
In a world that increasingly relies on the Internet to function, application developers rely on the implementations of protocols to guarantee the security of data transferred. Whether a chosen protocol gives the required guarantees, and whether the implementation does the same, is usually unclear. The Guided System Development framework contributes to more secure communication systems by aiding the development of such systems. The framework features a simple modelling language, step-wise refinement from models to implementation, interfaces to security verification tools, and code generation from the verified specification. The refinement process carries thus security properties from the model to the implementation. Our approach also supports verification of systems previously developed and deployed. Internally, the reasoning in our framework is based on the Beliefs and Knowledge tool, a verification tool based on belief logics and explicit attacker knowledge.
On Building Secure Communication Systems

This thesis presents the Guided System Development (GSD) framework, which aims at supporting the development of secure communication systems.

A communication system is specified in a language similar to the Alice and Bob notation, a simple and intuitive language used to describe the global perspective of the communications between different principals. The notation used in the GSD framework extends that notation with constructs that allow the security requirements of the messages to be described.

From that specification, the developer is guided through a semi-automatic translation that enables the verification and implementation of the system. The translation is semi-automatic because the developer has the option of choosing which implementation to use in order to achieve the specified security requirements. The implementation options are given by plugins defined in the framework. The framework’s flexibility allows for the addition of constructs that model new security properties as well as new plugins that implement the security properties.

In order to provide higher security assurances, the system specification can be verified by formal methods tools such as the Beliefs and Knowledge (BAK) tool — developed specifically for the GSD framework —, LySatool and OFMC. The framework’s flexibility and the existence of the system model in different perspectives — an overall global perspective and an endpoint perspective — allow the connection to new formal methods tools.

The modeled system is also translated into code that implements the communication skeleton of the system and can then be used by the system designer. New output languages can also easily be added to the GSD framework.

Additionally, a prototype of the GSD framework was implemented and an example of using the GSD framework in a real world system is presented.
Using Dafny, an Automatic Program Verifier
These lecture notes present Dafny, an automated program verification system that is based on the concept of dynamic frames and is capable of producing .NET executables. These notes overview the basic design, Dafny’s history, and summarizes the environment configuration. The key language constructs, and various system limits, are illustrated through the development of a simple Dafny program. Further examples, linked to online demonstrations, illustrate Dafny’s approach to loop invariants, termination, data abstraction, and heap-related specifications.

Protocol Implementation Generator
Users expect communication systems to guarantee, amongst others, privacy and integrity of their data. These can be ensured by using well-established protocols; the best protocol, however, is useless if not all parties involved in a communication have a correct implementation of the protocol and all necessary tools. In this paper, we present the Protocol Implementation Generator (PiG), a framework that can be used to add protocol generation to protocol negotiation, or to easily share and implement new protocols throughout a network. PiG enables the sharing, verification, and translation of communication protocols. With it, partners can suggest a new protocol by sending its specification. After formally verifying the specification, each partner generates an implementation, which can then be used for establishing communication. We also present a practical realisation of the Protocol Implementation Generator framework based on the LySatool and a translator from the LySa language into C or Java.
Period: 01/09/2010 → 22/11/2013
Number of participants: 6
Phd Student:
Carvalho Quaresma, Jose Nuno (Intern)
Supervisor:
Nielsen, Flemming (Intern)
Main Supervisor:
Probst, Christian W. (Intern)
Examiner:
Dragoni, Nicola (Intern)
Gollmann, Dieter (Ekstern)
Kammüller, Florian (Ekstern)

Financing sources
Source: Internal funding (public)
Name of research programme: Institut stipendie (DTU)
Project: PhD