A Comparative Analysis of State-Space Tools for Security Protocol Verification

KIERAN HEALY
TOM COFFEY
REINER DOJEN
Data Communications Security Laboratory
Department of Electronic and Computer Engineering
University of Limerick
IRELAND
kieran.healy@ul.ie, tom.coffey@ul.ie, reiner.dojen@ul.ie, http://www.ece.ul.ie

Abstract: - State-space techniques have been proved successful as a means for finding attacks on security protocols. As well as reproducing known flaws, state-space techniques has also been used to discover new flaws in protocols In this paper, a brief overview of current state-space techniques is given and a selection of state-space tools for verifying cryptographic protocols are described, indicating the benefits, limitations and scope of application of each tool.

Key-Words: - Cryptography, formal verification, cryptographic protocols, state-space tools.

1 Introduction

A security protocol is a procedure designed to provide services such as authentication and confidentiality.

A wide variety of formal techniques have been developed to analyse security protocols. Approaches such as logics of knowledge and belief, checking of state-space for attacks and using various theorem proving techniques have been used for security protocol analysis. This paper presents a comparative analysis of state-space tools for security protocol verification.

State-space techniques have been proved successful as a means for finding attacks on security protocols. As well as reproducing known flaws, state-space techniques has also been used to discover new flaws in protocols [1], [2], [3], [4]. State-space techniques are much better suited to finding attacks on protocols than to proving protocols correct.

2 Concept of State-Space Verification

State-space techniques for security protocol verification involve building a model of a system and checking that each reachable state satisfies certain properties. This means designing a model of a general system running a security protocol in adverse conditions, and searching for states in which there is a security violation.

When searching the state-space, the initial state can be either a secure state or an insecure state. In the former case, all possible states that can be reached from this state are searched for and tested for security violations. In the latter case a backwards search is performed in order to attempt to reach an initial state.

Assumptions are made about what knowledge each participant in the protocol has in the initial state. The system models the messages sent and received during the protocol run. This includes messages the intruder send, receives or blocks. During this process the state-space verification system automatically builds the entire state-space. As each state is created it is checked to ensure it is secure.

Limits have to be put on the number of protocol runs and the number of participants to ensure that the state-space doesn't grow too large to be manageable. This problem of the state-space growing too large is known as the state explosion problem.

The process outlined above will never terminate for an infinite-state model. Since a protocol is required to be secure with an arbitrarily large number of agents engaged in an arbitrarily large number of concurrent runs of the protocol, the full generality of the protocol cannot be captured with a finite-state model. Several approaches have been proposed to address the state-explosion problem, including symbolic verification, partial-order methods and symmetry

methods. Although these approaches have reduced the size of the state-space by several orders of magnitude, the state-space of many protocols is still too large to be handled. State explosion remains a fundamental problem in state-space-based verification techniques.

2.1 Analysing Protocols Generating Infinite State-Spaces

Some state-space methods work in a potentially infinite state-space. In these cases a finite subset of the state-space is analysed. During the process of generating the state-space from a protocol, the problem of infinite state-space may occur in any of the following cases [5]:

- Some participants are able to perform unbounded steps in the protocol.
- Some participants may enter into infinitely many states.
- The set of participants is infinite.
- The set of messages received by a participant is infinite.

Sometimes, even if the system itself is finite, a state-space too large to be handled may be generated. When the size of a system increases linearly, the state-space may increase exponentially. A set of special concepts, as outlined above, can be used to keep the state-space under consideration manageable.

2.2 Analysing Protocols Generating Finite State-Spaces

Finite-state verification methods are based on exhaustively simulating all the possible behaviours of a closed protocol model. The finite-state protocol verification approach uses verification assumptions, which limit the variety of attacks it can detect. Usually cryptographic algorithms used are assumed to be ideal and thus attacks based on cryptanalysis are left undetected. Also to keep the verification manageable only a small number of protocol participants and protocol runs are used. The level of abstraction used in the models is also quite high. Verification establishes the security of the protocol only under the assumptions used. If any of the assumptions do not hold–for example the number of principals is unbounded- then no conclusion can be derived about protocol security.

There have been several approaches for generating a finite-model from a protocol description [6], [7], [8], [9].

2.3 Properties of State-Space Techniques

There are a number of limitations and benefits in the state-space verification methodology:

Fixed number of protocol participants. The set of participants must be bounded otherwise the problem of infinite state may occur.

State explosion problem. This is a general problem with the finite-state analysis. It limits the complexity of protocols that can be analysed.

Specialized tools. There are a number of specialized tools available specifically for analysing cryptographic protocols.

Properties analysed. State-space tools can analyse properties of cryptographic protocols such as authentication, secrecy and non-repudiation properties.

3 State-space tools used for security protocol verification

There are a number of tools used for security protocol verification such as the NRL Protocol Analyser [10], FDR [6], Brutus [11], Athena [12], The Interrogator [13], Huima's model checker [5], Murφ [14]. The following gives a brief overview of some of the state-space tools with more recent publications.

3.1 The NRL Protocol Analyser

The NRL Protocol Analyser [10], developed by the US Naval Research Laboratory, is a specialized tool for analyzing security protocols. The user of the Analyser provides a description of the protocol. The user also specifies an insecure state - that is, a state of the system in which the security specification of the protocol has been violated. The Analyser works backwards to find a path to the initial state. If such a path is found, then the reverse of this path will correspond to an attack on the protocol.

The program allows an arbitrarily large number of concurrent runs of the protocol and so the number of states can be infinite. The Analyser therefore provides various mechanisms to allow the user to reduce the state-space to a finite size. The two most important of these are:

- Inductive proofs. The user can state and prove lemmas regarding the unreachability of certain infinite sets of states.

- Subset queries. The user can direct the analyser to analyse a subset of the state space.

The ability to search an infinite number of states makes it possible to use the NRL Protocol Analyser to verify security protocols in their full generality. However, in consequence, the tool is not fully automatic; it requires substantial human direction to complete a proof. In addition, the possibility of producing false attacks via subset queries means that any putative attack must be checked manually.

The NRL Protocol Analyser can be used to prove security properties of cryptographic protocols as well as locate security flaws and has been successful in doing both. In particular, it has been used to find previously unknown flaws in the Simmons Selective Broadcast Protocol [3] and the Burns-Mitchell Resource Protocol [4].

3.2 FDR

The Failures-Divergences Refinement Checker (FDR) [6] is a general-purpose verification tool which offers the choice of verification using any of the three models of Communicating Sequential Processes (CSP) [15]: Traces Refinement, Failures Refinement, and Failures-Divergences Refinement. FDR is based on a machine-readable form of CSP. It is possible to use CSP to specify a protocol, a network and a general intruder. The traces model of FDR is used to check whether the combined system meets a given specification. If the system does not meet the specification, then a trace of the system is output, which corresponds to an attack on the protocol

CSP is a language which allows the modeling of the communication of an inherently asynchronous composition of protocol sessions. Each instance of an agent trying to execute the protocol is modeled by a CSP process that alternates between waiting for a message and sending a message (replying). Channels are used for communication between processes (between participants in the protocol).

In FDR the user had to provide a description of the adversary. With the development of Casper [16], which is a front-end to FDR, the construction of the adversary is automated.

FDR has been successfully used to analyse security protocols [1], [2], [6], [7].

3.3 Brutus

Brutus [11] is a special-purpose tool for analyzing security protocols. Encoding a protocol in Brutus is comparatively easy because the intruder model is an integral part of the tool. Brutus also suffers from the problem of state-space explosion. Because only a finite number of states can be checked, a full proof of correctness of the protocol cannot be obtained using Brutus.

Brutus attempts to separate out the adversary from the model. Instead, the adversary is encoded as a set of rewrite rules that can be applied to messages sent during the execution of the protocol. In essence, Brutus has two orthogonal components. One is a state exploration component that actually performs the search. The other is the message derivation engine that uses the rewrite rules to model the adversary's capabilities. These components interact. The set of possible next states is determined by the messages the adversary can construct and send. In turn, the set of messages the adversary can generate is determined by the messages other agents have sent during the execution of the protocol.

Since Brutus was designed to analyse a variety of protocols, a specification language was developed which is powerful enough to describe a variety of security properties. In this language the user can specify what information participants (including the adversary) must know and should not know.

One of the advantages of Brutus is the fact that it has a built in model of the adversary. This is particularly useful when one wants to analyse possible interactions between different protocols. Because the adversary is built in, one does not need to anticipate what messages or submessages might interact. Brutus will catch those possible interactions automatically.

While analyzing protocols such as, Kerberos, TMN protocol, Otway-Rees, Needham-Schroeder-Lowe public-key protocol, all previously known attacks were found. However, Brutus did not find any new attacks.

3.4 Athena

Athena [12] is a tool aimed at providing a mechanism for automatic protocol analysis within the strand-space model [17]. It combines various techniques to allow for automatic verification with no bounds on the size of the network. Athena uses a backward search from an insecure state, starting with only a small state-space, dynamically increasing the size of the state-space where

necessary. Its state-space exploration is symbolic, allowing it in some cases to compress infinite classes of states onto a single state. In addition, lemmas can be used to discard certain classes of unreachable states.

Athena uses an extension of the strand space model to represent protocol execution and utilizes techniques from both model checking and theorem proving approaches. Athena incorporates a language that can express security properties including authentication and secrecy. Athena user the strand-space model to analyse the protocol according to its security specification. If a breach of the security specifications is found Athena returns a trace that corresponds to an attack.

Athena also exploits several state-space reduction techniques, such as backward search and symbolic representation. Hence it avoids the state-space explosion problem commonly caused by asynchronous composition and symmetry redundancy. A number of protocols have been analysed using Athena and previously known attacks have been found.

3.5 Summary

In this paper a number of state-space tools have been examined. Table 1 gives a summary of these tools. The ability to search an infinite number of states makes it possible to use the NRL Protocol Analyser to verify security protocols in their full generality. It can be used to analyse security properties of cryptographic protocols as well as locate security flaws. It has been used to find previously unknown flaws in protocols. The analyser provides various mechanisms to allow the user to reduce the state-space to a finite size. The two most important of these techniques are inductive proofs and subset queries. The analyser uses backward search from an insecure state and thus attempts to find a path to a valid initial state. The reverse of this path will correspond to an attack on the protocol.

FDR is a general-purpose tool that can handle a wide range of protocols. If the system does not meet the specification, then a trace of the system is output, which corresponds to an attack on the protocol. FDR has been used extensively to analyse security protocols, with a number of attacks having been discovered. Although it has proved its worth in finding attacks, it suffers from being able to explore only finite-state models.

Brutus was originally developed to analyse authentication protocols. Brutus proved capable of specifying and verifying secrecy, authorization and non-repudiation properties. The number of agents and the number of concurrent runs of the protocol is severely limited if the analysis is to be completed in a realistic amount of time. It uses partial order and symmetry reduction techniques. A number of protocols have been analysed using Brutus. While all previously known attacks have been found, no new attacks were discovered.

Athena can verify security properties of a protocol for arbitrary protocol configurations, and in particular, for an unbounded number of concurrent runs. Athena incorporates a language that can express security properties including authentication, secrecy and properties related to electronic commerce. If a breach of the security specifications is found Athena returns a trace that corresponds to an attack. A number of protocols have been analysed using Athena and previously known attacks have been found.

4 Conclusion

This paper presented a brief overview of current state-space techniques for security protocol verification. The benefits and limitations of these techniques were outlined. A comparative analysis of four state-space based verification tools was detailed.

State-space techniques have been proved successful as a means for finding attacks on security protocols. As well as reproducing known flaws, state-space techniques have also been used to discover new flaws in protocols.

State-space techniques are better suited to finding attacks on protocols than to proving protocols correct. There are a number of limitations to state-space techniques, however the limitations should be weighted against the benefits of finding flaws in cryptographic protocols.

	State-Space	Reduction Techniques	Examples of Protocols Analysed	Flaws Detected
NRL	Infinite	Inductive proofs + subset queries	Authentication and key management protocols	New flaws discovered and existing flaws reproduced [3] [4]
FDR	Finite	Implicit model-checking	Key exchange protocols	New flaws discovered and existing flaws reproduced [1] [2] [6] [7]
Brutus	Finite	Partial order + symmetry	Authentication, and non-repudiation properties	Existing flaws reproduced [11]
Athena	Infinite	Backward search + symbolic representation	Authentication, secrecy and properties related to electronic commerce	Existing flaws reproduced [12]

Table 1. Summary of State-space tools for security protocol verification

References:

[1] Meadows C., Analyzing the Needham-Schroeder public-key protocol: A comparison of two approaches, *In proceedings of ESORICS '96*, Vol. 1146 of Lecture Notes in Computer Science, Springer-Verlag, 1996, pp 351-364.

[2] Lowe G. and Roscoe A. W., Using CSP to detect errors in the TMN protocol, Technical Report 1996/34, Department of Mathematics and Computing Science, University of Leicester, 1996.

[3] Meadows C., A System for the Specification and Analysis of Key Management Protocols, In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, IEEE Computer Society Press, Los Alamitos, California, 1991, pp 182-195.

[4] Syverson P. and Meadows C., Formal requirements for Key distribution protocols, *Advances in Cryptology - EUROCRYPT'94*, vol 950 of lecture notes in computer science, Spinger Verlag, Berlin, 1995.

[5] Huima A., Efficient infinite-state analysis of security protocols, presented at *FLOC'99 Workshop on Formal Methods and Security Protocols*, July 1999.

[6] Lowe G., Breaking and fixing the Needham-Schroeder public-key protocol using FDR, In *Tools and Algorithms for the Construction and Analysis of Systems*, No. 1055 in Lecture Notes in Computer Science, Springer-Verlag, 1996, pp 147-166.

[7] Roscoe A. W., Modelling and verifying key-exchange protocols using CSP and FDR, In *Proceedings of 8th IEEE Computer Security Foundations Workshop*, IEEE Computer Society Press, 1995, pp 98.

[8] Shmatikov V. and Stern U., Efficient finite-state analysis for large security protocols, In *Proceedings of 11th IEEE Computer Security Foundations Workshop, Rockport, MA*. IEEE Computer Society Press, 1998.

[9] Mitchell J. C. Mitchell M. and Stern U., Automated analysis of cryptographic protocols using Murφ, In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, IEEE Computer Security Press, 1997, pp 141-151.

[10] Meadows C., The NRL Protocol Analyser: an overview, *Journal of Logic Programming*, February 1996.

[11] Clarke E. M. Jha S. Marrero W., Verifying security protocols with Brutus, ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 9 no. 4, Oct. 2000, pp.443-487

[12] Song D. Berezin S. and Perrig A., Athena: a novel approach to efficient automatic security protocol analysis, *Journal of Computer Security*, 2001

[13] Millen J. Clark S. and Freedman S., The Interrogator: Protocol security analysis, *IEEE Transactions on Software Engineering*, February 1987, pp. 274-288.

[14] Dill D. L., The Murφ verification system. *In Computer Aided Verification. 8th International Conference*, 1996, pp 390-3,
[15] Hoare C. A. R., *Communicating Sequential Processes*, Prentice-Hall International, 1985.
[16] Lowe G., Casper: A compiler for the analysis of security protocols, In *Proceedings of the 1997 IEEE Computer Society Symposium on Research in Security and Privacy*, 1997, pp 18-30,
[17] Thayer F. J. Herzog J. C. and Guttman J. D., Strand spaces: Why is a security protocol correct?, In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, 1998.