Hochschule Luzern
Technik & Architektur

Informatikprojekt | TA.PAWI.FS2012

Aspects of Privacy-Preserving Toll Pricing

Testplan

Horw, 8. Februar 2012

Projekt	Aspects of Privacy-Preserving Toll Pricing
Dokument	Testplan
Schule	Hochschule Luzern, Technik & Architektur
Modul	TA.PAWI.FS2012
Projektteam	**Galliker Thomas** Studiengang Informatik (BB) Panorama 6123 Geiss Tel. +41 79 504 80 70 thomas.galliker@stud.hslu.ch
	Moser Christoph Studiengang Informatik (VZ) Zugerbergstrasse 41 6314 Unterägeri Tel. +41 79 785 19 07 christoph.moser@stud.hslu.ch
Dozent	**Dr. Marc Pouly**
Experte	**Dr. Josef F. Bürgler**
Letzte Änderung	8. Februar 2012, 13:44:00 Uhr

Änderungsprotokoll

Version	Datum	Autor	Beschreibung
0.1	18.01.2012	gat	Initialversion von Vorlage erstellt
0.2	30.01.2012	gat	Test Cases anhand der Use Cases erstellt
0.3	05.02.2012	gat	Testbedingungen ergänzt
0.4	07.02.2012	gat	Korrekturen und Ergänzungen

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Ziel & Zweck dieses Dokuments

In diesem Dokument wird das Testing im PAWI-Projekt „Aspects of Privacy-Preserving Toll Pricing“ geplant. Anwendungsfälle werden in Testfällen abgebildet. Jeder Testfall enthält genaue Spezifikationen bezüglich Testbedingungen, Testvorgehen und erwarteten Resultaten. Alle Informationen, welche mit der Planung von Tests im Zusammenhang stehen, werden in diesem Dokument festgehalten.

1.2 Begriffe

Begriff	**Erklärung**
Unit Test	Dient zur Verifikation der Korrektheit von Modulen einer SW
Systemtest	Testphase wo das gesamte System auf die Anforderungen (funktionale und nicht funktionale Anforderungen) getestet wird
Integrationstest	Testet die voneinander abhängigen Komponenten

Tabelle 1: Begriffserklärungen

Abkürzung	**Erklärung**
CL-RSA	Signaturalgorithmus von Camenisch und Lysyanskaya basierend auf RSA
GPS	Global Positioning System
HSLU	Hochschule Luzern
moc	Namenskürzel für Christoph Moser
OBU (Client)	On-board Unit; mobiles Computersystem zur Verrechnung der Fahrleistung und Abrechnung mit dem Toll Service Provider
PAWI	Modulbezeichnung Industrieprojekt HSLU
PrETP	Privacy-preserving Electronic Toll Pricing; ein Strassenabrechnungssystem, welches die Privatsphäre des Anwenders schützt
RPC	Remote Procedure Call; Aufruf von Prozeduren auf entfernten Systemen
TSP	Toll Service Provider; staatlicher Strassenverwalter, erhebt und verrechnet Gebühren für die Benutzung von Strassen
TSP Client	Client Applikation zur Verwaltung von TSP Service
TSP Service	Service Applikation als zentrale Verwaltungsstelle des PrETP Systems
XMPP	Extensible Messaging and Presence Protocol, ehem. „Jabber”, Kommunikationsprotokoll für Instant Messanging (Facebook Chat, Google Talk)

Tabelle 2: Abkürzungserklärungen

2 Testphilosophie

Unsere Testphilosophie besteht aus drei wesentlichen Teilen:

- Testen jeder Komponente (Unit-Tests)
- Testen des Zusammenspiels der Komponenten (Integrations- und Systemtests)

2.1 Unit-Test

In den Unit-Tests werden einzelne, in sich abgeschlossene Klassen getestet. Die Unit-Tests sollen – wo immer möglich – parallel zur Entwicklung erstellt und ausgeführt werden. Die Unit-Tests sollen nach dem Test-First Konzept mit JUnit durchgeführt werden.

Der Test-First Ansatz geht davon aus, dass der Entwurf von Tests und deren Ausführung auch das Design des Programms vorantreibt. Bevor ein Stück eines Programms geschrieben wird, wird zuerst ein Test entworfen. Anschliessend wird nur so viel Programmcode geschrieben, wie der Test verlangt. Dementsprechend findet die Programmentwicklung in einer raschen Folge von Test- und Implementierungsschritten statt.

Funktionalität ist bei Unit-Test gleichbedeutend mit Ein-/Ausgabeverhalten der zu prüfenden Einheit. Dabei wird jede Einheit isoliert getestet um sicher zu stellen, dass andere Einheiten keinen Einfluss auf das Verhalten der zu prüfenden Einheit haben und somit ein Fehler klar zu zuordnen ist.

2.2 Integrationstest

Der Integrationstest hingegen prüft das Zusammenspiel von Komponenten und deren Wechselwirkung. Sowohl Unit-Tests als auch Integrationstests werden an den Schnittstellen der zu prüfenden Einheit durchgeführt.

Wichtig ist, dass die Komponenten auf Basis der Systemspezifikation untereinander korrekt agieren und dass eine gute Leistung gewährleistet ist.

2.3 Systemtest

Unit- und Integrationstest können mit gängigen Tools automatisiert werden. Komplette Systemtests werden in diesem Projekt manuell durch Testpersonen durchgeführt. Getestet werden hauptsächlich die in der Kundenanforderung erfassten Anwendungsfälle (Use Cases) sowie die funktionalen und nicht-funktionalen Anforderungen.

3 Testspezifikation

Grundsätzlich sollen alle entwickelten Komponenten und deren Wechselwirkung kontinuierlich getestet werden. Um das Konzept von „Continous Integration“ umzusetzen, werden die Unit- und Integrationstests in diesem Projekt auf einem TeamCity Build Server automatisch ausgeführt.

Unit-Tests werden parallel zum Entwicklungsprozess einer Klasse erstellt und ausgeführt werden. Der TeamCity Server sorgt dafür, dass auch beim Einchecken von neuem Source Code sowie täglich um Mitternacht ein Build von allen Testobjekten erstellt wird.

3.1 Testumgebung

TSP Service und TSP GUI Client wurden während den Tests auf Windows 7 Laptops ausgeführt. Die OBU Client Anwendung wurde sowohl auf dem Android Emulator wie auch auf einem Google Nexus One Mobiltelefon getestet.

3.2 Testobjekte

Testobjekt	**Vorgehensweise, Schwerpunkte beim Testen**
tspclient	Das GUI in tspclient wird vorwiegend manuell getestet. Sämtliche Funktionen, welche im GUI implementiert sind, werden vorgängig in Unit-Tests getestet.
tspservice	Bei diesem Objekt kommen Unit- und Integrationstests in Frage. Unit-Tests können im Zusammenspiel mit Datenbank- und Kommunikationsverbindung geprüft werden.
obuclient	OBU Client wird vorwiegend manuell getestet. (Android Test Framework lieferte keine aussagekräftigen Informationen zurück).
common	Der Fokus liegt hier vor allem bei Unit-Tests, welche die einzelnen Bibliotheken prüft. Die Schwerpunkte beim Testen liegen bei der Qualitätssteigerung der Crypto Library wie auch jener der XMPP-RPC Bibliothek.

3.3 Wesentliche Testaspekte

Auf folgende wesentliche Testaspekte muss grossen Wert gelegt werden:

- **Anforderungserfüllung**: Erfüllen der geforderten Anforderungen.
- **Robustheit**: Die Tests sollten beliebig oft wiederholbar sein.
- **Datenkonsistenz**: Vor und nach dem Ausführen von Tests muss die Testdatenbank in einem Zustand hinterlassen werden, welcher wiederholtes (automatisiertes) Testen zulässt.
- **Code Coverage**: Keine Vorgaben; Selbstkontrolle.
- **Interoperabilität**: Verschiedene Laufzeitumgebungen müssen getestet werden.

3.4 Testfälle

Die nachfolgenden Kapitel liefern Informationen zu den Testfällen, welche im Laufe des Projekts entstanden sind. Während Unit- und Integrationstests mittels automatisierter Softwaretests (JUnit)

3.4.1 Unit- und Integrationstests

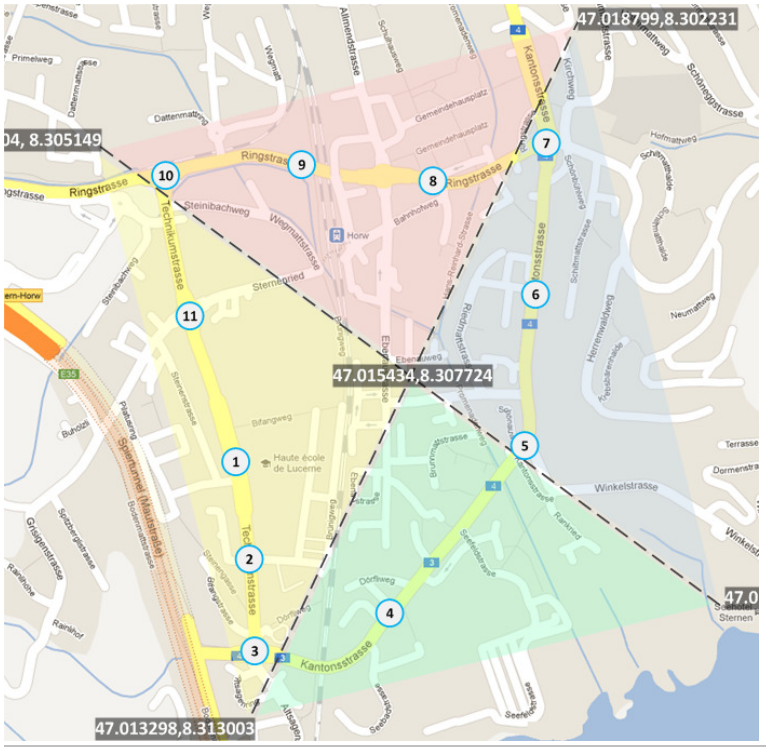
Folgende Unit- und Integrationstests wurden während der Entwicklung erstellt und ausgeführt:

Test Name	Beschreibung
Komponente common	
DateUtilsExtTest	Tests für die Formatumwandlung von Klasse Date zu String. Wird im Zusammenhang mit der SQLite Datenbank gebraucht, da dort keine SQL Datentypen für Date vorhanden sind.
GPSTest	Testet einfache mathematische Operationen, wie z.B. die Distanz zwischen zwei GPS Koordinaten.
SerializerTest	Prüft die Funktionsweise der Serializer Klasse, welche für das Serialisieren von ArrayList Datentypen eingesetzt wird.
GroupParametersTest	Prüft ob ein GroupParameter Objekt in ein String und wieder zurück gewandelt werden kann. Diese wird benutzt um Group Parameters in der Datenbank zu speichern.
PrivateKeyTest	Prüft ob ein PrivateKey Objekt in ein String und wieder zurück gewandelt werden kann. Diese wird benutzt um Private Keys in der Datenbank zu speichern.
PublicKeyTest	Prüft ob ein PublicKey Objekt in ein String und wieder zurück gewandelt werden kann. Diese wird benutzt um Public Keys in der Datenbank zu speichern.
CommitmentTest	Erstellt und vergleicht Commitments.
HashSchemeTest	Erstellt und vergleicht Hashes.
SignatureTest	Erstellt und verifiziert CL-RSA Signaturen.
ZeroKnowledgeProofTest	Erstellt und verifiziert Proofs.
Komponente tspservice	
PointToMapTest	Testet anhand der in der Datenbank konfigurierten Maps, ob und zu welcher Map die Testpunkte gehören.
XMPP_P2PTest	Testet eine XMPP Chat-Verbindung zwischen zwei Benutzern.
XMPP_IOBU_RoomTest	Testet eine XMPP-RPC Chat-Verbindung zwischen einer OBU und dem TSP.
XMPP_ITSP_RoomTest	Testet eine XMPP-RPC Chat-Verbindung zwischen dem TSP und einer OBU.
SecurityManagerTest	Prüft ob der Optimistic Payment Prozess auf der TSP korrekt funktioniert. Das heisst, ob korrekte Payments als solche gewertet werden und ob Manipulationen erkannt werden.

Tabelle 3: Unit- und Integrationstests

3.4.2 Systemtests

Die Systemtests dienen als praktischer Nachweis für die Erfüllung der Kundenanforderung. Die wurden sowohl mit dem Android Emulator wie auch mit einem physischen Android Device durchgeführt. Für die Systemtests wurde eine Testumgebung definiert, welche Horw in vier grosse Segmente unterteilt. Abbildung 1 zeigt die Karte mit den vier Zonen, abgegrenzt durch die aufgeführten GPS Eckpunkte. Die nachfolgende Tabelle enthält die Geo Fix GPS Koordinaten, welche dem Android Emulator während den Tests eingespeist wurden.



Ref.#	Geo Fix (Longitude/Latitude)
1	geo fix 8.304870 47.013883
2	geo fix 8.305085 47.012640
3	geo fix 8.305407 47.011206
4	geo fix 8.308218 47.011703
5	geo fix 8.311157 47.014381
6	geo fix 8.311307 47.016561
7	geo fix 8.311565 47.018814
8	geo fix 8.309119 47.018243
9	geo fix 8.306179 47.018462
10	geo fix 8.303089 47.018287
11	geo fix 8.303733 47.016341

Abbildung 1: GPS Testmessungen für Systemtests

ID	1	
Beschreibung	Strecke aufzeichnen	
Vorgehen	1. OBU Client starten 2. Im Abstand von 5 Sekunden die GPS Koordinaten der „Horwer Runde“ einspielen.	
Voraussetzung	• TSP Service konfiguriert und gestartet	
Erwartetes Resultat	• Auf dem OBU Client wird angezeigt, dass eine Strecke von ungefähr zwei Kilometern erfasst wurde.	
Protokoll	03.02.2012	>300km in der „Horwer Runde“ aufgezeichnet und ausgewertet. PaymentRecords einwandfrei erstellt. Abweichungen wegen Preisberechnungsfunktion von >20%! Das wird wohl an den grossen Abständen zwischen den GPS Punkten liegen.

Tabelle 4: Systemtest, Test Case ID1

ID	2	
Beschreibung	Abrechnungsperiode verarbeiten	
Vorgehen	1. TSPClient starten 2. Im Tab „Optimistic Payment“ das Fahrzeug auswählen, welches abgerechnet werden soll und das Enddatum bestimmen 3. Abrechnung starten	
Voraussetzung	• TSP Service konfiguriert und gestartet • Horwer Runde wurde aufgezeichnet	
Erwartetes Resultat	• Der TSPClient zeigt auf, dass das Optimistic Payment erfolgreich war.	
Protokoll	03.02.2012	Test erfolgreich durchgeführt. Nacharbeiten nötig.

Tabelle 5: Systemtest, Test Case ID2

ID	3	
Beschreibung	Beweis erfassen	
Vorgehen	1. TSPClient starten 2. Im Tab „Optimistic Payment“ die benötigten Attribute (Longitude, Latitude, Datum, Fahrzeugkennzeichen) für einen Beweis ausfüllen.	
Voraussetzung	• TSP Service konfiguriert und gestartet • Horwer Runde wurde aufgezeichnet	
Erwartetes Resultat	• Ein neuer Beweis wurde für das gewählte Fahrzeug in der Datenbank gespeichert.	
Protokoll	03.02.2012	Beweis wurde erfolgreich in der Datenbank gespeichert.

Tabelle 6: Systemtest, Test Case ID3

ID	4	
Beschreibung	Beweis prüfen	
Vorgehen	1. TSPClient starten 2. Beweise für das abzurechnende Fahrzeug erfassen (siehe TestCase 3) 3. Abrechnugsperiode für das abzurechnende Fahrzeug starten (siehe TestCase 2)	
Voraussetzung	• TSP Service konfiguriert und gestartet • Horwer Runde wurde aufgezeichnet	
Erwartetes Resultat	• Im TSPClient wird eine Auswertung der erfassten Challenges angezeigt. Die Challenges welche mit der aufgezeichneten Route übereinstimmen werden positiv gewertet.	
Protokoll	03.02.2012	Test erfolgreich durchgeführt. Nacharbeiten nötig.

Tabelle 7: Systemtest, Test Case ID4