Todo list

#1 Kristian: unrealistic 1
#2 Kristian: does not depent on the amount of time the attacker has but the fact that the smart card is inactive and he can only target constant values . 1
#3 Christoffer: do we have to choose one or do we analyse both? 1

Contents

1 Introduction **1**
1.1 Fault Scenarios 1
1.2 Java Card 2
1.3 Fault Injection 2

2 Program Analysis with Sawja **3**

3 Rewriting a Java Program **5**

4 UPPAAL and Formal Verification **7**
4.1 Property Verification 7

5 Building a UPPAAL Model **9**
5.1 Choosing a program representation 9

6 Conclusion **11**
6.1 Future Work 11

CHAPTER 1

Introduction

1.1 Fault Scenarios

We consider two general categories of faults that can occur to a Java Card: *persistent* and *transient* faults. The main difference between these is that the persistent faults will affect the program every run, while the transient faults will only be present for a limited amount of time. Faults can occur when hardware is exposed to radiation sources, e.g. infrared light, laser, heat, physical abuse or cosmic radiation from space. Persistent faults in a piece of hardware, such as system memory, can occur in several ways. One way is when a Java Card is exposed to physical abuse, such as a hammer hitting the card's chip to induce a fault . Since the structure of the chip is permanently altered, the fault is persistent, but the fault injection is not precise. Another way is a directed fault injection, e.g. a laser beam, targeting a persistent part of memory, such as the EEPROM of a Java Card. This could cause a bit flip in a value that is persistent across power-ups, and thus cause a persistent fault since the wrong value will always be used. If one wishes to create a persistent fault, precision is important, both to strike a persistent part of memory, but also to strike the correct value.

#1 Kristian: unrealistic

Transient faults do not cause any permanent damage to the hardware. They can cause a temporary bit flip, resulting in a corrupted value, changed control flow to cause unintended behaviour, or a crash of the hardware. The altered behavior will disappear and the fault injection will have to be performed again, if the effect is to be reproduced. Nonetheless, both persistent and transient faults can have fatal consequences, if they strike at the right time and the right place, e.g. for an attacker trying to change a programs control flow and thus execute a sensitive piece of code. The two categories of fault injection are thus sensitive to two variables, *time* and *place*, to different degrees. For example, an attacker who is trying to alter a constant in a program on a chipped access card, is able to work on the card in private surroundings. He can remove the protective layer on the chip and induce a persistent error at the right place at his leisure. He is therefore not affected by the timing of the fault introduced . The fault will still be present when he tries to use it later. On the other hand, an attacker who wants to change transient properties such as program flow dependent on a non-constant value, determined at run-time, is very dependent on both time and precision of his attack. He has to affect the correct place in memory at just the right time to alter for example program flow. Table T1-1 illustrates the dependencies of persistent and transient fault injections.

#2 Kristian: does not depent on the amount of time the attacker has but the fact that the smart card is inactive and he can only target constant values

	Persistent	Transient
Timing		X
Precision	X	X

Table T1-1: Table showing dependencies of induced faults

#3 Christoffer: do we have to choose one or do we analyse both?

1.2 Java Card

1.3 Fault Injection

CHAPTER 2

Program Analysis with Sawja

CHAPTER 3

Rewriting a Java Program

CHAPTER 4

UPPAAL and Formal Verification

4.1 Property Verification

CHAPTER 5

Building a UPPAAL Model

5.1 Choosing a program representation

5.2 Rewriting the Model

CHAPTER 6

Conclusion

6.1 Future Work