

# Slede: Framework for Automatic Verification of Sensor Network Security Protocol Implementations\*

Youssef Hanna  
Iowa State University  
ywhanna@cs.iastate.edu

Hridayesh Rajan  
Iowa State University  
hridayesh@cs.iastate.edu

## Abstract

*Verifying security properties of protocols requires developers to manually create protocol-specific intruder models, which could be tedious and error prone. We present Slede, a verification framework for sensor network applications. Key features include automation of: extraction of models, generation and composition of intrusion models, and verification of security properties.*

## 1. Introduction

A *sensor network* is a collection of small, low power, low-cost sensor nodes that have limited computational, communication and storage capacity. These nodes can operate unattended, sensing and recording detailed information about their surroundings. Applications of sensor networks include civil and military applications such as target tracking, remote surveillance, and habitat monitoring. The operating environments of sensor networks are often hostile, requiring mechanisms for secure communication. A number of security protocols for sensor networks have been proposed in the past decade.

Establishing the correctness of the security protocols has been a daunting task. Formal methods have been widely used to verify security protocol specifications; however, building models is a time consuming task, requires expertise in modeling languages, which may prevent domain experts from attempting such tasks, and the abstraction required to build the model may abstract some of the details of the implementation that may contain security flaws, which in turn will go undetected.

While there exist many frameworks for applying formal methods on program implementations such as Bandera [3], CMC [9], etc, the main goal of such frameworks is to detect bugs such as deadlocks. Applying the same approach to verify security features of security protocol implemen-

tations requires the presence of protocol specific malicious activity against which the verification is done.

To be able to launch attacks on different protocol implementations, the intruder model needs to be aware of different types of messages exchanged between principals in the protocol implementation and the structure of each message i.e. which fields in the message represent information about sender, receiver, data, etc. This way, the intruder can read the message contents and launch attacks such as corrupting data of the message and forwarding the corrupt message. On the other hand, if the intruder model is not protocol specific, it will not know which part of the message is data, sender, etc. Hence, to launch attacks, it will randomly try to change the bits of the message, which will dramatically increase the time of the verification process, or the verification may never halt due to the huge number of possible interactions caused by the change of every bit by the intruder.

The problem with manually writing protocol specific intruder models for every protocol is that, similar to the problem of manually writing protocol models, it can be a tedious task and error prone.

We recently designed a framework [6] for automatic verification of Wireless Sensor Network (WSN) security protocol implementations. The framework *Slede* is able to generate protocol specific intruder models from the protocol implementations, extract the protocol model from the implementation and verify security properties against these extracted models.

The main contribution of *Slede* is the reduction of the cost of verification of security protocol implementations by providing network developers with a tool that automatically verifies the implementation of their networks at a reduced cost of learning a small lightweight specification language similar to the implementation language. *Slede* eases the application of formal methods by extracting the model from the implementation, automatically generating protocol-specific intruder models, verifying the security properties against the composition of the models and providing the counterexamples for property violations in terms of the domain language.

\*Tool available at <http://www.cs.iastate.edu/~slede>

## 2. Slede Framework

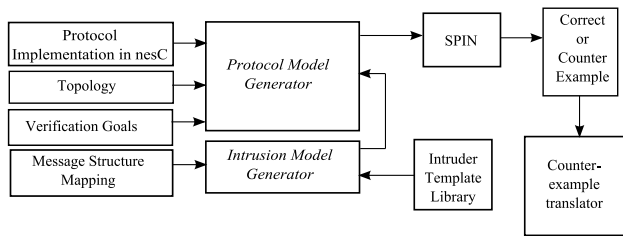


Figure 1. Overview of *Slede*

Our framework provides automatic verification of sensor network security protocols by extracting models from the protocol implementation. In addition, the framework automatically generates intruder models that are necessary for verification of the security protocols. It is built on top of the nesC compiler version 1.1.1 [10] and uses the Spin model checker [8] as the backend.

**Overview:** The overview of *Slede* is shown in Figure 1. The framework takes the source code of the protocol as input. In order to verify a protocol implementation, besides the source code, the framework requires a small amount of extra information such as the structure of messages used in the protocol, a deployment topology, and properties that need to be verified about the protocol. This information is provided using our annotation language in a different file with extension `sld` [5]. The generated protocol model is merged with the generated intruder model and then verified using Spin. If there is a violation of the protocol objective, the counterexample generated by Spin is translated into nesC statements using the counterexample translator.

## 3 Related Work

The closest work related to our approach is by Bhargavan *et al.* [1] and by Goubault-Larrecq and Parrennes [4]. Bhargavan *et al.* [1] present an approach for verifying protocol implementations written in F# using ProVerif [2], a theorem prover as the underlying mechanism. Our work is different in that the intruders are automatically generated to be protocol specific using the information provided by the annotation language, while in their case the intruder has to be implemented as a program by the user according to an *attacker interface*, which creates an overhead on the user and makes detection of security breaches dependent on how well the developer has implemented the intruder.

Goubault-Larrecq and Parrennes [4] present an approach for verifying protocol implementations in C. Their approach models secrecy properties as reachability properties of the C implementation and analyzes these properties. Unlike our

approach that provides support for the entire nesC language, this approach is useful only for C implementations; however, the insights described by Goubault-Larrecq and Parrennes could be used to enhance the underlying verification technique for our framework.

Tools for model checking source code directly are also related to this work. In particular, Bandera [3], Java Path Finder [7], CMC [9], etc, have successfully verified C and Java implementations. Similar to these approaches, our framework also verifies source code directly; however, unlike these techniques our framework is concerned with security properties as opposed to deadlock and bug detection.

## 4. Evaluation

We verified two sensor network security protocol implementations using *Slede* [6]. *Slede* was able to generate protocol specific intruder models of these protocol implementations. Verification of the extracted protocol model and generated intruder models detected known security breaches in these two protocol implementations.

**Acknowledgements:** This work is supported in part by the NSF under grant CNS-06-27354.

## References

- [1] K. Bhargavan, C. Fournet, A. D. Gordon, and S. Tse. Verified interoperable implementations of security protocols. In *CSFW '06*, pages 139–152, Washington, DC, USA, 2006.
- [2] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *CSFW-14*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001.
- [3] J. C. Corbett, M. B. Dwyer, J. Hatcliff, S. Laubach, C. S. Preanu, Robby, and H. Zheng. Bandera: extracting finite-state models from java source code. In *ICSE '00*, pages 439–448, New York, NY, USA, 2000. ACM Press.
- [4] J. Goubault-Larrecq and F. Parrennes. Cryptographic protocol analysis on real C code. In R. Cousot, editor, *VMCAI'05*, volume 3385 of *Lecture Notes in Computer Science*, pages 363–379, Paris, France, Jan. 2005. Springer.
- [5] Y. Hanna and H. Rajan. Slede website. <http://www.cs.iastate.edu/~slede/>, 2007.
- [6] Y. Hanna, H. Rajan, and W. Zhang. Slede: A domain-specific verification framework for sensor network security protocol implementations. In *WiSec '08*, pages 109–118, 2008.
- [7] K. Havelund and G. Rosu. Monitoring Java programs with Java PathExplorer. In K. Havelund and G. Roşu, editors, *Runtime Verification*, volume 55 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 23July 2001.
- [8] G. J. Holzmann. The model checker SPIN. *IEEE Trans. Softw. Eng.*, 23(5):279–295, 1997.
- [9] M. Musuvathi, D. Y. W. Park, A. Chou, D. R. Engler, and D. L. Dill. Cmc: a pragmatic approach to model checking real code. *SIGOPS Oper. Syst. Rev.*, 36(SI):75–88, 2002.
- [10] nesC Compiler. <http://sourceforge.net/projects/nesc>.