JAT, I.A. KOREJO

Institute of Maths and Comp. Science, University of Sindh, Jamshoro, Pakistans

Email: hanizamani@gmail.com : chandiofida@gmail.com sadafnasimch@hotmail.com cskorejo@gmail.com

Received 02th June 2012 and Revised 22th July 2012

Abstract: This paper introduces a heuristic to mitigate the state space explosion problem inherent in model checking of security protocols. The heuristic is property-dependent as it derives the hints from the security property to be verified. The heuristic has been implemented in a tool and the preliminary results show the effectiveness of the heuristic. The heuristic extends our previous work in which we suggested that directed model checking can be effectively used for security protocol verification.

Keywords: Model Checking, Heuristics, Security Protocols.

1. INTRODUCTION

In this paper we give account of the use of directed methods in the model checking of security protocols. Our main contributions are the definition of a property-dependent heuristic for security protocols, and the experimental results obtained by analyzing a few protocols.

The verification of security protocols is a rather challenging problem: despite their specification is typically rather compact, the attacker model at hand and the non-compositionality of the correctness of security protocols make their analysis rather complex. More precisely, the analysis typically considers executions where a protocol runs in parallel with an attacker that, depending on the model, can introduce a high degree of non-determinism. For instance, the so-called Dolev-Yao (Dolev and Yao, 1998) can interfere in all communications and possibly sends an infinite set of messages to honest participants. Typically the infinite state space originating in the Dolev-Yao model can be tackled by resorting to symbolic model-checking.

However, symbolic state spaces may still grow exponentially with the number of participants one has to consider in the analysis. In fact, flaws in the protocol can be found when a number of participants are considered, but they are not present for smaller numbers. Also, such number cannot be decided in general. To our knowledge, very few attempts have been made to tame this problem using directed model-checking

Technically, our heuristic method is applied to a symbolic model checker hinging on a process calculus and on an ad-hoc logic introduced in (Ferrari *et al.*, 2008). The calculus (called cIP after cryptographic interaction pattern) is used to model protocols and its operational semantics yields the state space upon which security properties expressed in the logic (called PL after protocol logic) are checked. Although the heuristic is defined for cIP and PL, we argue that they are rather general and amenable to be adapted to other frameworks. For instance, it would be straightforward to adapt our heuristic to a framework based on correspondence assertions as e.g. in (Boreale and Buscemi, 2005).

We have implemented our heuristic by integrating them into ASPASyA (Baldi *et al.*, 2005), a symbolic model-checker based on cIP and PL. The experimental results presented here are rather encouraging and show that directed methods provide substantial improvements when applied to the model checking of security protocols.

This hints that further improvements could be obtained by exploiting more sophisticated heuristics. For instance, one could define general heuristics about the choice of messages the attacker sends to the protocol so to further limit the size of the state space

2. BACKGROUND

In this section, we introduce the reader with a process calculus called cIP (cryptographic interaction pattern) and a logic called PL (protocol logic). The former is used to formalize the security protocols and latter for specifying the security property to be verified. The calculus and the logic were introduced in (Ferrari et al., 2008), where the details can be found.

We use the example of Wide-mouthed frog symmetric key protocol to explain the key aspects of cIP. The informal notation of the protocol is given below.

i) $A \rightarrow S : A, \{T_a, B, K_{ab}\}_{Kas}$

ii) $S \rightarrow B : \{ T_s, A, K_{ab} \}_{Kbs}$

In step i) the initiator *A* sends his identity and a cryptogram encrypted by the secret key shared between A and S to the server S. The cryptogram comprises of three components, a timestamp T_a, the identity of the responder B, and the session key K_{ab} to be used for the secure communication between A and B. The server checks the timestamp and then forwards the session key

++ Corresponding author: Q.U. A. NIZAMANI: Email: q_agro@hotmail.com

together with the identity of A and his own timestamp encrypted by the secret key (shared between S and B)

The formalized protocol in cIP is given below.

A : (x,s) [out (A, {na, (x, kab)}$_s$)]
S : (u, a, v, b) [in(u, {?t, (v, ?r)}$_a$).out({(ns, u, r)}$_b$) (1)
B : (s)[in({(?t, ?z, ?r)}$_s$)]

A principal in cIP is represented by his identity i.e., the name of the principal, a list of open variables (open variables are used to share the initial conditions such as identities and the symmetric keys), and the actions performed by the principals. For instance in (1), the initiator, the responder, and the server are represented by A, B, and S respectively. Each participant may have an associated list of open variables. For example, principal A has open variable *x* for sharing the identity of the responder and *s* for sharing the symmetric key shared between A and S. The actions performed by the participants can be *out(M)* and *in(M).* An action sends a message on a public channel. For instance, in WMF protocols A sends out a message out (A, {na, (x, kab)}$_s$) , to a public channel. An *in* action specifies the pattern that is expected to be received by the participant. In our running example, S is expected to receive a message of the form in(u, {?t, (v, ?r)}$_a$. Thus any message that has two components where first component should be identity of a participant and the second components must be a cryptogram comprising of three components (a nonce, an identity and any other message) and encrypted by the symmetric key shared between A and S will be accepted by S. On successful matching of the pattern, the variables will be instantiated with the received values.

The semantics of *cIP* is given as a labeled transition system. A state is a triple $\langle C, \chi, \kappa \rangle$ where *C* is a finite set of principal instances (called context), χ is a mapping of variables to messages, and κ is a set of messages κ that represent the intruder's knowledge. Transitions take the format:

$$t : \langle C, \chi, \kappa \rangle \xrightarrow{\delta} \langle C', \chi', \kappa' \rangle \qquad (2)$$

The label δ is a cIP action such as out(M), or in(M) or a join transition. A join transition can be defined as

$$t_{join} : \langle C, \chi, \kappa \rangle \xrightarrow{join\ S_i} \langle C', \chi', \kappa' \cup \{S_i, S_{i^+}\} \rangle$$

Informally, when a principal instance S_i joins a session, his identity and the public key is included to the intruder knowledge.

We give the definition of PL and refer the reader to (Ferrari et al., 2008) for detailed description:

$\varphi, \psi ::= xi = m \mid \kappa \rhd m \mid \forall i : A.\psi \mid \exists i : A.\psi \mid \neg\psi \mid \psi \wedge \varphi \mid \psi \vee \varphi$

The formulae $x_i = m$ and $\kappa \rhd m$ are called atomic formulae and hold respectively when the variable x_i is assigned the message m and when the intruder knowledge κ contains *m*. Notice that quantification is over indexes which represents instances as *PL* predicates over such instances of the principals.

As an example, consider the following PL security formula for WMF protocol:

$$\Psi_{WMF} = \forall j : B.\ \exists l : S.\ \exists i : A\ (v_l = B_j \wedge u_l = A_i \wedge x_i = B_j) \Rightarrow (t_l = na_i \wedge t_j = ns_l \wedge r_j = kab_i)$$

Informally, Ψ_{WMF} states that for all instances j of *B*, there exists an instance of A and an instance of S such that if the initiator and responder for server are A and B respectively and if A is connected to B then nonce received by the server is the one sent by A , the nonce received by B is the one sent by S, and the session key received by B is the one sent by S.

Based on cIP and PL, a tool ASPASyA has been developed. ASPASyA allows the verifier to perform verification using varying number of instances, different security and join formula. A join formula specifies constraints on the principals for connecting to each other. For instance, consider following join formula for WMF protocol: (∃i: A. true) ∧ (∃j: B. true) ∧ (∃l: S. true) which specifies that we are interested only in those runs of verification in which there is at least an instance of A, an instance of *B* and an instance of S.

3. HEURISTIC H2

We introduce a heuristic called H_2 that relies on security formulae and open variables. The heuristic H_2 requires two inputs, a state and the disjunctive normal form (**DNF, cf., Definition 1**) of the negation of a formula and returns an integer.

Definition 1 (*Disjunctive Normal Form*) *A formula φ, is in Disjunctive Normal Form if it is of the form $\varphi \equiv \varphi_1 \vee \cdots \vee \varphi_n$ where each φ_i has the form $\psi_{1,i} \wedge \cdots \wedge \psi_{n,i}$ and each $\psi_{j,i}$ is (the negation of) an atomic formula.*

Before defining H_2 we give some auxiliary functions. Fix a state s = $\langle C, \chi, \kappa \rangle$

Definition 2 Given a formula φ of the form $\varphi_1 \wedge \cdots \wedge \varphi_n$ where each φ_i is in DNF, the function $H_\wedge$ (s, φ) is defined as

$$H_\wedge(s, \varphi) = \begin{cases} \max H_2(s, \varphi_i), & H_2(s, \varphi_i) \geq 0 \\ \min H_2(s, \varphi_i), & otherwise \end{cases}$$

In order to satisfy φ, a state must satisfy each

the maximum value if every invocation results into a positive value otherwise it returns the minimum value (and hence a negative value). Therefore a positive heuristic value for a state indicates that the state s satisfies φ and a negative value otherwise.

Definition 3 Given a formula φ of the form x = m, the function $H_{=}(s, \varphi)$ is defined as

$$H_{=}(s,\varphi) = \begin{cases} \nu_{(s,\varphi)} & \chi(x) = m \text{ or } (\chi(x) = x[\kappa] \text{ and } \kappa \rhd m) \\ -1 & \chi(x) \neq m \text{ or } (\chi(x) = x[\kappa] \text{ and } \kappa \not\rhd m \\ 0 & \chi(x) = \perp \end{cases}$$

where $\nu(s,\varphi)$ is the number of open variables in s and φ

$H_{=}(s,\varphi)$ ranks high states satisfying φ while statesviolating φ are assigned -1 and must be pruned. Thefinal case in $H_{=}(s,\varphi)$ deals with a state that does not instantiate x. Since, no hint is available to evaluate the state therefore a neutral value such is 0 is assigned to it.

Definition 4 Given a formula φ of the form x $\neq$ m the function $H_{\neq}(s,\varphi)$ is defined as

$$H_{\neq}(s,\varphi) = \begin{cases} \nu_{s,\varphi} & \chi(x) \neq m \text{ or } (\chi(x) = x[\kappa] \text{ and } \kappa \not\rhd m) \\ -1 & \chi(x) = m \\ \nu_{s,\varphi} - 1 & \chi(x) = x[\kappa] \text{ and } \kappa \rhd m) \\ 0 & \chi(x) = \perp \end{cases}$$

where $\nu_{s,\varphi}$ is the number of open variables in s and φ.

Cases 1, 2, and 4 in $H_{\neq}(s,\varphi)$ follow the same intuition as for $H \neq (s,\varphi)$ that a state satisfying φ should be ranked high. Noteworthy, case 3 evaluates a state where x is mapped symbolically. Since such state can neither be pruned nor ranked high, $H_{\neq}(s,\varphi)$ lowers its heuristic value by decrementing $\nu_{s,\varphi}$.

We now define H_2 that depends on functions given in Definition 2, 3, and 4.

Definition 5 (Heuristic H_2) Given a formula φ the heuristic function H_2 is defined ass

$$H_2(s,\varphi) = \begin{cases} \max \; H_2(s,\varphi_i), & \varphi \equiv \varphi_1 \vee \cdots \vee \varphi_n \\ H_{\wedge}(s,\varphi), & \varphi \equiv \varphi_1 \wedge \cdots \wedge \varphi_n \\ H_{=}(s,\varphi), & \varphi \equiv x = m \\ H_{\neq}(s,\varphi), & \varphi \equiv x \neq m \\ 1, & true \\ -1, & false \\ 0, & otherwise \end{cases}$$

The recursive application of H_2 over the formula φ along with s determines the heuristic value to be assigned to the state. We discuss each case and describe why a particular value has been chosen.

Case 1 is applied when φ is a disjunction, i.e., $\varphi \equiv \varphi_1 \vee \cdots \vee \varphi_n$. Each disjunct φ_i is evaluated individually and the maximum value is returned. Intuitively, a state satisfies φ when it satisfies at least one φ_i. Thus $H_2(s,\varphi)$ returns a positive value if, φ is satisfied. Hence, if $H_2(s, \varphi)$<0 then s can be pruned.

Case 2 is used when φ is a conjunction, i.e., $\varphi \equiv \varphi_1 \wedge \cdots \wedge \varphi_n$. The state must be model for all the conjuncts and is evaluated by $H_{\wedge}(s,\varphi)$ given in Definition 2.

Cases 3-7 are used when φ an atomic formula is. Cases 3 and 4 respectively deals with the atomic formula of the form x = m and x ≠ m and are evaluated by the functions $H_{=}(s,\varphi)$ and $H_{\neq}(s,\varphi)$ respectively. The other cases are trivial.

4. ANALYSIS OF THE RESULTS

The heuristic introduced in (**section 3**) has been integrated into ASPASyA and has been put under test to evaluate its effectiveness. We present here a few preliminary results to show that heuristic has a potential to reduce the state space and direct the search towards attack containing states.

The results are presented in (**Table 1**) and (**Table 2**). (**Table 1**) gives the results when the protocols are tested with a trivial join formula 'true'. (**Table 2**) gives the results for a specified join formula thatcan restrict the state space as described in (**Section 2**).

The first column of both tables lists the protocols that have used as test cases. Column no. 2 and 3 respectively gives the number of instances considered for verification and presence/absence of attacks. The last two columns give the number of states explored during complete state space exploration and for finding the first attack.

Table 1. Verification of Protocols with *True* Join

Protocol	Inst.	Attack	Complete state space		First Attack	
			A	H_2	A	H_2
DS	3	Yes	76273	31381	39438	26
Carlsen	2	No	9792	1142	NA	NA
NS	4	Yes	378265	375597	60192	57524

The results show that in the case of Denning-Sacco (DS) protocol (Denning and Sacco, 1981), ASPASyA explores 76273 states for complete state space exploration, whereas H_2 explores only 31381 states. This suggests that H_2 is able to prune 58% of the original state space. The results for discovering first attack for DS protocol are remarkable where only a tiny portion of the state space is explored to give the results.

Similarly in the case of Carlsen protocol (Carlsen, 1994), H_2 is able to prune 88% of the original state space. Since Carlsen is attack-free for the specified property, therefore the results for first attack will be same as complete state space. This is represented in the tables with an entry marked as NA.

The results are less significant in the case of Needham-Schroeder (NS) protocol (Needham and Schroeder, 1978). For complete state space, H_2 gives a very little efficiency over ASPASyA, while for first attack it gives only 4% improvement over ASPASyA.

Table 2. Verification of Protocols with specified Join

Protocol	Inst.	Attack	Complete state space		First Attack	
			A	H_2	A	H_2
BY	2	Yes	1292	440	335	179
Carlsen	2	No	534	534	NA	NA

(Table 2) gives the number of states explored when protocols are tested using a specified join formula. As given in **(Section 2)**, join formula is also a way to restrict the state space. A carefully designed join formula can reduce the number of states explored for finding attacks. The verification of Carlsen protocol reflects this fact. The join formula is appropriate therefore the number of states explored by ASPASyA and H_2 are same. However, note that in the case of BY (Beller and Yacobi, 1993), H_2 gives better results as compared to ASPASyA. This shows that H_2 can be efficient even when a join formula is specified.

5. CONCLUSION

We have given a heuristic called H_2 for directed model checking of security protocols. The heuristic is property dependent, i.e., it uses the security formula to derive the hints and evaluate the states. Furthermore, the heuristic can also possibly prune certain parts of the state space, thus mitigating the problem of state space explosion The heuristic has been defined and implemented in terms of a symbolic model checker ASPASyA. The preliminary results show that heuristic provides efficiency in most of the cases as compared to ASPASyA.

Directed model checking has received attention in recent past for addressing the state space explosion problem. However, at the best of our knowledge, the study of DMC for security protocol is minimal to date. One of the early works in this direction is of (Basin, 1999), where some heuristics for directed model checking of security protocols have been suggested. However, the heuristics are very basic and as observed in (Basin *et al.*, 2003), they do not give much efficiency. The heuristic reported in (Eldekemp *et al.*, 2004, 2001) are efficient for general directed model checking. However, at the best of our knowledge, the heuristics in (Eldekemp *et al.*, 2001, Eldekemp *et al.*, 2003) allows pruning of state space for few trivial cases.

The heuristics suggested in (Gradara *et al.*, 2007) is relevant to our approach as the heuristics are property dependent and utilize the model and formula to rate the states.

Our work can be extended in many directions. We see the possibility of associating our work to other verification frameworks for security protocols; possibly we can define similar heuristics for other model checkers as most of the model checkers employ logic formula for specifying security property.

REFERENCES:

Baldi, G., A. Bracciali, G. Ferrari, and E. Tuosto, (2005) A Coordination-based Methodology for Security Protocol Verification, Electronic Notes in Theoretical Computer Science, **(121):** 23-46.

Basin, D. A. (1999) 'Lazy infinite-state analysis of security protocols, Proceedings of the International Exhibition and Congress on Secure Networking - CQRE (Secure '99), (1740) 30-42, UK.

Basin, D. A., S. Mödersheim, and L. Vigano, (2003) An on-the-fly model-checker For security protocol analysis, Proceedings of Esorics'03, LNCS**(2808):** 253-270, Heidelberg

Beller, M., and Y. Yacobi, (1993) Fully-fledged Two-way Public Key Authentication and Key Agreement for Low-cost Terminals. Electronics Letters, 29.

Boreale, M., and M. Buscemi. (2005) A Method for Symbolic Analysis of Security Protocols, Theoretical Computer Science 338 **(1-3):** 393-425.

Carlsen. U., (1994) Optimal Privacy and Authentication on a Portable Communications System, SIGOPS Operating System Review, 28.

Denning, D., and G. Sacco, (1981) Timestamps in Key Distribution Protocols,Communications of the ACM, 24.

Dolev, D., and A. Yao, (1998) On the security of public key protocols, IEEE Transactions on Information Theory 29 **(2):** 198-208.

Edelkamp, S., A. Lafuente and S. Leue, (2001) Protocol verification with heuristic search., AAA Symposium on Model-based Validation of Intelligence,75-83.

Edelkamp, S., S. Leue, and A. Lluch-Lafuente, (2004) A. Directed explicit-state model checking in the validation of communication protocols, International Journal of Software Tools and Technology. Transfer. 5 **(2):** 247-267.

Ferrari, G., and A. Bracciali, (2008) A symbolic framework for multifaceted security protocol analysis. International Jour. of Information Security 7 **(1):** 55-84.

Gradara, S., A. Santone, and M.L. Villani, (2007) Formal verification of concurrent systems via directed model checking, Electronic Notes in Theoretical Computer Science **(185):** 93-105.

Kupferschmid, S., K. Draäger, J. Hoffmann, B. Finkbeiner, H. Dierks, and G. Behrmann, (2007) UPPAAL/DMC-abstraction-based heuristics for directed model checking, TACAS, 679-682.

Needham R., and M. Schroeder, (1978) Using Encryption for Authentication in Large Networks of Computers, Communications of ACM, **(2):** 993–999.

Nizamani, Q., and E. Tuosto, (2009) Heuristic Methods for Security Protocols, Electronic Proceedings in Theoretical Computer Science **(7):** 61-75.