Modellbasiertes Testen der deutschen Gesundheitskarten

Christoph Apel[1], Jürgen Repp[2], Roland Rieke[2], Dr. Jürgen Steingruber[1]

[1]Giesecke & Devrient München
{christoph.apel | juergen.steingruber}@gi-de.com

[2]Fraunhofer-Institut für Sichere Informationstechnologie
{repp | rieke}@sit.fraunhofer.de

Zusammenfassung

Dieser Beitrag beschreibt die Anwendung eines modellbasierten Verfahrens zur Testfolgengenerierung auf die deutschen elektronischen Gesundheitskarten, Heilberufsausweise und Sicherheitsmodulkarten, um deren Datensicherheit, Interoperabilität und Robustheit zu testen. Zur Modellierung der Chipkartenanwendungen werden asynchrone Produktautomaten (APA) verwendet. Daraus werden abstrakte Kommandofolgen für die Chipkarten berechnet, die dann in einem weiteren Schritt in konkrete ausführbare Kommandos übersetzt werden. Das verwendete Verfahren hat den Vorteil, dass damit komplexe Chipkartenanwendungen in kompakter und übersichtlicher Weise modelliert und automatisiert Testfolgen mit hoher Testabdeckung erzeugt werden können.

1 Motivation

Da die manuelle Testerstellung für umfangreiche Spezifikationen sehr aufwändig und fehleranfällig ist, bieten modellbasierte Verfahren zur Testgenerierung eine interessante Alternative. Dabei werden die Anwendungen nach ihrer Spezifikation modelliert und geeignete Testkriterien definiert, um daraus automatisch Testfälle zu erzeugen. Dieses sogenannte modellbasierte Testen stellt sicher, dass die Testfälle mit der zugrundeliegenden Spezifikation konsistent sind und dass systematisch die verschiedenen Aspekte der Spezifikation abgedeckt werden.

Ein modellbasiertes Testverfahren wurde bei G&D mit Unterstützung durch Fraunhofer SIT am Beispiel der Chipkarten für das deutsche Gesundheitswesen implementiert. Das zu testende System besteht hier aus mehreren Chipkartentypen, die miteinander interagieren: die elektronische Gesundheitskarte (eGK), der Heilberufsausweis (HBA) und die Sicherheitsmodulkarte (SMC). Diese werden von verschiedenen

Krankenkassen, Ärzte- und Zahnärztekammern und Apothekerverbänden herausgegeben. Sie enthalten sensible Daten und besitzen Sicherheitsmechanismen, um diese zu schützen. Zudem müssen sie geeignet zusammenwirken können und auch auf eine fehlerhafte Benutzung definiert reagieren.

2 Testaspekte in Bezug auf die Chipkartensicherheit

Die Chipkarte dient als Behälter für geheime Schlüssel, die als Sicherheitsanker für den Zugriff auf sensible Daten dienen. Über diese geheimen Schlüssel werden Authentisierungen der Karten gegeneinander oder gegenüber anderen Instanzen abgewickelt. Zudem werden damit elektronische Signaturen realisiert. Eine erfolgreiche Authentisierung führt zum Setzen eines Sicherheitszustandes in der Chipkarte und öffnet einen sicheren Kanal, der den Zugriff auf sensible Daten erlaubt. An einer Authentisierung können auch mehrere Chipkarten gleichzeitig beteiligt sein. So gibt es z.B. ein Verfahren, wo sich zuerst eine HBA gegen eine SMC authentisiert, damit diese sich gegenüber einer eGK authentisieren kann.

Dieser Zugriff auf sensible Daten ist rollenabhängig, d.h. je nachdem welche Rollen die Authentisierungspartner innehaben, werden unterschiedliche Zugriffe erlaubt oder verboten. Sensible Daten auf der Chipkarte sind neben den Rezepten und Notfalldaten die Einwilligung zum Führen einer Patientenakte und der Schlüssel und das Pseudonym für den Zugriff auf eine Patientenakte.

Die hier entwickelten Tests sollen dabei die folgenden Sicherheitsaspekte abdecken:

1. Es soll das Setzen und Löschen aller Sicherheitszustände in allen Karten getestet werden.
2. In allen Sicherheitszuständen soll sowohl der erlaubte Zugriff, als auch der verbotene Zugriff für alle relevanten Kommandos und Files auf der Chipkarte getestet werden.
3. Es sollen alle Rollen berücksichtigt werden.
4. Es sollen alle möglichen korrekten Kommandoreihenfolgen getestet werden.
5. Zur Sicherstellung der Interoperabilität sollen sämtliche Kryptogramme und Zertifikate von den Chipkarten selbst produziert oder gelesen werden.

3 Asynchrone Produktautomaten

Die formale Modellierung der Chipkartenanwendung wurde auf Basis „Asynchroner Produktautomaten (APA)“, einer allgemeinen Klasse kommunizierender Automaten, durchgeführt. APA sind ein universelles und sehr flexibles Beschreibungsmittel für kooperierende Systeme [ORR00c], zu denen mit dem von Fraunhofer-SIT entwickelten SH

Verification Tool ein komfortables Werkzeug für Spezifikation und Analyse zur Verfügung steht [SHVT,ORR00b]. Ein Asynchroner Produktautomat APA kann als eine Familie von Elementarautomaten, auch Transitionen genannt (siehe Abschnitt. 4), angesehen werden. Die Zustandsmenge des APA ist als Produktmenge strukturiert, d.h. jeder Zustand besteht aus mehreren Zustandskomponenten. Die Zustandsmenge einer Zustandskomponente wird als Definitionsbereich der Zustands-komponente bezeichnet. Die Zustandsmengen einzelner Elementarautomaten bestehen wiederum aus Komponenten der Zustandsmenge des APA. Unterschiedliche Elementarautomaten können über gemeinsame Zustandskomponenten ihrer Zustandsmengen „verklebt“ sein. Elementarautomaten können kommunizieren, indem sie auf solche gemeinsamen Zustandskomponenten zugreifen.

Abbildung 2 zeigt die graphische Darstellung eines APA mit den 5 Elementarautomaten „Command“, „Response“, „Read_EF“, „Update_EF“ und „Verify_PIN“ und den Zustandskomponenten „Command“, „IN“, „Response“ und „State“. Kreise repräsentieren die Zustandskomponenten und Rechtecke die Elementarautomaten. Die Nachbarschaftsrelation spezifiziert, welche Zustandskomponenten im Zustand eines Elementarautomaten enthalten sind. Die Nachbarschaftsrelation wird durch Kanten zwischen Elementarautomaten und Zustandskomponenten dargestellt. Nur Zustände dieser Zustandskomponenten können bei einem Zustandsübergang eines Elementarautomaten geändert werden. In der Abbildung 2 kann zum Beispiel ein Zustandsübergang von „Verify_PIN“ nur die Zustände von „IN“, „Response“ und „State“ ändern.

Die graphische Darstellung des APA zeigt lediglich die Struktur aus Elementarautomaten, Zustandskomponenten und Nachbarschaftsrelation. Zur vollständigen Spezifikation eines APA müssen zusätzlich die Definitionsbereiche, Zustandsübergangsrelationen und der Initialzustand definiert werden. Ein Zustandsübergang ist dabei ein Tripel (Zustand, Elementarautomat, Folgezustand). Beginnend mit dem Initialzustand definiert die Zustandsübergangsrelation die möglichen Zustandsübergangsfolgen. Diese Folgen von Zustandsübergängen beschreiben das Systemverhalten. Der sogenannte „Erreichbarkeitsgraph“ eines APA ist nun ein gerichteter Graph, dessen Knoten die möglichen globalen Zustände und dessen Kanten die Zustandsübergänge des APA beschreiben. Eigenschaften dieses Graphen, und damit des dynamischen Systemverhaltens, können mittels der durch das SH Verification Tool bereitgestellten Verfahren (Abstraktionen, temporale Logik, etc.) untersucht werden. Vor der Berechnung des vollständigen Graphen ist es möglich, Simulationen durchzuführen.

4 Modellierung der Karten und des Testgenerators

4.1 Überblick

Es wurden drei Typen von Chipkarten für das System der deutschen Gesundheitskarte mittels asynchroner Produktautomaten (APA), siehe oben, modelliert: die elektronische Gesundheitskarte (eGK), der Heilberufsausweis (HPC) und die Sicherheitsmodulkarte (SMC). Diese Chipkarten sind im Detail in [Gema06a], [Gema06b], [Gema06c], [Gema06d] beschrieben. Die Chipkarten sind über ein Hintergrundsystem miteinander verbunden. Dieses besteht im Wesentlichen aus den Kartenlesern und der Software, welche die Transaktionen zwischen den Karten steuert (Abb. 1). Dabei können die HPC und die SMC unterschiedliche Rollenkennzeichen tragen, welche den Zugriff auf die eGK beeinflussen.

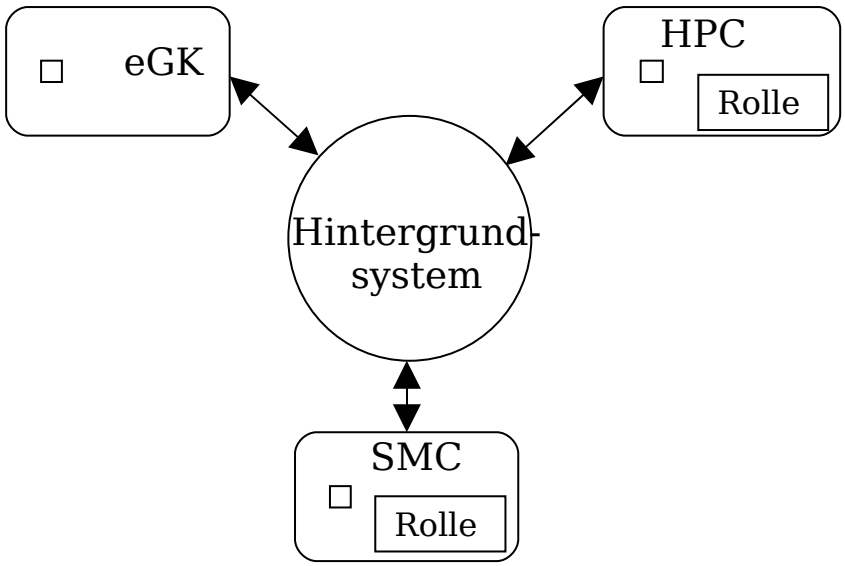


Abb. 1: Das Multikartensystem der deutschen Gesundheitskarte

Für jede Karte wurde ein separates Modell erstellt, welches das Antwortverhalten der Karte beschreibt. Das Hintergrundsystem haben wir als Testgenerator modelliert, welcher zufällige Kommandofolgen für die einzelnen Karten erzeugt. Diese aus abstrakten Kommandos bestehenden Folgen wurden in konkrete Kommandos übersetzt und auf dem Testtool [Gies07] mit realen Chipkarten ausgeführt. Die erzeugten Kommandofolgen testen die Zugriffsicherheit und Interoperabilität der verschiedenen Karten und deren Robustheit gegenüber falschen Kommandoreihenfolgen. Zudem werden damit alle im Rahmen des Modells möglichen Anwendungsfälle abgedeckt.

4.2 Abbildung der Sicherheitsaspekte im Modell

In Abb. 2 ist der schematische Aufbau des APA einer Chipkarte gezeigt. In der Zustandskomponente „Command“ wird vom Testgenerator ein

Kommando abgelegt, und dann durch Schalten der Transition „Command“ in die Zustandskomponente „In“ befördert. Diese Zustandskomponente ist mit Transitionen verbunden, die die Kommandos darstellen, welche die Chipkarte verarbeiten kann. Eine Kommando-Transition schaltet, wenn das entsprechende Kommando auf der Zustandskomponente „In“ liegt. Es wird weiter geprüft, ob das Kommando im aktuellen Zustand erlaubt ist, indem der Inhalt der Zustandskomponente „State“ ausgewertet wird. Je nachdem, ob das Kommando erlaubt ist oder nicht, wird in der Zustandskomponente „Response“ der entsprechende Rückgabewert eingetragen und evtl. der Inhalt von „State“ geändert.

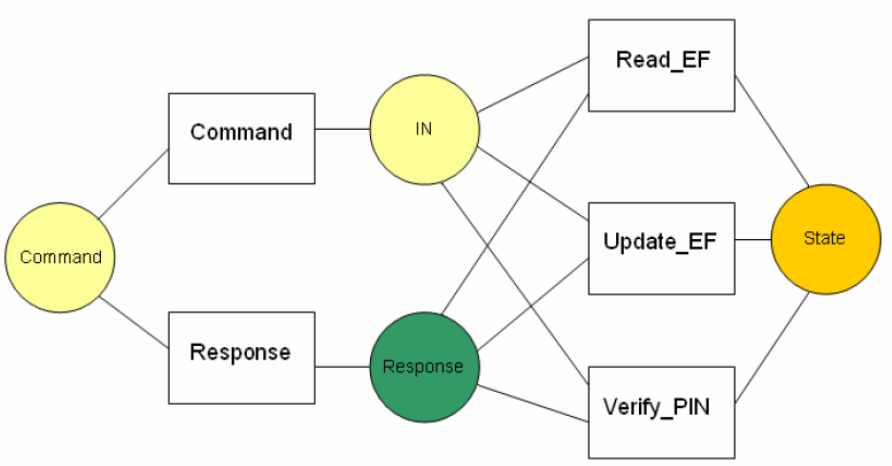


Abb. 2: Schematischer Aufbau des Produktautomaten zur Simulation einer Chipkarte

Auf diese Weise wird das zustandsabhängige Antwortverhalten der Karte beschrieben. Der Sicherheitsaspekt 1 in Abschnitt 2 wird abgedeckt, indem z.B. bei der eGK das Setzen und Löschen der folgenden Sicherheitszustände berücksichtigt wird:

Tab. 1: Sicherheitszustände in der eGK

Zustand	Beschreibung
„init“	Initialzustand nach dem Reset
„pin“	Zustand nach Eingabe der User-PIN
„pin_home“	Zustand nach Eingabe der PIN-home über Internet
„role_verified“	Zustand, nachdem mit VerifyCertificate eine Rolle übertragen wurde
„auth_started“	Zustand, nachdem das erste Teilkommando einer Authentisierung übertragen wurde
„role_authenticated“	Zustand, nachdem eine Rolle mittels vollständiger Authentisierung übertragen wurde

Die obigen Zustände können auch kombiniert auftreten, so kann gleichzeitig eine PIN und eine Rolle authentisiert sein.

Das tatsächliche Modell ist etwas komplizierter aufgebaut und enthält für die eGK z.B. 9 Kommandos und 4 Zustandskomponenten für die Verwaltung der Zustände.

Zur Abdeckung des Sicherheitsaspektes 2 in Kapitel 2 betrachten wir die Modellierung des Testgenerators, die in Abb. 3 dargestellt ist.

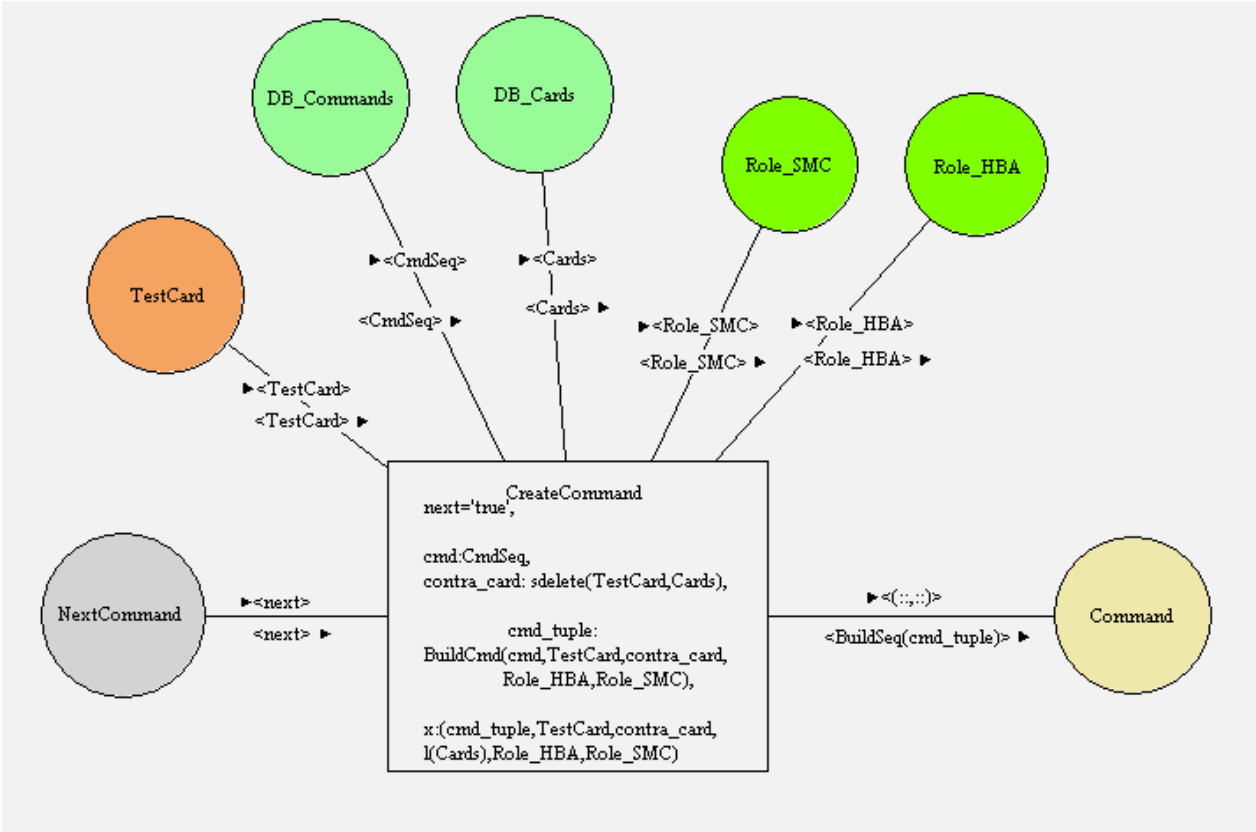


Abb. 3: Modell des Testgenerators

Der Testgenerator wird als eigenständiger APA modelliert. Er besteht nur aus einer Transition „CreateCommand“, die dafür sorgt, dass ein Kommando in die Zustandskomponente „Command“ gelegt wird, von wo aus es dann von den Kartenmodellen weiterverarbeitet wird (siehe Abb. 2). Das Kommando wird über die beiden Zustandskomponenten DB_Commands und DB_Cards zusammengesetzt. DB_Commands enthält alle möglichen Kommandos und DB_Cards die möglichen Kartentypen die als Gegenkarte verwendet werden sollen, also „eGK“, „HBA“ und „SMC“. Zustandskomponenten können also nicht nur zur Verwaltung von Zuständen, sondern auch zur Speicherung von Datenmengen dienen.

Der Testgenerator weiß nicht, in welchem Zustand sich die Karten befinden, er wählt zufällig ein Kommando und eine Karte aus. Damit wird der Sicherheitsaspekt 2 realisiert, an die Karten werden auch Kommandos geschickt, die abgelehnt werden müssen, und dies vollständig, d.h. in jedem Zustand werden alle in DB_Commands enthaltenen Kommandos geschickt. Das ist durch die Anlage des Modells als Produktautomat gewährleistet. Damit ist auch der Sicherheitsaspekt 4 erfüllt, denn wenn die Kommandos in zufälliger Reihenfolge geschickt werden, sind auch alle Gutfallvariationen dabei.

Die weiteren Zustandskomponenten „RoleSMC“, „RoleHBA“, „TestCard“ dienen zur Konfiguration der Tests. „RoleSMC“, „RoleHBA“ enthalten die Rollen der SMC und der HBA, die getestet werden sollen, und „TestCard“ bezeichnet die Karte, die im Testfokus steht. Damit können alle Rollen berücksichtig werden, indem für jede Rollenkombination ein Testlauf erzeugt wird, wodurch der Sicherheitsaspekt 3 erfüllt wird.

Das Modell ist so angelegt, dass eine Karte im Testfokus steht, und jeweils die beiden anderen Karten (die dann aus „DB_Cards“ gewürfelt werden) als Lieferanten für Zertifikate, Kryptogramme und Zufallszahlen benutzt werden. Damit wird der Sicherheitsaspekt 5 automatisch abgedeckt.

5 Testgenerierung aus dem Modell

5.1 Generierung abstrakter Testfolgen

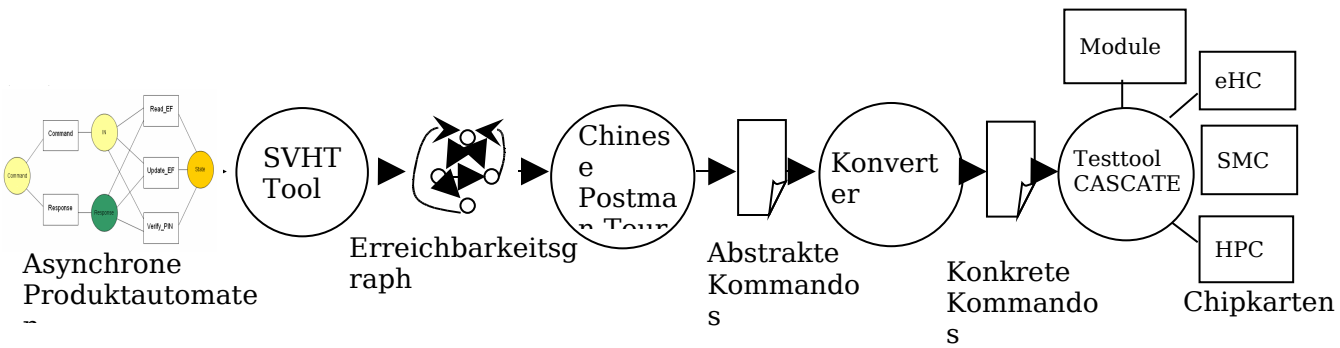
Für die modellierten Teile eGK, HBA, SMC, Testgenerator, die ein zusammenhängendes Netz bilden, wurde ein Erreichbarkeitsgraph berechnet, der alle im Modell abgebildeten Kartenzustände, Kommandokombinationen und Kartenkombinationen beinhaltet. Für diesen Erreichbarkeitsgraphen wurde dann eine Chinese Postman Tour (minimaler Pfad, der alle Kanten des Graphen mindestens einmal durchläuft) berechnet, woraus sich eine abstrakte Kommandofolge ergab.

5.2 Konkretisierung zu ausführbaren Testfolgen

Um die erzeugten abstrakten Kommandofolgen auf realen Chipkarten laufen zu lassen, müssen diese noch auf konkrete lauffähige Kommandos abgebildet werden. Abb. 4 zeigt die Kette der verschiedenen Verarbeitungsschritte. Diese können so automatisiert werden, dass per Knopfdruck das lauffähige Testskript erzeugt wird.

Die Konkretisierung besteht zum einen in einer Aufteilung in mehrere Teilkommandos, zum anderen in einer Anpassung der Syntax, so dass die Kommandos für das Testtool in Form eines Testskripts lesbar werden. So bedeutet z.B. das abstrakte Kommando „IntAuth eGK_HPC“, also die interne Authentisierung einer eGK gegenüber der HPC, dass zunächst bei der HPC mit dem Kommando GetChallenge eine Zufallszahl abgeholt wird, da diese als Eingangsdatum für das Kommando InternalAuthenticate benötigt wird, bevor dieses Kommando dann zur eGK gesendet wird.

Jedes abstrakte Kommando wird im Testtool als Modul angelegt, das dann die Kommandodetails abbildet. Das Testtool [Gies07] besitzt eine



Multikarten-Schnittstelle und kann vollautomatisch Multikarten-Testskripte abarbeiten.

Abb. 4: Verarbeitungsschritte für die Erzeugung von lauffähigen Kommandofolgen

5.3 Ergebnisse

In einem ersten Schritt wurden die folgenden Kommandos bzw. Kommandogruppen abgebildet:

ReadCert: lesen der Kartenzertifikate

VerifyCert: verifizieren der Kartenzertifikate

VerifyPIN: verifizieren der User-PIN

VerifyPIN_home: verifizieren der PIN-home

IntAuth: interne Authentisierung

ExtAuth: externe Authentisierung (mit MSE+GetChallenge)

ReadEF: Files lesen

UpdateEF: Files schreiben

Reset: Reset der Karte (mit SelectDF)

Manche Kommandos wurden dabei zu Gruppen zusammengefasst. Die Kommandos einer Gruppe werden als Block an die Karte geschickt. Dies ist dann sinnvoll, wenn die Reihenfolge keinen Einfluss auf die Zustände der Karte hat, wie z.B. Lese- und Schreibzugriffe auf die verschiedenen Files (ReadEF, UpdateEF).

Für die eGK als Testfokus ergaben sich (für ein Rollenpaar HBA/SMC) 2283 Zustände im Erreichbarkeitsgraphen und eine abstrakte Kommandosequenz von 3304 Kommandos. Für eine HBA oder SMC als Testfokus ergaben sich etwa 150 Zustände und eine abstrakte Kommandosequenz von 300 Kommandos. Die geringere Zustandsmenge bei der HBA und SMC kommt daher, dass diese keine rollenabhängigen Zugriffe wie in der eGK verwalten müssen. Die Durchlaufzeit für einen Testlauf auf dem Testtool mit einer Rollenpaarung Betrug für eGK und HBA zusammen etwa 40 Minuten. Die sich hier ergebenden Zustandsmengen und Laufzeiten sind noch relativ gering, so dass die geplanten Erweiterungen (siehe Kapitel 6) ohne weiteres realisierbar sind.

Bei der Analyse der entstandenen Kommandofolgen konnten auch wichtige Abläufe wie die Freischaltung der SMC durch die HBA identifiziert werden, die sich aus dem Modell automatisch ergaben.

Bei den Testdurchläufen traten einige Fehler auf. Zum einen waren die verwendeten Testkarten nicht fehlerfrei implementiert und mussten entsprechend angepasst werden, zum anderen war auch die Unsetzung im Modell nicht fehlerfrei. Nach Korrektur dieser Abweichungen liefen die Testläufe ohne Fehler.

Es hat sich damit gezeigt, dass die erwendete Methode sehr gut in der Lage ist, effiziente Kommandofolgen für Testzwecke zu liefern.

6 Ausblick

Modellbasiertes Testen erleichtert das Erzeugen effizienter Testfolgen mit einer hohen Testabdeckung und spart im Vergleich zur manuellen Testgenerierung Zeit bei Entwicklung und Wartung der Testfolgen. Nachdem ein Modell der zu testenden Anwendung entwickelt wurde, kann dessen Korrektheit mit Tool-Unterstützung validiert werden.

Um eine Erweiterung des Modells der Chipkarten für das Gesundheitswesen vorzunehmen, müssen im Wesentlichen die APA angepasst werden. Falls neue Kommandos hinzukommen, müssen auch die Module im Testtool angepasst werden. Es ist geplant, das Modell um die folgenden Funktionalitäten zu erweitern:

- Anzahl der Kommandos. Kommandos, die bisher nicht explizit modelliert worden sind, wie z.B. GetChallenge und ManageSecurityEnvironment, sollen hinzugefügt werden. Damit ist auch eine Erweiterung der Kartenzustände verbunden.
- Detaillierte Rückgabewerte: Im bisherigen Modell werden als Rückgabewerte von der Karte 9000, 6982 und 6xxx abgebildet. Der generische Rückgabewert 6xxx soll weiter präzisiert werden.
- Die von der HPC und SMC unterstützten logischen Kanäle sollen berücksichtigt werden.
- Neben der bisher unterstützten Authentisierungsvariante soll noch eine weitere mit Aushandlung von Session Keys hinzugefügt werden.

Danksagung

Wir danken unseren Kollegen Olaf Henniger und Peter Ochsenschläger für die Mitarbeit an dieser Veröffentlichung und für Ihre Beiträge zur Entwicklung der Methoden und Werkzeuge über die hier berichtet wird. Weiter danken wir Herrn Eberhard Spatz für die hilfreiche Unterstützung.

Literatur

[Gema06a] Gematik: eGK Teil 1: Kommandos, Algorithmen, Funktionen der Betriebssystem-Plattform, Version 1.1.0 (07.02.2006)

[Gema06b] Gematik: eGK Teil 2: Anwendungen und anwendungsspezifische Strukturen. Version 1.2.1. (07.09.2006)

[Gema06c] Gematik: German Health Professional Card and Security Module Card Part1: Commands, Algorithms and Functions of the COS Platform, Version 2.1.0 (2006)

[Gema06d] Gematik: German Health Professional Card and Security Module Card Part 2 and 3: HPC Applications and Functions, Version 2.1.0 (2006)

[Giel93] H. Giehl: Verifikation von Smartcard-Anwendungen mittels Produktnetzen. In: Gesellschaft für Mathematik und Datenverarbeitung mbH, ISBN 3-88457-225-3, November 1993

[Nebe94] M. Nebel: Ein Produktnetz zur Verifikation von Smartcard-Anwendungen in der STARCOS-Umgebung. In: Gesellschaft für Mathematik und Datenverarbeitung mbH, ISBN 3-88457-234-2, Juli 1994

[ORR00b] P. Ochsenschlger, J. Repp, and R. Rieke: The SH-Verification Tool . In Proc. 13th International Florida Artificial IntelligenceResearch Society Conference (FLAIRS-2000) , pages 18-22, Orlando, FL, USA, May 2000. AAAI Press.

[ORR00c] P. Ochsenschlger, J. Repp, and R. Rieke: Abstraction and composition - a verification method for co-operating systems . Journal of Experimental and Theoretical ArtificialIntelligence , 12:447-459, June 2000.

[StEW06] B. Struif., L. Eckstein, U. Waldmann: Use Cases zum elektronischen Arztausweis, Version 1.0, (31.07.2006)

[SHVT] Simple Homomorphism Verification Tool – Manual. Fraunhofer-Institut für Sichere Informationstechnologie

[Gies07] Giesecke & Devrient: Das Testtool CASCATE® ist ein Test-Studio für Chipkartentests. Es umfasst Testfallmanagement, Testausführung, Testdebugging und Testdokumentation. Entwickelt von Giesecke & Devrient (2007)