Motivation

Nowadays our society becomes increasingly depended on computer systems. In more and more areas small computers take over the control. If it's our SmartTV, car or the control of the lights in our home. We are forced to confront our self with the topic how secure and reliable these systems are.
Especially if we entrust a computer our live this gets an essential meaning. We want to expect from board-computers in planes or cars that they are free from defects and not so easily hackable. This is not the reality. We know about cars whose board-computers simple can be taken over with a smartphone in the car next to it.
A key component in developing secure systems is the operating-system (OS) kernel of the system. The kernel has full access to hardware resources. One defect in the kernel can have the consequence that the security and reliability of the entire system can be lost.
The weakness of most previous kernels were their huge amount of code and mostly their monolithic design. This makes it impossible to review or verify the code. The weak point of the monolithic design is that not only fundamental functions as Interprocesscommunication, scheduling or memory management are implemented in the kernel mode but also functions like driver for hardware or virtual filesystems are integrated in it. This makes the system more vulnerable for bugs. One crashed modul can lead to a crash of the entire system.
The motivation behind microkernels is to reduce the possibility of bugs in the kernel code through reducing the kernelcode to an amount as minimal as possible and excluding functions from the kernel mode. With less code it becomes more feasible to guarantee the absence of defects within the kernel through formal verification.
Due to the fact that we feed our smartphones, tablets, board-computers, ... with growing amounts of sensitive informations like bank data, passwords, e-mails, chats, the significance of Security in the area of embedded systems increases.
Through isolation of small subsystems, like it is done in microkernels, the security already can be raised to a higher level. With testing one can depict an huge amount of bugs. But as Dijkstra said "Testing can only show the presence, not the absence, of bugs." [p. 16 of http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1969.PDF]
Like I already mentioned less lines of codes make it more feasible to verify it relating to its specification.
The seL4 microkernel is the first microkernel whose correctness is formally verified. It's a high-assurance, high-performance microkernel, primarily developed, maintained and formally verified by NICTA (now Trustworthy Systems Group at Data61) for secure embedded systems. It's security model is based on the take-grant model, which was extended for being able to reason about kernel memory consumption of components.

Aim of the thesis

With this thesis I want to show or disprove that the extended take-grant model is strong enough to show the noninterference property on it. The security property of noninterference ensures that there is no unwanted information flow within a system. The take-grant model is an access control model. Therefore its duty is to "control" the access or the transfer of access on objects of a system. The noninterference property assured that there is no way information can flow to undesirable parties. Also with information about third parties that have access.
The thesis should investigate the different systemoperations of the model regarding the thereby occurring information flow. With the collected information I want to decide two questions. First if the noninterference property can be verified with the existing model and second if the noninterference property is fulfilled.

Structure of the thesis

First I want to give an overview of the seL4 kernel, his set-up, the implementation of services and the memory management. For a better comprehension I then give a brief overview of the take-grant and the noninterference model.
In chapter 3 the formalisation of the take-grant model takes place and in chapter 4 it's the formalisation of the noninterference model.
From chapter 5 on I dedicate myself to the validation of the noninterference property. In chapter 7 the validation is subdivided into the different systemoperations. To show the property for the model I have to extend the model in chapter 6.
Finally I'll take a short resume and give a prospect on the current status of the seL4 project and the possibilities to enhance this topic.