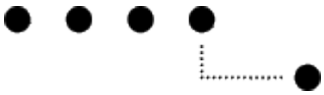
On-Line Meinungsumfragen

Diplomschlussbericht



Berner Fachhochschule
Technik und Informatik

Projekt	On-Line Meinungsumfragen	
Dokument	Diplomschlussbericht.odt	
Ausgabe	Freitag, 9. November 2007	
Version	1.0 Final	
Projektteam	Adrian Aeby	aebya@bfh.ch
	Martin Wiget	wigem1@bfh.ch
Betreuer	Bernhard Anrig	Bernhard.Anrig@bfh.ch
	Rolf Haenni	haenni@iam.unibe.ch
Experte	Han van der Kleij	han.vanderkleij@rtc.ch

0 Allgemeines

0.1 Änderungskontrolle

Version	Datum	Name	Bemerkungen
0.1	30.08.2007	Adrian Aeby	Erster Entwurf
0.2	15.10.2007	Adrian Aeby	Erweiterungen
1.0	09.11.2007	Adrian Aeby	Kapitel zusammenführen

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Zweck des Dokuments

Die Diplomarbeit On-Line Meinungsumfrage (OLM) findet in diesem Dokument seinen Abschluss. Dieses Dokument dient dazu, die geleisteten Arbeiten während der Diplomarbeit (inkl. vorangehender Semesterarbeit) zu beschreiben und das Vorgehen zu schildern. Wir beschreiben unsere Vorgehensweise, gewonnenen Erkenntnisse und die Erfahrungen, die wir während des Projektes OLM gemacht haben. Der Stand der Arbeiten wird ebenso beschrieben wie mögliche Erweiterungen der OLM.

2 Management-Übersicht

Ziel dieser Diplomarbeit ist, eine Meinungsumfrage über das Internet abwickeln zu können und damit die möglichen Sicherheitsprobleme zu lösen.

Was unser System von anderen unterscheidet, ist, dass wir den Hauptfokus auf die teilnehmenden Personen (Poller) gerichtet haben. Konkret heisst das: Ein Poller kann sicher sein, dass er anonym an der Umfrage teilnehmen kann, aber dennoch nachprüfen kann, dass seine Antworten in der Gesamtstatistik mitgezählt werden. Dieses Problem ist nicht ganz trivial zu lösen und ist das Herzstück unserer Arbeit. Natürlich wird auch gewährleistet, dass eine unautorisierte Person nicht teilnehmen kann und ein Poller nur einmal seine Antworten abschickt.

Der Auftraggeber (Customer) kann einen Fragebogen erstellen und den Teilnehmerkreis bestimmen. Ebenfalls lädt der Customer alle Poller ein. Damit ist für die Poller klar ersichtlich, wer die Umfrage initiiert hat. Für die Durchführung der Umfrage sind nun Drittparteien (Provider, Certifying Authority und Delayer) zuständig. Der Provider publiziert letztendlich das Umfrageergebnis.

Durch die Publizierungen bestimmter Informationen der Agents kann jeder Poller selbst nachvollziehen, dass seine Antworten in die Gesamtstatistik eingeflossen sind.

3 Ausgangslage

3.1 Problemstellung

On-Line Meinungsumfragen sind ein populäres Mittel, auf rasche und einfache Weise die Meinung einer Zielgruppe zu erfassen. Wie kann eine On-Line Meinungsumfrage sicher und vertrauenswürdig durchgeführt werden?

Viele On-Line Meinungsumfragen, die per E-Mail initiiert werden, vermitteln nicht den Eindruck einer seriösen Umfrage oder der Urheber der Umfrage ist nur schwer ersichtlich. Da oft persönliche Angaben der Teilnehmer verlangt werden, muss die Meinungsumfrage klare Angaben zum Zweck und zur Verwendung haben. So muss auch klar sein, wer Zugang zu den Daten hat. Zudem muss gewährleistet werden, dass jeder Teilnehmer anonym und nur einmal abstimmen kann, dass die Daten unversehrt zum Provider gelangen und ausgewertet werden.

Als besondere Herausforderung gilt, dass ein registrierter Umfrageteilnehmer sicher sein kann, dass niemals ein Umfrageergebnis einem Teilnehmer zugeordnet werden kann.

3.2 Aufteilung der Arbeit

Das ganze Projekt teilt sich in zwei separat bewertete Teilprojekte auf. Der Einfachheit halber nennen wir den ersten Teil Semesterarbeit (dieser Teil des Projekts zählt zum letzten Semester an der BFH-TI) und den zweiten Teil Diplomarbeit (dies ist die Diplomarbeit im eigentlichen Sinne, die nach Abschluss des Unterrichts durchgeführt wird). Die Semesterarbeit dient einerseits zur Entscheidung, ob sich das Projekt als Diplomarbeit eignet und andererseits als Vorarbeit zur Diplomarbeit.

3.2.1 *Semesterarbeit*

Die Semesterarbeit dient als Vorarbeit zur Diplomarbeit. Während der Semesterarbeit haben wir das Sicherheitskonzept für die On-Line Meinungsumfrage erarbeitet.

3.2.2 *Diplomarbeit*

In der Diplomarbeit wurde das in der Semesterarbeit erstellte Sicherheitskonzept implementiert, um mit einem Prototyp eine einfach On-Line Meinungsumfrage durchführen zu können.

4 Resultate Semesterarbeit

Ziel der Semesterarbeit war das Erarbeiten eines Sicherheitskonzepts und die Implementierung eines rudimentären Prototypen.

4.1 Erreichte / nicht erreichte Projektziele

Das Sicherheitskonzept stellte sich dann als so umfangreich heraus, dass wir uns mit den Fachdozenten dazu entschieden, im Umfang der Semesterarbeit nur das Sicherheitskonzept zu erstellen.

4.1.1 *Sicherheitskonzept*

Die ersten Arbeiten bestanden darin uns mit den gängigen kryptographischen Technologien auseinanderzusetzen. Wir waren uns bewusst, dass der Sicherheitsaspekt in diesem Projekt eine sehr wichtige Rolle spielen wird. Trotzdem unterschätzten wir den Umfang der Sicherheitsfragen. Nach etwa einem Drittel der Semesterarbeit entschieden wir uns das Sicherheitskonzept nicht neu zu erstellen, sondern das Protokoll *Voting Over The Internet* [A.1] als Grundlage für unser Sicherheitskonzept zu verwenden.

Viele unserer Sicherheitsfragen waren in diesem Protokoll schon weitgehend beantwortet. Unsere Aufgabe bestand nun darin, dieses Protokoll zu verstehen, für unsere Verwendung zu adaptieren und zu verbessern. Das Protokoll gab uns eine gute Struktur des Sicherheitskonzepts, und wir konnten die einzelnen Schritte unabhängig vom Ganzen erarbeiten.

Das Sicherheitskonzept konnte in der letzten Woche der Semesterarbeit zu unserer Zufriedenheit abgeschlossen werden.

4.1.2 *Prototyp*

Den Prototypen, wie wir ihn zu Ende der Semesterarbeit haben wollten, haben wir nicht umgesetzt. Trotzdem konnten wir einige wichtige Erfahrungen beim Implementieren kryptographischer Elemente in Java machen, die wir in der Diplomarbeit auch verwenden. Die Erstellung eines webbasierten Umfrageeditors mit Ajax-Dojo haben wir eingestellt, da wir uns entschieden haben, den Editor als Java-Applikation zu erstellen.

4.2 Ziele und Lösungen

4.2.1 *Systemadaption*

Als Grundlage für unser Sicherheitskonzept verwendeten wir das Protokoll *Voting Over The Internet* [A.1]. Hier sind nun die wichtigsten Adaptionen, die wir zur Vereinfachung oder Verbesserung für unser Protokoll

vorgenommen haben, aufgelistet.

4.2.2 *Voter → Poller*

Das Original-Protokoll handelt über den Ablauf einer Abstimmung (Voting) und Wähler (Poller). Unser Protokoll beschreibt eine Umfrage (Poll) und bietet dies Pollern an. Deswegen heisst bei uns die Votermark Pollermark und die Teilnehmer sind Poller und nicht Voter.

4.2.2.1 *Polling Initiation*

Zuerst wollten wir eine Certificate Authority implementieren, die für jeden Poller ein Zertifikat heraus gibt. Da dies jedoch nichts mit dem Polling Protokoll zu tun hat, entschieden wir uns, diese aufwändige zusätzliche Arbeit fallen zu lassen. Als Voraussetzung für einen Poller gilt, dass er ein gültiges Zertifikat besitzt, in welchem seine E-Mail-Adresse als Distinguished Name verwendet wird. Zudem übernimmt der Customer die Rolle des Ballot Distributors. Weiter gehen wir davon aus, dass der Customer im Besitz aller Zertifikate der Poller ist. So haben wir uns entschieden, dass die Initiierung, wie sie im Original-Protokoll beschrieben ist, aus dem Protokoll entfällt und stattdessen der Customer den Pollern mit der Einladung die benötigten Informationen und Files zukommen lässt.

4.2.2.2 *Vote Casting*

Für uns war aus dem Protokoll *Voting Over The Internet* [A.1] nicht ersichtlich, wie ein Voting eines Attackers – also eines unautorisierten Voters – vom Vote Compiler, der die Ergebnisse entgegen nimmt und auswertet, erkannt wird. Deswegen machten wir folgende Anpassungen am Voting Protokoll:

- Der Poller versalzt den Hash der Pollermark anstatt der Pollermark und lässt diesen von der CA blind signieren.
- Beim Senden der Umfrageantworten gibt der Poller die Pollermark m mit zugehöriger Signatur HmS_{CA} mit.
  Mit m und HmS_{CA} kann der VC den Poller authentifizieren:
  - Durch Entschlüsseln von HmS_{CA} erhält VC h(m)
  - Durch Hashen vom m erhält VC h(m)

  Sind die beiden Werte gleich, ist der Poller authentifiziert.

Würde der Poller nur die Pollermark von der CA signieren lassen, könnte ein Attacker ein zufälliges Bitmuster als HmS_{CA} ausgeben und mit CA_e die dazugehörige gefälschte Pollermark m erstellen.

4.2.2.2.1 Beispiel alt

In diesem Beispiel wird gezeigt, wie nach dem Vorgehen im Original-Protokoll leicht eine Pollermark mit

entsprechender Signatur selbst erstellt werden kann und somit das Teilnehmen als unautorisierter Poller trotzdem unbemerkt bleibt.

Normales Signieren durch die CA

Der Poller generiert sich selbst eine Pollermark m.

$m = 52$

Diese wird nun mit der Zufallszahl r versalzen.

$m_v = m * r^{CAe} \bmod n = 52 * 17^{23} \bmod 143 = 78$

Danach wird die versalzene Pollermark von der CA blind signiert.

$m_{vs} = m_v^{CAd} \bmod n = 78^{47} \bmod 143 = 78$

Anschliessend erhält der Poller mit dem multiplikativ inversen Element von r in Zn die wahre Signatur der Pollermark.

$m_s = m_{vs} * r^{-1} \bmod n = 78 * 101 \bmod 143 = 13$

Nun besitzt der Poller das nötige Wertepaar aus Pollermark und Signatur.

$m = 52$
$m_s = 13$

Der Poller kann die Richtigkeit der Signatur mit dem öffentlichen Schlüssel der CA kontrollieren.

$m = m_s^{CAe} \bmod n = 13^{23} \bmod 143 = 52$

Eigenes Erstellen eines {Pollermark, Signatur} Wertepaares

Nun könnte jemand auf die Idee kommen, eine Signatur selbst zu definieren und mit dem öffentlichen Schlüssel der CA die dazu entsprechende Pollermark zu erstellen.

$m = m_s^{CAe} \bmod n = 53^{23} \bmod 143 = 14$

Das gefälschte Wertepaar {$m = 14$, $m_s = 53$} lässt sich nicht von einem normalen – durch die CA erstellten – Wertepaar unterscheiden.

Deswegen haben wir uns dazu entschieden, den Hash der Pollermark zu versalzen und diesen blind signieren zu lassen. Für die CA ändert sich nichts am Prozedere. Nun ist es aber nicht mehr möglich, selbst ein Wertepaar zu erstellen. Wohl kann man jetzt immer noch eine Signatur annehmen und den Hash berechnen, aber aus dem Hash kann man die Pollermark nicht mehr bestimmen.

4.2.2.2.2 Beispiel neu

Ein detailliertes Beispiel nach dem neuen verbesserten Vorgehen ist im Kapitel 7. Anonymisierung beschrieben.

4.2.3 *Sicherheitskonzept*

Die folgenden fünf Ziele, die im Dokument Pflichtenheft [B.1] Kapitel 4 beschrieben sind, wurden weitgehend während der Semesterarbeit gelöst und werden deshalb hier beschrieben.

Die Hauptziele des Sicherheitskonzepts sind die Gewährleistung von:

1. **Anonymität des Teilnehmers**
   Zu keinem Zeitpunkt darf eine Zusammenführung von Teilnehmer und abgegebener Stimme hergestellt werden können.

2. **Authentifizierung des Teilnehmers**
   Nur berechtigte Teilnehmer dürfen an der Meinungsumfrage teilnehmen, und jeder darf seine Antworten nur einmal abgeben.

3. **Integrität der Daten**
   Es darf an keiner Stelle – weder bei der Übertragung noch bei der Speicherung – möglich sein, Umfragedaten unbemerkt zu verändern, zu löschen oder hinzuzufügen.

4. **Integrität der Umfrageergebnisse**
   Die Berechnung von Zwischen- oder Teilergebnissen muss ausgeschlossen werden. Andernfalls könnte ein Auftraggeber nach einer gewissen Zeit ein Zwischen- oder Teilergebnis dazu missbrauchen, um die nachfolgenden Meinungsumfragen zu beeinflussen.

5. **Korrektheit der Ergebnisse**
   Das Ergebnis muss korrekt ausgewertet werden, insbesondere müssen alle abgegebenen Umfragen auch gewertet werden.

Diese Ziele werden wie folgt sichergestellt:

4.2.3.1 *Anonymität des Teilnehmers*

Die Anonymität eines Teilnehmers zu gewährleisten ist nicht so ein grosses Problem, da das ganze System ohne Benutzeridentifikation und Login ablaufen kann. Zusätzlich muss aber auch die Authentifizierung des Teilnehmers sichergestellt werden – siehe 4.2.3.2.Authentifizierung des Teilnehmers – nun ist es natürlich nicht mehr möglich, den gesamten Ablauf ohne eine Identifikation des Teilnehmers durchzuführen. Dieses Dilemma zu lösen ist das Hauptziel des Sicherheitskonzepts.

Dieses Problem wird so gelöst, dass jeder Teilnehmer ein eindeutiges Kennzeichen (nachfolgend Pollermark genannt) besitzt und niemand anders als der Teilnehmer selbst eine Verbindung zwischen seiner Pollermark und seiner Identität herstellen kann.

Damit nur der Teilnehmer selbst die Verbindung zwischen Poller-ID und Pollermark kennt, erstellt er diese selber. Es muss aber auch sichergestellt werden, dass jeder Poller eine andere Pollermark besitzt. Deshalb muss der Poller die erhaltene Seriennummer als Grundlage zur Generierung der Pollermark verwenden. Weitere Informationen für die Erstellung der Pollermark stehen weiter unten im Kapitel 7.1.2.Pollermark Erstellen.

Mit seiner Pollermark kann sich nun jeder Teilnehmer quasi einloggen und die gewünschte Bearbeitung vornehmen.

4.2.3.2 *Authentifizierung des Teilnehmers*

Nun könnte sich jeder Teilnehmer beliebig viele Pollermarks erstellen und auch nicht berechtigte Personen könnten sich Pollermarks aneignen. Um dieses Problem zu lösen, muss sich der Teilnehmer registrieren und die Pollermark signieren lassen. Diese Registrierung wird bei einer Drittpartei vorgenommen, die wir als Certifying Authority (CA) bezeichnen.

Dazu benötigt der Teilnehmer seine Pollermark, sein Zertifikat und sein vom Customer signiertes Zertifikat. Alles schickt der Teilnehmer an die CA, welche anhand des Zertifikats erkennt, ob dieser Teilnehmer zum ersten Mal eine Pollermark signieren lassen will und anhand des vom Customer signierten Zertifikats bestimmen kann, dass der Teilnehmer für diese Wahl zugelassen ist. Sind beide Prüfungen positiv, signiert die CA die Pollermark und schickt sie dem Teilnehmer zurück.

So entsteht die Situation, dass die CA in den Besitz der Pollermark und der Identität eines Teilnehmers kommt. Um dies zu verhindern, wird die Pollermark blind signiert. Dies geschieht so, dass der Teilnehmer nicht die Pollermark selbst an die CA schickt, sondern diese zuerst mit einer Zufallszahl r zu m' versalzt und diese von der CA signieren lässt. Das signierte m' kann er nun mit dem multiplikativ inversen Element zu r wieder entsalzen und erhält so sein signiertes m, ohne dass die CA weiss, wie die Pollermark tatsächlich aussieht. Genauere Informationen und ein Beispiel dazu folgen im Kapitel 7.1.Erstellen und Signieren einer Pollermark.

4.2.3.3 *Integrität der Daten*

Alle an der Umfrage beteiligten Parteien besitzen ein digitales Zertifikat. Die sicherheitssensitiven Daten werden mit dem entsprechenden Zertifikat verschlüsselt und so an den Empfänger geschickt. Zum Verifizieren der Daten können diese digital signiert werden. Falls nun die Daten verändert werden, schlägt die Verifizierung fehl und wird bemerkt.

4.2.3.4 *Integrität der Umfrageergebnisse*

Jeder Teilnehmer schickt seine Umfrage an den Vote Compiler (VC), dieser sammelt alle Umfrageantworten bis zum Endtermin der Umfrage und berechnet dann die Umfrageergebnisse. Vor Ablauf der Umfrage ist es niemandem möglich die einzelnen Umfrageantworten anzusehen.

4.2.3.5 *Korrektheit der Ergebnisse*

Ist der Endtermin durchlaufen, werden die einzelnen Umfrageantworten und das Umfrageergebnis öffentlich publiziert. Jeder Teilnehmer kann nun das Umfrageergebnis betrachten und besonders wichtig; anhand seiner Pollermark kann er seine eigene Umfrageantworten bestimmen und deren Richtigkeit prüfen.

4.3 Die verwendeten kryptographischen Funktionen

Unser Protokoll verwendet keine komplexen Verschlüsselungssysteme. Dennoch sollen die verwendeten

kryptographischen Funktionen erwähnt werden:

- **Hash einer Nachricht**
  Aus einer beliebig grossen Nachricht wird durch einen Hash-Algorithmus ein so genannter Fingerprint der Nachricht erstellt. Dieser Hash ist eine kurze möglichst eindeutige Identifikation fester Länge der Nachricht. Damit kann nach der Übertragung die Integrität der Nachricht gewährleistet werden. Stimmt der selbst erstellte Hash der Nachricht nicht mit dem beigelegten Hash überein, so wurde die Nachricht verändert.

- **Signieren einer Nachricht**
  Der Hash der Nachricht wird mit dem privaten Schlüssel des Senders verschlüsselt. Da nur der Unterzeichner seinen privaten Schlüssel kennt, kann nur er den Hash der Nachricht verschlüsselt haben und gilt so als ursprungssicher. Der Empfänger kann mit dem Hash der Nachricht und dem Entschlüsseln der Signatur die Nachricht verifizieren.

- **Verschlüsseln einer Nachricht**
  Die Nachricht wird mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. So kann nur der vorgesehene Empfänger mit seinem privaten Schlüssel die Nachricht entschlüsseln. Auch der Ersteller des Geheimtextes kann diesen nicht mehr entschlüsseln.

- **Permutation**
  Nehmen wir an, dass es eine Permutation P über eine endlichen Menge von Zahlen gibt, bei denen die Umkehrung schwer zu berechnen ist. Zum Beispiel bei einem gegebenen y wobei x sehr schwer zu berechnen ist. Für unsere Verwendung muss zudem sichergestellt werden, dass die Funktion injektiv ist, das heisst, für zwei verschiedene x ergibt es in jedem Fall zwei verschiedene y.

  $$x_1 \neq x_2 \rightarrow y_1 \neq y_2$$

- **Blinde Signatur**
  Der Zweck der blinden Signatur ist, dass eine Mitteilung von einem Unterzeichner signiert wird, ohne dass dieser den Inhalt der Mitteilung kennt.

- **Nachricht versalzen**
  Das Versalzen einer Nachricht ist nötig, wenn man diese blind signieren lassen will. Das heisst, eine Nachricht wird mit einem zufälligen Wert r multipliziert. Dieser versalzene Wert wird nun von einer anderen Partei X signiert. Dividiert man nun den erhaltenen Wert wieder mit r, erhält man die Signatur des ursprünglichen Wertes.

5 Resultate Diplomarbeit

Um den ganzen Ablauf etwas zu vereinfachen und die Anzahl der Agents gering zu halten, haben wir uns entschieden, einigen Agents mehrere Aufgaben zuzuteilen. So übernimmt der Customer – also der Auftraggeber – zusätzlich die Funktion des Ballot Distributors. Dies hat zwar einen Mehraufwand für den Customer zur Folge, hat aber den grossen Vorteil, dass die Poller vom Auftraggeber eingeladen werden und so direkt klar wird, wer die OLM initiiert hat. Dem Customer wird eine Applikation zur Verfügung gestellt, um die zusätzlichen Aufgaben erledigen zu können.

- Der Provider übernimmt alle Funktionen, die direkt mit der Umfrage zu tun haben und stellt den Pollern eine Applikation zur Verfügung, womit sie die OLM ausfüllen können.
- Der Customer erstellt beim Provider die OLM.
- Aus dem Protokoll übernimmt der Provider die Funktion des Vote Compilers und erstellt die Umfrageergebnisse.
- Die Certifying Authority (nicht zu verwechseln mit einer Certificate Authority, die digitale Zertifikate ausstellt) ist dafür zuständig, dass jeder Poller höchstens eine signierte Pollermark erhält.

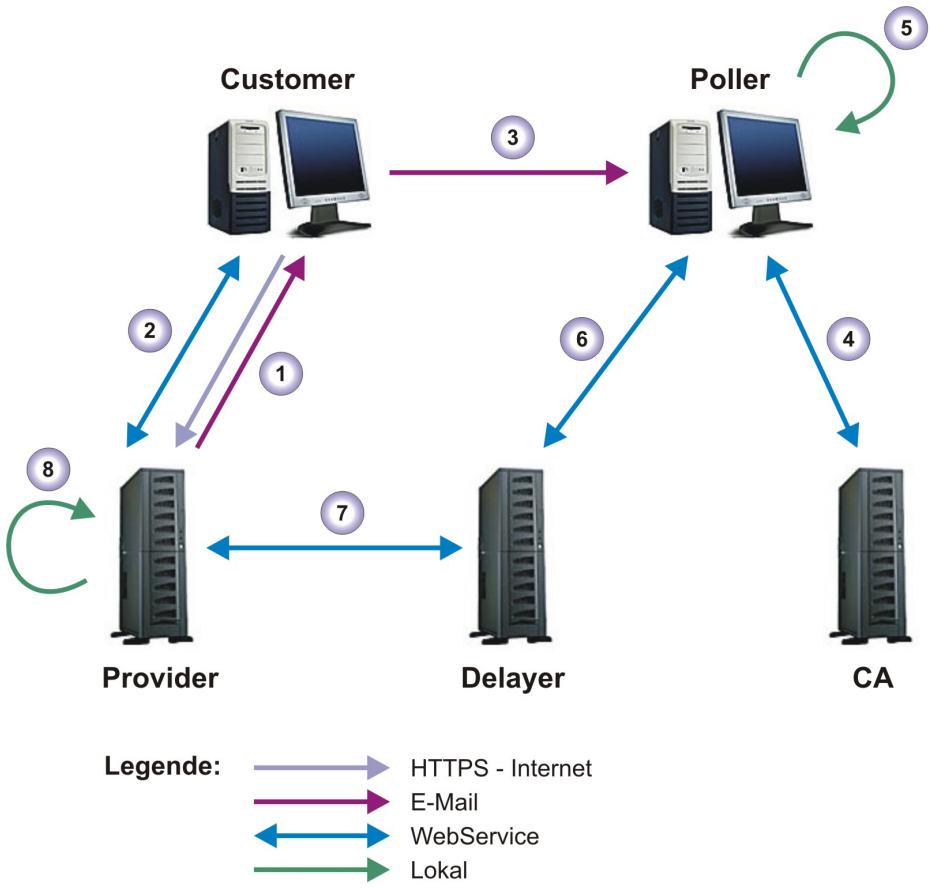


Abbildung 1: Übersicht On-Line Meinungsumfrage

Jede Partei besitzt ein Public-Key-Zertifikat (die Verteilung der Zertifikate wird in 5.3.1.Zertifikate beschrieben), mit welchem die Daten vor dem Versenden verschlüsselt werden. So können unsichere Transportprotokolle problemlos verwendet werden (ausser bei der Registrierung des Customers und der Publikation der Daten).

1 Registrierung des Customers beim Provider
  1.1 Registrierung ausfüllen (HTTPS)
  1.2 Logindaten (E-Mail)
2 Umfrage erstellen (WS)
3 Poller einladen (E-Mail)
4 Pollermark blind signieren (WS)
5 Umfrage ausfüllen (lokal)
6 Umfrage abschicken (WS)
7 Umfrageantworten holen (WS)
8 Auswertung (lokal)
9 Publikationen
  9.1 Customer (E-Mail)
  9.2 Certifying Authority (HTTPS)
  9.3 Provider (HTTPS)

5.1 Erreichte / nicht erreichte Projektziele

Die Anforderungen haben wir in die drei Kategorien Muss-, Soll- und Kannkriterien unterteilt. Die Musskriterien sind im Vergleich mit den beiden anderen Kriterien so enorm viel umfangreicher, dass die Nichtumsetzung der Soll- und Kannkriterien kaum ins Gewicht fallen. Die Anforderungen sind im Pflichtenheft [B.2] im Kapitel 5 beschrieben.

5.1.1 *Musskriterien*

Damit die Umfrage durchgeführt werden kann, müssen alle Punkte des Sicherheitskonzepts [B.1] implementiert werden. Die Musskriterien aus dem Pflichtenheft werden ein wenig detaillierter aufgezeigt.

Kriterium aus Pflichtenheft	neue Kriterienbezeichnung	Schritt
Umfrage erstellen	Registrierung des Customers beim Provider	1
	Umfrage erstellen	2
Poller einladen	Poller einladen	3
Pollermark blind signieren	Pollermark blind signieren	4
Umfrage ausfüllen	Umfrage ausfüllen	5
Umfrage abschicken	Umfrage abschicken	6
	Umfrageantworten holen	7
Publikationen	Umfrage auswerten	8
	Publikationen	9

Die Musskriterien des Prototyps sind (die Nummerierung ist die gleiche wie in der Abbildung Übersicht On-Line Meinungsumfrage):

1 **Registrierung des Customers beim Provider**

Um eine OLM erstellen zu können, muss sich der Customer beim Provider registrieren. Dies muss nur einmal gemacht werden. Die Registrierung geschieht in zwei Schritten:

1.1 **Registrierung ausfüllen**

Auf der Homepage des Providers kann der Customer ein Formular ausfüllen und sich registrieren lassen.

1.2 **Logindaten**

Sind alle Daten plausibel – der Username und die E-Mail-Adresse dürfen nur einmal verwendet werden – schickt der Provider dem Customer eine E-Mail mit Username und Passwort.

2 **Umfrage erstellen**

Der Customer kann mit der Customerapplikation eine On-Line Meinungsumfrage erstellen – Dazu muss er beim Provider registriert sein.

3 **Poller einladen**

Nach dem Erstellen einer On-Line Meinungsumfrage definiert der Customer die Menge der teilnehmenden Poller und lädt diese per E-Mail ein. Diese E-Mail enthält:

3.1 *generelle Informationen über die OLM*

3.2 *die Zertifikate der beteiligten Agents* (Customer und Provider)

3.3 *das vom Customer signierte Zertifikat des Pollers* (damit kann sich der Poller bei der CA authentifizieren)

3.4 *eine Seriennummer und die entsprechende Signatur* (mit der Seriennummer wird eine eindeutige Pollermark generiert)

3.5 ein XML-File mit der leeren Umfrage.

4 **Pollermark blind signieren**

Um die Anonymität des Pollers zu gewährleisten, darf nur der Poller selbst eine Verbindung zwischen seiner Identität und seiner Pollermark herstellen können. Deshalb wird die Pollermark von der CA blind signiert. Dies heisst, der Hash der Pollermark wird versalzen (mit einer Zufallszahl multipliziert) und signiert, danach kann die Signatur wieder entsalzen werden. Dies hat den Vorteil, dass die CA zwar die Identität des Pollers kennt, nicht aber seine Pollermark. Für das Signieren der Pollermark benötigt der Poller eine Internetverbindung.

5 **Umfrage ausfüllen**

Nachdem der Poller seine Pollermark signieren liess, kann er die Umfrage mit der Customerapplikation lokal auf seinem Computer ausfüllen. Da er dies lokal ausfüllt, braucht er keine Verbindung zum Internet.

6 **Umfrage abschicken**

Nach dem Ausfüllen der Umfrage, kann der Poller die Antworten mit der Pollerapplikation an den Delayer schicken, und die Arbeit ist für den Poller vorerst erledigt. Für diesen Schritt ist wieder eine Internet-Verbindung nötig.

7 **Umfrageantworten holen**

Der Provider lädt die Antworten nach Ablauf des Enddatums vom Delayer herunter und prüft die Korrektheit der Antworten (die Daten wurden nicht verändert, und die mitgelieferte Pollermark hat nicht schon zu einem früheren Zeitpunkt eine Antwort abgegeben).

8 **Umfrage auswerten**

Sobald alle Umfrageantworten vom Delayer heruntergeladen sind, werden die Antworten ausgewertet und die Gesamtstatistik erstellt.

9 **Publikationen**

Nach Beendigung der On-Line Meinungsumfrage publizieren verschiedene Agents Folgendes:

9.1 Der **Customer** publiziert die Anzahl Poller und für jeden Poller seine Seriennummer und Identität.

9.2 Die **CA** publiziert die Anzahl der signierten Pollermarks und den versalzten Hash der Pollermark mit der dazugehörenden Signaturen des Pollers und der CA sowie die Identität jedes Pollers.

9.3 Der **Provider** publiziert alle Antworten zusammen mit der Pollermark, sowie die Gesamtstatistiken.

5.2 Ziele und Lösungen

Die folgenden vier Ziele, die im Dokument Pflichtenheft [B.2] Kapitel 4 beschrieben sind, wurden während der Implementierung in der Diplomarbeit gelöst und werden deshalb hier beschrieben.

- **Interface zur Erstellung einer Umfrage**
  Dem Customer wird ein Java-Programm zum Erstellen einer OLM zur Verfügung gestellt.
- **Implementierung der Protokoll-Schritte mittels geeigneter Kommunikationsmethoden**
  Für die Kommunikation unter den verschiedenen Parteien werden so wenig verschiedene Protokolle wie möglich verwendet (E-Mail, HTTP, ...)
- **Erstellen einer funktionierenden Test-Umgebung**
  Als Test wird eine gesamte Umfrage durchgeführt mit Versuchen, die Umfrage zu manipulieren.
- **Beschreibung der Ausgangssituation**
  Es soll genau beschrieben werden, was bisher geleistet wurde und was nun in der Diplomarbeit gemacht wird.

Diese Ziele werden wie folgt sichergestellt:

5.2.1 *Interface zur Erstellung einer Umfrage*

Dem Customer wird eine Javaapplikation zur Verfügung gestellt. Mit dieser kann er sich On-Line Meinungsumfragen erstellen und diese verwalten. Die Daten werden nicht lokal gespeichert, sondern werden mittels Web Service an den Provider gesendet und von diesem verwaltet.

5.2.2 *Implementierung der Protokoll-Schritte mittels geeigneter Kommunikationsmethoden*

Ziel war es, möglichst wenig verschiedene Kommunikationsprotokolle zu verwenden. Dies ist sehr gut gelungen. Es werden nur Web Services, E-Mail und HTTPS verwendet:

- **Web Services (WS)**
  Die beiden Applikationen für den Customer und die Poller verwenden grösstenteils Web Services für die Kommunikation mit den Agents. Für diese Applikationen ist es unumgänglich, dass die Kommunikation synchron ist. Da sind Web Services geradezu prädestiniert.
- **E-Mail**
  Der Customer lädt die Poller mit einem E-Mail zur OLM ein. Dieses Kommunikationsmittel wurde gewählt, da E-Mails für jeden Computer-Benutzer etwas Bekanntes ist und zur Verfügung steht.
- **HTTPS**
  Für die Registrierung eines Customers beim Provider wird eine Webseite bereitgestellt. Auch die Publikationen des Providers und der CA werden auf einer Webseite dargestellt.

5.2.3 *Erstellen einer funktionierenden Test-Umgebung*

Die Test-Umgebung ist in unserem Fall der Prototyp selbst. Wegen des grossen Aufwands haben wir uns dazu entschieden, dass wir nur den Erfolgsfall testen. Das heisst, wir gehen davon aus, dass der Customer und die Poller alles so machen, wie es sein soll und auch keine Attakingversuche unternommen werden.

5.2.4 *Beschreibung der Ausgangssituation*

Die Ausgangssituation sind die Ergebnisse aus der Semesterarbeit. Dabei handelt es sich um die Erstellung des Sicherheitskonzepts [B.1], welches eine Zusammenfassung der Ergebnisse aus der Semesterarbeit enthält.

5.3 Annahmen

Für die einfachere und schnellere Implementierung des Prototypen haben wir uns dazu entschlossen, einiges als vorgegeben anzunehmen, da gewisse Dinge nicht zum Umfang des Protokolls gehören.

5.3.1 *Zertifikate*

Jeder Poller muss vor Beginn der On-Line Meinungsumfrage ein Public-Key-Zertifikat besitzen und der Customer muss im Besitz aller Zertifikate der Poller sein. Die selbst signierten Zertifikate der Agents und der Poller können nur für den Prototypen verwendet werden, da diese vor Beginn der OLM bekannt sind. Für eine produktive Verwendung müssten die Zertifikate natürlich von einer offiziellen Certificate Authority ausgestellt und signiert werden. Eine Anleitung zum Erstellen eines selbst signierten Zertifikats befindet sich im Anhang A Zertifikate erstellen.

Jeder (ausser dem Delayer), der am Ablauf der Umfrage beteiligt ist, benötigt Zertifikate seiner Kommunikationspartner. Entweder ist die Partei schon vor Beginn der Umfrage im Besitz bestimmter Zertifikate oder diese werden während dem Ablauf der Umfrage ausgetauscht.

Die Nummerierung in der Zeichnung entspricht der Nummerierung in der Abbildung Übersicht On-Line Meinungsumfrage.

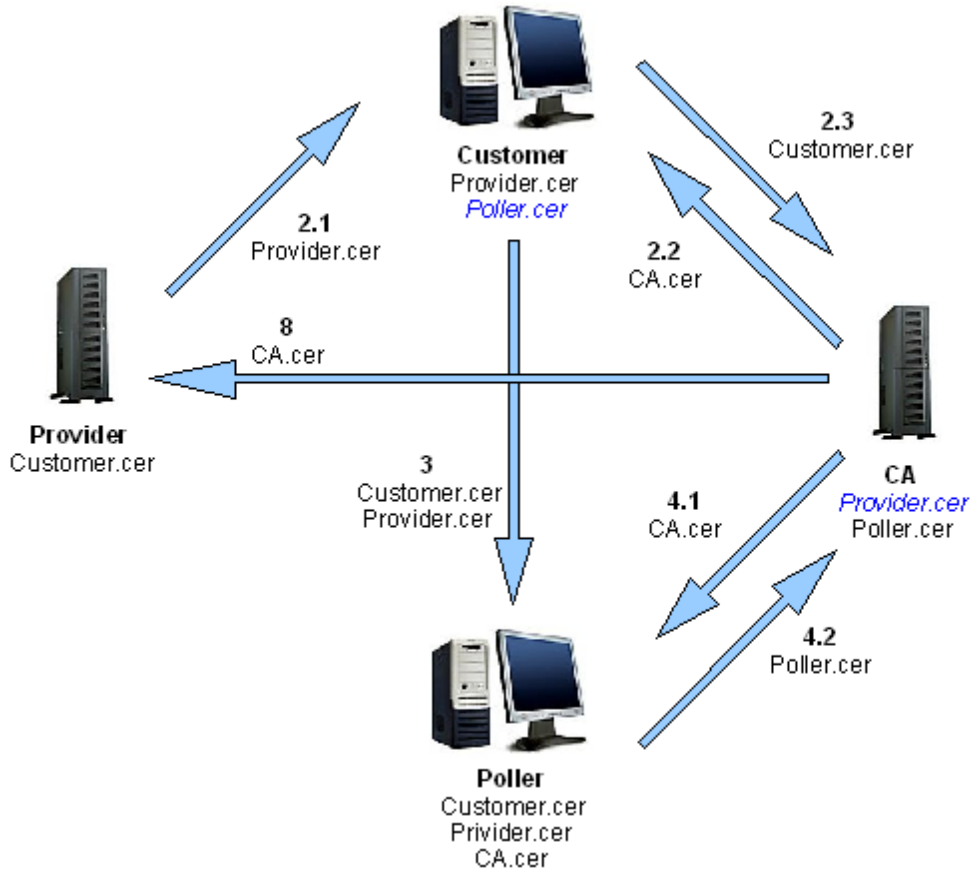


Abbildung 2: Verteilung der Zertifikate

Der Customer besitzt bereits alle Zertifikate der teilnehmenden Poller. Die Restlichen Zertifikate werden währen der Umfrage ausgetauscht:

2 Umfrage erstellen

2.1 Beim Starten der Customerapplikation wird eine Verbindung zum Provider aufgebaut und dessen Zertifikat verlangt. Damit werden die Daten für die Erstellung der Umfrage verschlüsselt.

2.2 Die Customerapplikation verlangt das Zertifikat der CA. Dieses wird verwendet um die Poll-ID zu Verschlüsseln und der CA mitzuteilen.

2.3 Bevor die Poller eingeladen werden, schickt die Customerapplikation die Poll-ID und das Zertifikat des Customers an die CA. Mit dem Zertifikat kann die CA die signierten Pollerzertifikate verifizieren.

3 Mit der Einladung des Pollers durch den Customer schickt dieser jedem einzelnen Poller sein eigenes Zertifikat (Customer.cer) und das Zertifikat der Providers, dieses verwendet der Poller zum verschlüsseln der Umfrageantworten.

4 Der Poller lässt sich seine Pollermark von der CA blind signieren.

4.1 Beim Starten der Pollerapplikation wird eine Verbindung zur CA aufgebaut und das Zertifikat der CA verlangt.

4.2 Lässt sich der Poller seine Pollermark von der CA signieren, sendet er auch sein Zertifikat mit. Dieses verwendet die CA zum einen für das Authentifizieren des Pollers mit dem signierten Pollerzertifikat des Customers, zum anderen wird die Pollermark mit ihrer Signatur verifiziert.

8 Damit der Provider die Signatur der Pollermark verifizieren kann, benötigt er das Zertifikat der CA. Die Pollermark mit Signatur erhält der Provider vom Delayer.

5.4 Pendenzen

Aus zeitlichen Gründen haben wir uns dazu entschieden, diejenigen Punkte zuerst zu erledigen, die zum normalen Ablauf der Umfrage gehören. Einige andere Punkte – im Gesamtrahmen des Protokolls nicht weniger wichtige – die aber zu Kontrollzwecken verwendet werden und nicht unbedingt erforderlich sind, haben wir noch nicht implementiert.

5.4.1 *Signierte E-Mails*

Die E-Mails, die der Customer an die Poller schickt, müssen natürlich signiert sein. Die E-Mails, die im Rahmen des Prototypen verschickt werden, enthalten keine Signatur des Customers. Um eine Umfrage durchführen zu können, ist dies auch nicht nötig. Für eine produktive Verwendung des Systems ist dies aber unumgänglich, da sich der Poller nur so der Identität des Customers sicher sein kann.

5.4.2 *Verfälschte Daten*

Die eingetroffenen Antworten der Poller beim Provider müssen laut Protokoll auf ihre Richtigkeit geprüft werden. Der Prototyp geht davon aus, dass die Daten unverfälscht sind.

5.4.3 *Publikationen*

5.4.3.1 *Customer*

Der Customer soll nach Beendigung der On-Line Meinungsumfrage eine E-Mail an alle Poller schicken, mit der Anzahl der Teilnehmer und für jeden Teilnehmer das Identifikationskennzeichen und die Seriennummer. Dies ist bis jetzt noch nicht möglich, die Funktion soll in die Customerapplikation eingebaut werden.

5.4.3.2 *Provider*

Die Publikationen die der Provider bereit stellen muss, um dem Poller zu versichern, dass seine Antworten in die Gesamtergebnisse eingeflossen sind, ist noch nicht implementiert. Auf dieser Seite sollte der Poller die Signatur seiner Antworten und der signierten Pollermark ausgestellt vom Provider finden. Diese beiden Signaturen könnte er dann mit dem Zertifikat der Providers verifizieren.

5.4.4 *HTTPS*

Die Webseiten des Providers und der CA, sollten mittels HTTPS übertragen werden. Da es einfacher ist die Daten mit HTTP zu transportieren, haben wir es so gemacht. Die Publikationsseiten müssen sicher sein, damit ein Poller weiss, dass dies die richtigen Seiten der Agents sind.

5.4.5 *Website Provider*

Der Provider stellt eine Website zur Verfügung, von denen der Customer und die Poller ihre jeweilige Applikation herunterladen können. Momentan sind die Applikationen nur auf der CD verfügbar.

5.4.6 *signierte Software*

Damit der Customer und die Poller sicher sein können, dass sie die original Software verwenden und nicht einer Fälschung zum Opfer fallen, müssen die Applikationen noch signiert werden.

5.4.7 *VPN für E-Mail*

Benutzt der Customer eine E-Mail-Adresse der BFH-TI zum Einladen der Poller, muss eine VPN Connection bestehen. Dies sollte noch verbessert werden.

5.5 Bekannte Fehler

Im ganzen Ablauf eine On-Line Meinungsumfrage sind noch einige kleine Fehler, die wir nicht mehr eliminieren konnten:

- Falls der Poller die Pollerapplikation startet und seine Pollermark signieren lässt, das Zipfile aber aus irgend einem Grund geöffnet hat, kann die Pollerapplikation nicht in das Zipfile schreiben. Aus der Sicht der CA hat der Poller eine Signatur erhalten und kann dies nicht wiederholen. Der Poller andererseits muss die Pollerapplikation neu starten und muss die Pollermark noch einmal signieren lassen. Daraus resultiert, das der Poller die On-Line Meinungsumfrage nicht ausfüllen kann.

- Der Poller kann die On-Line Meinungsumfrage auch ausfüllen, wenn der Endtermin der Umfrage schon vorbei ist. Die Antworten werden in das Gesamtergebnis nicht einbezogen, jedoch sieht der Poller nicht direkt, dass seine Arbeit vergebens ist. Optimalerweise sollte die Prüfung in der Pollerapplikation, bei der CA und beim Delayer gemacht werden, so sind alle Möglichkeiten abgedeckt.

- Die Signatur der Pollermark wird in der Pollerapplikation verifiziert. Falls die Verifizierung fehl schlägt – was natürlich niemals vorkommen sollte – wird die Verarbeitung nicht abgebrochen, sondern läuft normal weiter.

- Sind die abgegebenen Antworten eine Pollers aus irgend einem Grund ungültig, werden diese nicht in die Gesamtstatistik aufgenommen. Dies ist korrekt, jedoch gibt es noch keinen Mechanismus, der diese fehlerhaften Antworten aufnimmt und Alarm schlägt.

- Zum Signieren der Pollermark wird der Hash in einen BigInteger konvertiert, um die Berechnungen durchzuführen. Wird diese Zahl negativ, kann die Verifizierung fehl schlagen, obwohl die Signatur stimmt. Dies kann dazu führen, dass eine korrekte Antwort nicht ausgewertet wird.

6 Kommunikationswege

Während der Durchführung einer On-Line Meinungsumfrage werden viele Daten hin und her geschickt. In diesem Kapitel wird genau beschrieben, wer mit wem kommuniziert und was mitgesendet wird.

6.1 Registrierung

Jeder Customer kann nur ein Account beim Provider erstellen. Das heisst konkret, der Benutzername und die E-Mail-Adresse müssen eindeutig sein. Natürlich kann sich jemand mehrmals registrieren, jedoch unter Verwendung eines unterschiedlichen Benutzernamens und einer unterschiedlichen E-Mail-Adresse.

6.1.1 *Registrierung ausfüllen*

HTTPS	C → Pro:	{Username, E-Mail, Name, Vorname, Strasse, PLZ, Ort}

Die Registrierung erfolgt über die Internetadresse http://webumfrage.bfh.ch:8080/register Auf dieser Web-Seite muss der Customer nun die gewünschten Informationen eingeben.

6.1.2 *Logindaten*

E-Mail	Pro → C:	{username, password}

Nach einer Prüfung der Customerdaten (username eindeutig, E-Mail-Adresse eindeutig, alle obligatorischen Felder vorhanden) wird das Passwort generiert und per E-Mail an den Customer geschickt.

6.2 Customer

6.2.1 *Übersicht*

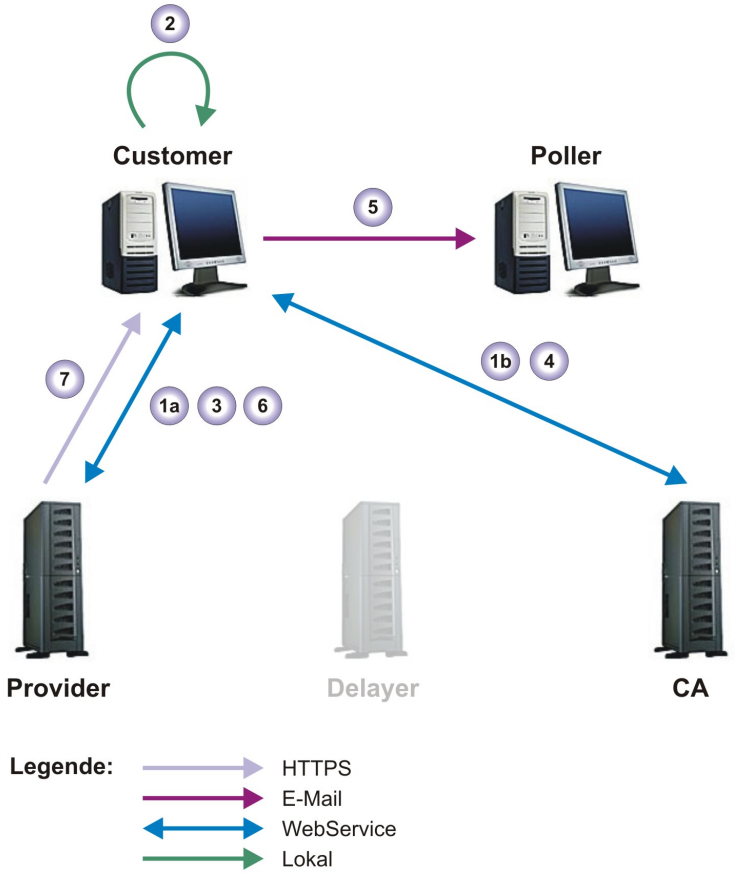


Abbildung 3: Ablauf Customer

6.2.2 *Schritt 1: Customer – Applikation starten*

WS	Pro ↔ C:	ProviderCert

Nachdem sich der Customer die Customerapplikation vom Provider heruntergeladen hat, entpackt und gestart hat. Wird als erstes das Zertifikat des Provider geholt. (1a). Dieses wird verwendet um den späteren Datenaustausch, mit dem Provider, zu verschlüsseln.

WS	CA ↔ C:	CACert

Als nächstes wird das Zertifikat der CA heruntergeladen (1b). Dieses wird für den Schritt 4 verwendet.

Um die Applikation verwenden zu können, muss sich der Customer mit seinem Usernamen und Password einloggen, die Daten werden mittels Web Service übertragen und vom Provider verifiziert. Bei jeder Kommunikation mit dem Provider werden die Login Daten mitgeschickt.

6.2.3 *Schritt 2: Umfrage erstellen*

WS	C ↔ Pro:	key, username, h(password)

Nach dem einloggen erscheint eine Übersicht mit allen erstellten Umfragen. Die Umfragen werden als Mehrdimensionales Array dem Customer gesendet.

WS	Pro ↔ C:	{
		{ poll_id, status, start_time, end_time },
		...
		}

Dem Customer stehen 4 Möglichkeiten zur Auswahl:

6.2.3.1 *Neue Umfrage erstellen*

WS	C ↔ Pro:	key, username, h(password)

Der Customer sendet dem Provider eine Mitteilung, wie bereits erwähnt werden bei allen Mitteilungen die Login-Daten mitgeschickt.

WS	Pro ↔ C:	poll_id

Als Antwort erhält der Customer die Poll-ID der neuen Umfrage.

6.2.3.2 *Umfrage entfernen*

WS	C ↔ Pro:	key, username, h(password), poll_id

Der Customer sendet dem Provider eine Mitteilung inkl. der Poll-ID der Umfrage, die entfernt werden soll.

Als Antwort erhält er ob das entfernen erfolgreich war oder nicht..

6.2.3.3 *Anzeigen der Ergebnisse*

HTTPS	C → Pro:	http://vmaris.bfh.ch:8080/Provider-Publication/login.faces?Poll-ID= poll_id

Sobald eine Umfrage beendet wurde besteht die Möglichkeit die Auswertung der Umfrageergebnisse zu betrachten. Anhand der Poll-ID wird die dazugehörende URL erstellt.

6.2.3.4 *Umfrage editieren*

WS	C ↔ Pro:	key, username, h(password), poll_id

Die Poll-ID der Umfrage welche editiert werden soll, wird dem Provider gesendet.

WS	Pro ↔ C:	{
		{ title, description, end_time },
		poll
		}

Als Antwort erhält der Customer ein Mehrdimensionales Array. Der Titel, die Beschreibung und das End-Datum wurde als bequemlichkeit zusätzlich zur Umfrage (poll) mitgeschickt.

6.2.4 *Schritt 3: Umfrage speichern*

Nachdem der Customer die Umfrage mit allen benötigten Frage erstellt hat. Kann er die Umfrage speichern.

WS	C ↔ Pro:	key, username, h(password)
		{
		poll_id,
		{ title, description, end_time },
		poll
		}

Als Antwort erhält er ob das Speichern erfolgreich war oder nicht.

6.2.5 *Schritt 4: Vor dem Einladen der Poller*

WS	C ↔ CA:	caKey
		{
		poll_id,
		CustomerCert
		}

Nachdem der Customer die Umfrage mit allen benötigten Frage erstellt und gesepichert hat, kann er eine Liste der Poller - als Identifikation eines Pollers wird der Distinguished Name aus den Zertifikaten benutzt werden – angeben. Aber bevor die Einladung der Teilnehmer erfolgt, sendet der Poller der CA sein Zertifikat zusammen mit der Poll-ID. Die Daten werden mit dem öffentlichen Schlüssel der CA – das Zertifikat wurde im Schritt 1b geholt – verschlüsselt. Das Zertifikat des Customers verwendet die CA später beim signieren der PollerMark (Poller).

6.2.6 *Schritt 5: Einladen der Poller*

E-Mail	C → P:	Z_C, Z_{Pro}, [{y; [h(y), C_d]; [h(Z_P), C_d]}, P_e], poll

Der Customer lädt nun alle Poller per E-Mail zur OLM ein. Mit den allgemeinen Informationen zur OLM schickt der Customer folgende Dateien mit (die Fett dargestellten Daten werden verschlüsselt verschickt):

- Zertifikate der Agents (Customer, Provider)
- **Seriennummer y**
- **Signatur der Seriennummer [h(y), C_d]**
- **Signatur des Pollerzertifikats [h(Z_P), C_d]**
- die leere Meinungsumfrage (poll)

6.2.7 *Schritt 6: Umfrage gestartet*

WS	C ↔ Pro:	key, username, h(password), poll

Gleich nachdem die Einladung beendet wurde wird dem Provider mitgeteilt das die Umfrage begonnen hat. Der Provider erstellt anhand der Umfrage (poll) die JCR Struktur, in der später die Antworten gespeichert werden

6.2.8 *Schritt 7: Publikation*

HTTPS	Pro → C:	http://vmaris.bfh.ch:8080/Provider-Publication/login.faces?Poll-ID= poll_id

Wurde die Umfrage beendet so kann der Customer die Ergebnisse beim Provider betrachten.
(siehe 6.2.3.3.Anzeigen der Ergebnisse)

6.3 Poller

6.3.1 *Übersicht*

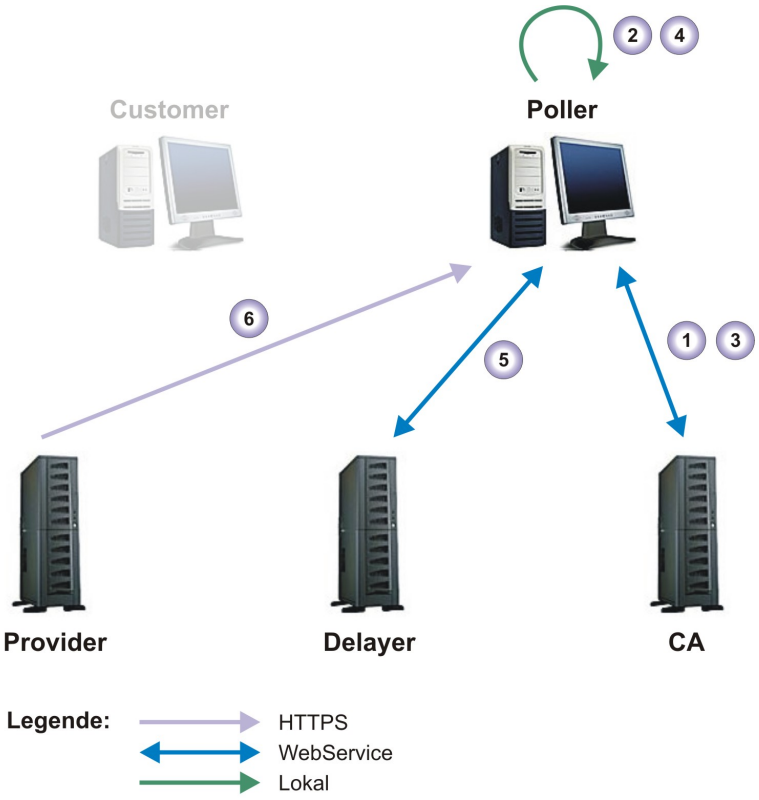


Abbildung 4: Ablauf Poller

6.3.2 *Schritt 1: Poller – Applikation starten*

WS	P ↔ CA:	ProviderCert

Entscheidet sich der Poller an der OLM teilzunehmen, so muss er zuerst die Pollerapplikation vom Provider heruntergeladen. Nachdem er sie entpackt und gestart hat, wird als erstes das Zertifikat der CA geholt. Dieses wird verwendet um den späteren Datenaustausch – Pollermark blink signieren - zu verschlüsseln.

6.3.3 *Schritt 2: Pollermark erstellen*

Nachdem sich der Poller das *.olm File – Einladung durch Customer – aus dem E-Mail in das Verzeichniss der Applikation kopiert hat, die Applikation gestartet und die Umfrage geladen hat. Kann er seine eigene Pollermark lokal erstellen.

6.3.4 *Schritt 3: Pollermark blind signieren*

WS	P ↔ CA:	caKey,
		{
		poll_Id, [{h(m)×[r,CA_e]; [h(m), P_d]; P; [h(Z_P), C_d]; Z_P}, CA_e],
		[[h(m)×[r,CA_e], CA_d], P_e]
		}

Die nun lokal erstellte Pollermark soll von der CA signiert werden. Dazu schickt der Poller folgende Daten an die CA (die Fett dargestellten Daten werden verschlüsselt verschickt.):

- Umfrageidentifikation poll_Id
- **versalzter Hash der Pollermark h(m)×[r,CAe]**
- **Signatur der Pollermark [h(m), P_d]**
- **Polleridentifikation P**
- **Signatur des Pollerzertifikats [h(Z_P), C_d]**
- **Pollerzertifikat Z_P**

Und erhält die Signatur des versalzten Hashes der Pollermark **[h(m)×[r,CA_e], CA_d]** zurück.

Die Ausgangszahl x besteht aus zwei Teilen x = y + rnd ('+' ist hier eine Konkatenation): Der Seriennummer (y) – die sicherstellt, dass jeder Poller eine andere Ausgangszahl erstellt und einer zufälligen Bitfolge aus 128 Bit (rnd). Die 128 Bit sind nötig, da mit Kenntnis der Seriennummer nicht alle möglichen Kombinationen mittels Brute Force erstellt werden können und man so auf den Inhaber der Pollermark schliessen könnte. Aus x wird nun mit einem Generator Modulus Paar – bekannt aus dem Diffie-Hellman-Algorithmus – die Pollermark m generiert. Von der Pollermark (m) wird dann der Hash Hm berechnet und durch die Certifying Authority signiert. Damit die Certifying Authority aber keine Verbindung zwischen Poller und Hash der Pollermark herstellen kann, wird Hm mit r versalzen (Hm_v) und von der CA blind signiert. Mit dem multiplikativ inversen Element von r in Z_n wird dann aus Hm_vS_{CA} die signierte Pollermark H_mS_{CA} berechnet.

Für genauere Informationen siehe 7.Anonymisierung.

6.3.5 *Schritt 4: Umfrage ausfüllen*

Nun kann der Poller Umfrage lokal ausfüllen. Da die Pollerapplikation keine Verbindung mit dem Provider aufbaut, ist auch die Identität des Pollers für den Provider nicht bekannt. Nach dem Ausfüllen der OLM wird ein XML File mit allen Fragen und Antworten erstellt. Dieses File enthält keinen Text sondern nur den Index oder die Id der Labels.

6.3.6 *Schritt 5: Umfrage abschicken*

WS	P ↔ CA:	{
		poll_id, [providerKey, P_e] , m,
		[m, CA_d], [h(m), CA_d]
		h(completed_poll), completed_poll
		}

Nun kann er die Antworten (completed poll) zusammen mit seiner Pollermark m, der signierten Pollermark [h(m), CA_d] und dem Hash der Antworten und signierten Pollermark h(completed poll; [h(m), CA_d]) mit dem öffentlichen Schlüssel des Providers verschlüsseln und an den Delayer schicken.

6.3.7 *Schritt 6: Publikation*

HTTPS	Pro → P:	http://vmaris.bfh.ch:8080/Provider-Publication/login.faces?Poll-ID= poll_id

Wurde die Umfrage beendet so kann der Poller die Ergebnisse beim Provider betrachten.

6.4 Provider

6.4.1 *Umfrageantworten holen- Statistik erstellen*

WS	D → Pro:	[{completed_poll; m; [h(m), CA_d]; h(completed_poll; [h(m), CA_d])}, Pro_e]

Periodisch überprüft der Provider ob eine Umfrage beendet wurde. Nach Ablauf lädt der Provider die Umfrageantworten vom Delayer herunter und erstellt die Auswertung der Umfrage. Die Antworten werden im JCR abgelegt damit sie später von der Publikation angeziegt werden können. Der Umweg über den Delayer findet statt, damit keine Verbindung zwischen Polleridentifikation und Pollermark hergestellt werden kann, falls die CA und der Provider zusammenarbeiten würden.

6.5 Publikationen

Als letzter Schritt veröffentlichen die Agents bestimmte Informationen, damit kann jeder Poller sicherstellen, dass seine Antworten auch in den Umfrageergebnissen enthalten sind.

6.5.1 *Customer*

E-Mail	C → P:	Poller {P, y}, number of Pollers

Der Customer veröffentlicht für jeden Poller sein Identifikationszeichen P und seine Seriennummer y. Dazu wird die Anzahl eingeladener Poller publiziert.

6.5.2 *Certifying Authority*

HTTP	CA → P:	Poller {h(m)×[r,CA_e], [h(m), Pd], [h(m)×[r,CA_e], CA_d], P}, number of signed poller marks

Die Certifying Authority veröffentlicht für jeden Poller den versalzenen Hash der Pollermark h(m)×[r,CAe], die Signatur der Pollermark [h(m), Pd], die Signatur des versalzten Hashes der Pollermark [h(m)×[r,CAe], CAd] und die Polleridentifikation P. Zudem die Anzahl signierter Pollermarks, welche kleiner oder gleich der eingeladenen Poller sein muss.

6.5.3 *Provider*

HTTP	Pro → P:	Poller {completed poll, m}, number of completed polls, poll result

Der Provider signiert die Umfrageergebnisse zusammen mit der signierten Pollermark und veröffentlicht diese. Zudem wird das Gesamtergebnis publiziert, so kann jeder Poller alle anderen Umfrageergebnisse auch betrachten und anhand der Pollermark seine Antworten ausfindig machen sowie sein Resultat verifizieren.

6.6 Überprüfungen

6.6.1 *Poller*

Anhand der Veröffentlichungen kann der Poller sicherstellen, dass sein Umfrageresultat mitgezählt wird. Er kann sogar seine Antworten mit der Pollermark identifizieren und kontrollieren.

6.6.2 *unautorisierter Poller*

Mit m und HmS_{CA} kann der Provider überprüfen, dass

1. P autorisiert ist (m mit CA_e entschlüsseln ergibt Hm den Hash von m).
2. P zum ersten Mal abstimmt.

Dies geschieht in 6.4.1. Umfrageantworten holen.

7 Anonymisierung

7.1 Erstellen und Signieren einer Pollermark

Ein zentraler Punkt des Protokolls ist, dass der Poller anonym seine Umfrage ausfüllen kann, aber trotzdem sichergestellt werden kann, dass seine Umfrageergebnisse mitgezählt werden und er nur einmal teilnehmen kann. Dies geschieht mit der signierten Pollermark und wird hier Schritt für Schritt anhand eines Beispiels mit kleinen Zahlen beschrieben.

Bedeutung der Variablen:

y	Seriennummer, erhalten vom Customer
rnd	Zufallszahl, selbst generiert vom Poller
x	Ausgangszahl zur Berechnung der Pollermark, Konkatenation von y und rnd
P(x)	Funktion, die die Pollermark berechnet
g	Generator der Funktion P
n	Modulus
r	Zufallszahl, zum Versalzen des Hash der Pollermark
r^{-1}	Multiplikativ inverses Element von r in Z_n, zum entsalzen der signierten versalzten Pollermark
m	Pollermark
Hm	Hash der Pollermark
Hm_v	Versalzener Hash der Pollermark
Hm_vS_{CA}	Signatur des versalzenen Hashs der Pollermark
HmS_{CA}	Signatur der Pollermark
CA_e	Öffentlicher Schlüssel der CA
CA_d	Privater Schlüssel der CA

7.1.1 *Eindeutige Ausgangszahl x Generieren*

Zuerst muss sich der Poller eine eindeutige, nicht nachvollziehbare Ausgangszahl x generieren. x besteht aus zwei Teilen, der Seriennummer y und einer zufälligen Bitfolge rnd aus 128 Bit. Die Seriennummer y wird verwendet um sicherzustellen, dass jeder Poller eine andere Pollermark erhält. Die Anonymität wird dadurch gewährleistet, dass sich jeder Poller eine zufällige Bitfolge erzeugt. Diese muss mindestens 128 Bit lang sein, damit man mit Kenntnis der Seriennummer mittels Brute Force nicht dieselbe Pollermark erstellen kann und somit eine Beziehung zur Identifikation des Pollers herstellen kann.

```
y = 7, rnd = 5
x = 75
```

7.1.2 *Pollermark Erstellen*

Aus x kann er nun mit Hilfe einer injektiven, nicht invertierbaren Funktion die Pollermark m erstellen, die

dank der Seriennummer eindeutig ist aber auch keinem Poller mehr zugewiesen werden kann. Diese Permutation wird mit einem mitgeliefertem Generator-Modulus Paar – bekannt aus dem Diffie-Hellmann-Algorithmus – durchgeführt: P: $m = g^x \bmod n$

$m = P(x) = 41^{75} \bmod 143 = 21$ mit $g = 41$ und $n = 143$

Die Erstellung eines Generator-Modulus Paares ist im Anhang B Generator-Modulus Paar erstellen beschrieben.

7.1.3 *Hash*

Das Erstellen des Hashes der Pollermark ist im original Protokoll nicht vorgesehen. Wir machen dies aus folgendem Grund: Der Poller schickt seine Umfrageergebnisse mit seiner Pollermark m und der signierten Pollermark (im original ohne Hash [m, CA_d]) an den Vote Compiler. Nun kann der VC mit dem öffentlichen Schlüssel der CA aus [m, CA_d] wieder die Pollermark herstellen, und diese muss gleich sein wie m. Nun könnte sich der Poller überlegen: "Ich ändere einige Bits meiner signierten Pollermark und erstelle mir meine neue gefälschte Pollermark m' selbst (dies ist mit dem CA_e kein Problem) und schicke meine Umfrageergebnisse noch einmal ab." Der Vote Compiler kann diesen Betrug nur entdecken, falls m' gleich einer anderen Pollermark ist oder alle Poller teilnehmen würden, und so zu viele Umfrageergebnisse eintreffen würden. Beide genannten Möglichkeiten sind aber höchst unzuverlässig, deswegen haben wir uns entschieden den Hash der Pollermark zu erstellen und diesen blind signieren zu lassen. So kann kein Poller aus [h(m), CA_d] das dazugehörige m bestimmen.

$Hm = h(m) = 28$

7.1.4 *Hm Versalzen*

Damit die Pollermark für die OLM brauchbar wird, muss sie von der CA signiert werden. Nur darf die CA nicht in den Besitz der Pollermark und der Identität des Pollers kommen, da sonst die Anonymität verloren geht. Deswegen wird Hm zuerst mit einer Zufallszahl r versalzen, das heisst mit r multipliziert: V: $Hm_v = Hm * r^{CAe} \bmod n$. Nun kann die CA keine m zu irgendeinem Hm_v zuordnen und so auch den Besitzer von m nicht bekannt geben.

$Hm_v = V(Hm) = 28 * 17^{23} \bmod 143 = 20$ | mit $r = 17$ und $CA_e = 23$

7.1.5 *Blind Signieren*

Der Poller sendet nun Hm_v, Hm_vS_P, sein Zertifikat Z und das vom BD signierte Zertifikat ZS_{BD} an die CA. Mit Z, ZS_{BD} und P_e kann die CA den Poller als berechtigten Poller verifizieren und feststellen, ob er zuvor schon eine Pollermark signiert hat, da er die erhaltenen Dateien in einer Liste abspeichert. Mit Hm_v, Hm_vS_P und P_e stellt die CA fest, dass die Daten fehlerfrei übertragen worden sind. Ist bisher alles in Ordnung, signiert die CA Hm_v und schickt Hm_vS_{CA} an den Poller zurück. S: $Hm_vS_{CA} = Hm_v^{CAd} \bmod n$

$Hm_vS_{CA} = S(Hm_v) = 20^{47} \bmod 143 = 15$ | mit $CAd = 47$

7.1.6 *HmvSCA Entsalzen*

Mit dem multiplikativ inversen Element von r in Zn kann der Poller das "Salz" entfernen und erhält seine signierte Pollermark HmS_{CA}. Mit Hm und CA_e kann er HmS_{CA} verifizieren. E: $HmS_{CA} = Hm_vS_{CA} * r^{-1} \bmod n$

$HmS_{CA} = E(Hm_vS_{CA}) = 15 * 101 \bmod 143 = 85 \quad | \text{ mit } r\text{-}1 = 101$

7.1.7 *Kontrolle*

Zur Kontrolle wird nun die Signatur der Pollermark mit dem öffentlichen Schlüssel der CA verifiziert. Dieser Wert muss gleich dem Hash der Pollermark sein.

$h(m) = 28$

$v = HmS_{CA}{}^{Cae} \bmod n = 85^{23} \bmod 143 = 28$

Somit stimmt die Aussage h(m) = v und die Signatur ist gültig.

7.2 Blind Signieren (Theorie)

Wie oben gesehen, kann man eine Zahl w signieren lassen, ohne dass der Signierer den Wert von w lesen kann. Wie ist dies möglich? Mathematisch gesehen ist der Umstand dafür verantwortlich, dass es in einer Restklasse n zu jedem Element, das teilerfremd zu n ist, ein multiplikativ inverses Element gibt. Für n wird in unserem Fall der Modulus aus dem Zertifikat genommen, dieser ist das Produkt aus zwei Primzahlen p und q. So dürfen genau diese zwei Zahlen nicht als r verwendet werden.

*	0	1	2	3	4
0	0	0	0	0	0
1	0	**1**	2	3	4
2	0	2	4	**1**	3
3	0	3	**1**	4	2
4	0	4	3	2	**1**

Für jedes Element (1 – 4) gibt es genau ein multiplikativ inverses Element

$e * e^{-1} \bmod n = 1$ z. B.:

$1 * 1 \bmod 5 = 1$

$2 * 3 \bmod 5 = 1$

Zur Berechnung werden folgende Werte verwendet:

w	zu signierender Wert
Z_e	öffentlicher Schlüssel des Signierers
Z_d	privater Schlüssel des Signierers
n	Modulus des Signierers
r	Zufällig gewählte Zahl kleiner n
r^{-1}	multiplikativ inverses Element zu r in der Restklasse n

Die Berechnung sieht folgendermassen aus:

1. Wert versalzen

   $w_v = w * r^{Ze} \bmod n$

2. versalzter Wert blind signieren

   $w_{vs} = w_v{}^{Zd} \bmod n$

3. blind signierter Wert entsalzen

$$w_s = w_{vs} * r^{-1} \bmod n$$

Nun wird gezeigt, wie r aus der Berechnung verschwindet:

$$\begin{aligned} w_s &= ((w * r^{Ze})^{Zd}) * r^{-1} \bmod n \\ &= (w^{Zd} * r^{Ze*Zd}) * r^{-1} \bmod n \quad | \; r^{Ze*Zd} \text{ wird in der Restklasse n zu } r^1, \text{ also r} \\ &= w^{Zd} * r * r^{-1} \bmod n \quad | \; r * r^{-1} \bmod n = 1 \\ &= w^{Zd} \bmod n \end{aligned}$$

So erhält der Inhaber von w die Signatur von w, aber der Wert von w muss dem Unterzeichner nicht bekannt gemacht werden.

8 Implementierung

Für die Implementierung unseres Sicherheitsprotokolls haben wir uns entschieden Java als grundlegende Softwaretechnologie zu verwenden. Dazu haben wir verschiedene zusätzliche Java API verwendet. Auch die von uns selbst erstellte Software gründet auf Java.

8.1 Verwendete APIs

Für die Implementierung wurde Java Version x.x als Basissprache verwendet, dazu wurden mehrere Java-APIs hinzugefügt:

- **JCR Java Content Repository [C.3]**
  Um die Daten bei den Agents zu speichern, verwenden wir das Java Content Repository.
- **Java Cryptography Extension (JCE) [C.4]**
  JCE dient als Grundlage für die kryptographischen Verschlüsselungsverfahren. JCE ist Bestandteil der Java Enterprise Edition (Java EE).
- **JavaMail [C.1]**
  Für das Verschicken von E-Mails wird die JavaMail API verwendet.
- **Tomcat [C.5]**
  Als Webserver wurde Apache Tomcat 5.5.x verwendet.
- **Axis [C.6]**
  Für die Web Services wurde die Implementierung von Apache Axis verwendet. Axis läuft unter Tomcat.
- **Java Server Faces (JSF) [C.7]**
  Die Registrierung des Customers und die Publikationen des Providers und der Certifying Authority werden mit Webseiten gelöst. Zur Darstellung dieser Webseiten wird JSF verwendet.

8.2 Erstellte Software

Unsere selbst erstellten Javaklassen befinden sich in den Java Projekten beginnend mit ch.bfh.webumfrage. Sie sind so aufgebaut, dass mehrfach verwendete Klassen in einem Basisprojekt sind und dies von den konkreten Projekten verwendet werden. Das nachfolgende Diagramm zeigt die vielen Abhängigkeiten zwischen den einzelnen Projekten auf.

Abbildung 5: Abhänigkeiten zwischen den Projekten

Legende für die Package- und Klassendiagramme:

Derive	Klasse erbt aus der Superklasse.
Implement	Klasse implementiert das Interface.
Instantiate	Instanziert ein neues Objekt der Klasse.
Create	Erstellt ein neues Objekt mit einer Factoryklasse.
Call	Ruft eine Methode der Klasse auf.

Send	Wirft eine Exception der entsprechenden Klasse.
Access	Greift auf ein öffentliches Feld der Klasse zu.
Import	Verwendet die Klasse oder das Interface.

Die Projekte lassen sich in verschiedene Kategorien einteilen:

- **Basisprojekte**
  Die Basisprojekte enthalten Pakete, die keine Abhängigkeiten haben, aber von vielen anderen Projekten verwendet werden. Innerhalb dieser Projekte bestehen nur sehr geringe Beziehungen.
  - **ch.bfh.webumfrage.common**
    Enthält verschiedene allgemeine Pakete, die den anderen Projekten zur Verfügung gestellt werden. So für Security (verschlüsseln, signieren, hashen), Serialisierung, Versenden von E-Mails, usw.

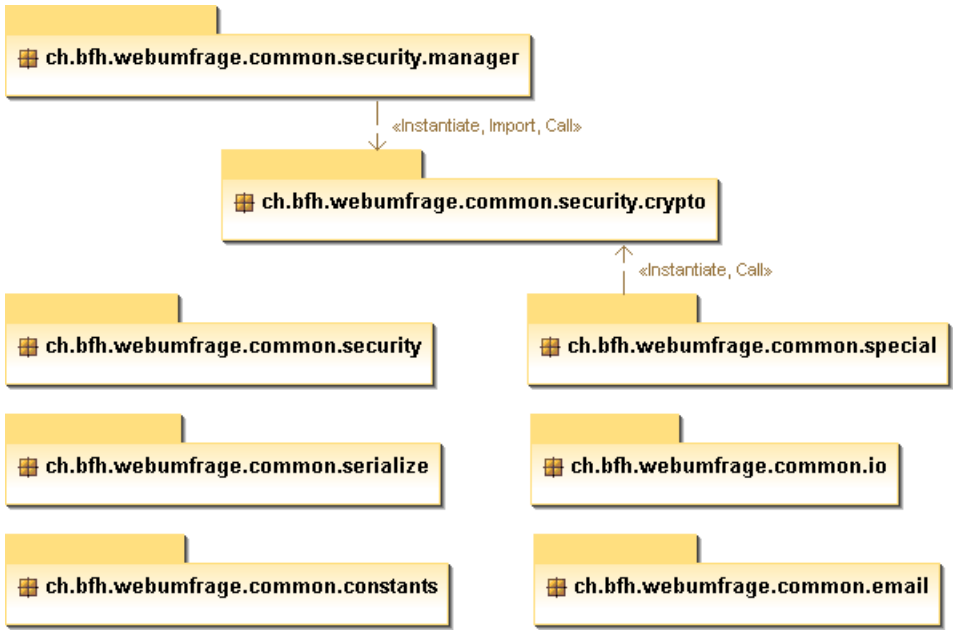


Abbildung 6: Packagediagramm ch.bfh.webumfrage.common

Die einzige Klasse, die komplexere Abhängigkeiten besitzt, ist ch.bfh.webumfrage.common.special.PollerZip, die vom Customer dazu verwendet wird, das Zip-File für den Poller zu schnüren und vom Poller, um die Daten aus dem Zip-File zu lesen.

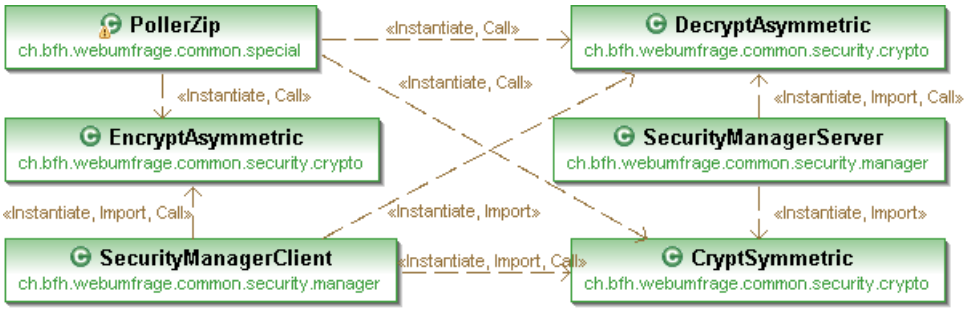


Abbildung 7: Klassendiagramm ch.bfh.webumfrage.common.PollerZip

- **ch.bfh.webumfrage.communication**

  Dieses Projekt wird von den Clients verwendet, um eine Web Services Anfrage abzusetzen und von den Servern, um die Web Service Anfrage anzunehmen und die Antwort zurück zuschicken.

  Die Klasse `CommunicatorClient` wird von allen Clients verwendet, um den Web Service zu starten. Dazu muss der Name des Agents und dessen URL angegeben werden. Auf der Serverseite wird die Anfrage von der Klasse `CommunicatorServer` abgefangen. Anhand des Namens des Agents wird nun ein Objekt des Interfaces `MessageProcessor` per Klassloader erzeugt. Jeder Agent besitzt eine Klasse, die `MessageProcessor` implementiert.

```
mp = (MessageProcessor) getClass().getClassLoader().loadClass(
"ch.bfh.webumfrage." + message.getAgent() +
".communication.MessageProcessorImpl").newInstance();
```

  Die Klasse `MessageProcessorImpl` des Agents weiss nun, was zu machen ist und startet die Verarbeitung.

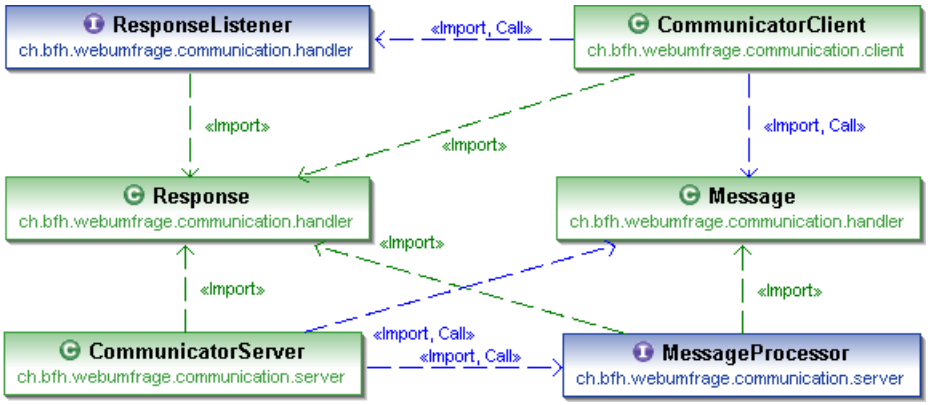


Abbildung 8: Klassendiagramm ch.bfh.webumfrage.communication

- **Umfrageeditor-Projekte**

  Diese Projekte dienen als Grundlage für die Java Applikationen Customerapplikation und Pollerapplikation. Da diese Projekte nichts mit dem Sicherheitskonzept an sich zu tun haben, werden sie nicht detailliert Beschrieben.

  - **ch.bfh.webumfrage.poll.base**

    Dieses Packet beinhaltet die Logik einer Umfrage (Poll), das Parsen und die Darstellung der Umfrage inklusive den darin enthaltenen Elementen. Die Logik wurde so aufgebaut das die einzelnen Elemente Interfaces für die Aktionen bereit stellen. Die Implementierten Klassen befinden sich in den Projekten ch.bfh.webumfrage.poll.editor und ch.bfh.webumfrage.poll.viewer.

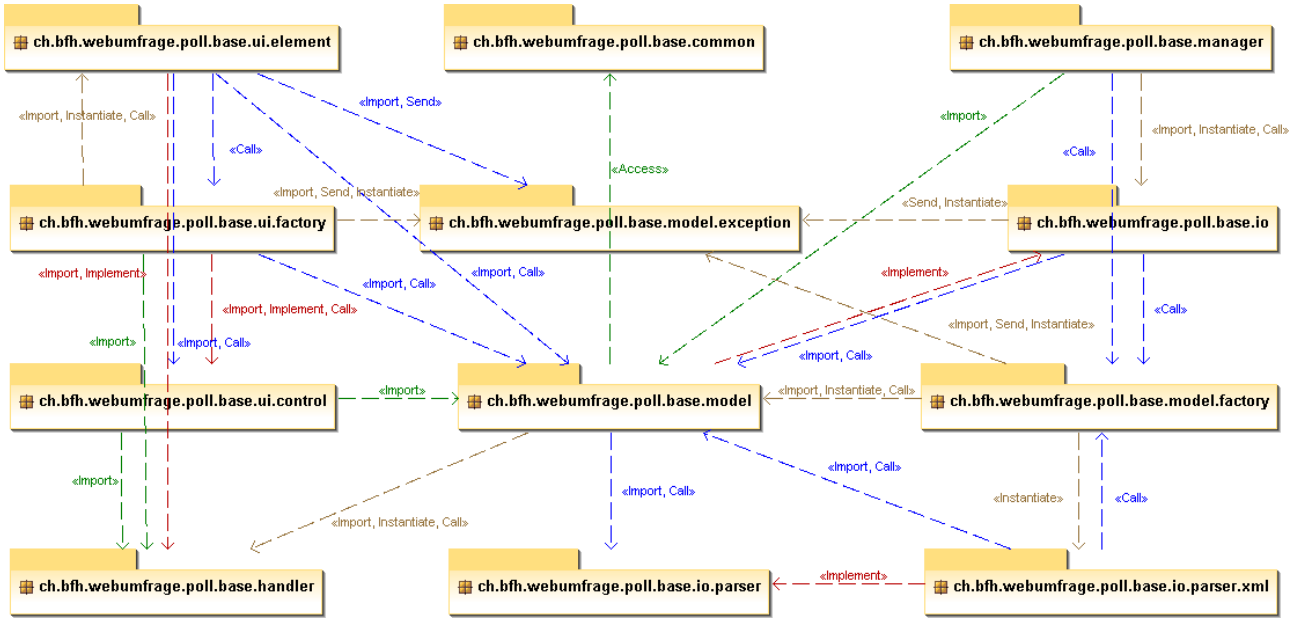


Abbildung 9: Packagediagramm ch.bfh.webumfrage.poll.base

- **ch.bfh.webumfrage.poll.editor**

  Diese Packet stellt dem Customer eine Grafische Oberfläche, zum editieren einer Umfrage, zur Verfügung.

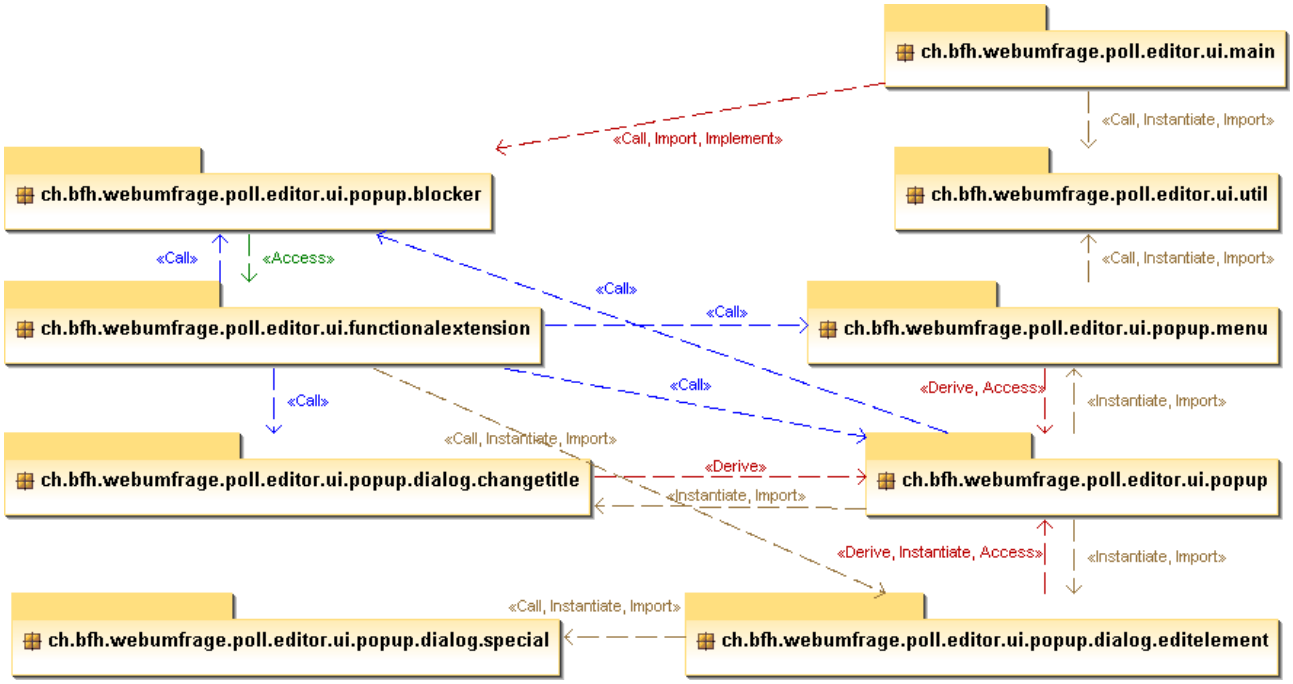


Abbildung 10: Packagediagramm ch.bfh.webumfrage.poll.editor

- **ch.bfh.webumfrage.poll.viewer**

  Diese Packet stellt dem Poller eine Grafische Oberfläche, zum Beantworten einer Umfrage, zur Verfügung.

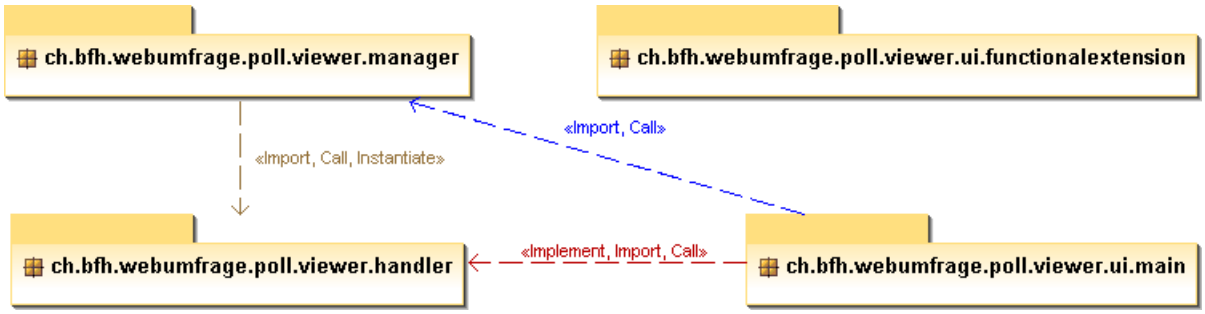


Abbildung 11: Packagediagramm ch.bfh.webumfrage.poll.viewer

- **Java Applikationen**

  Dem Customer und den Pollern wird je eine Java Applikation zur Verfügung gestellt, mit der eine Umfrage erstellt und ausgefüllt werden kann.

  - **ch.bfh.webumfrage.customer**

    Die meisten Klassen in diesem Projekt werden verwendet, um dem Customer die verschiedenen Schritte zur Erstellung einer On-Line Meinungsumfrage im GUI darzustellen. Diese werden hier nicht beschrieben. Mit dem Package ch.bfh.webumfrage.customer.communication werden die Web Service Abfragen abgesetzt. In der Klasse ch.bfh.webumfrage.customer.model.PollManager wird das E-Mail an die Poller verschickt.

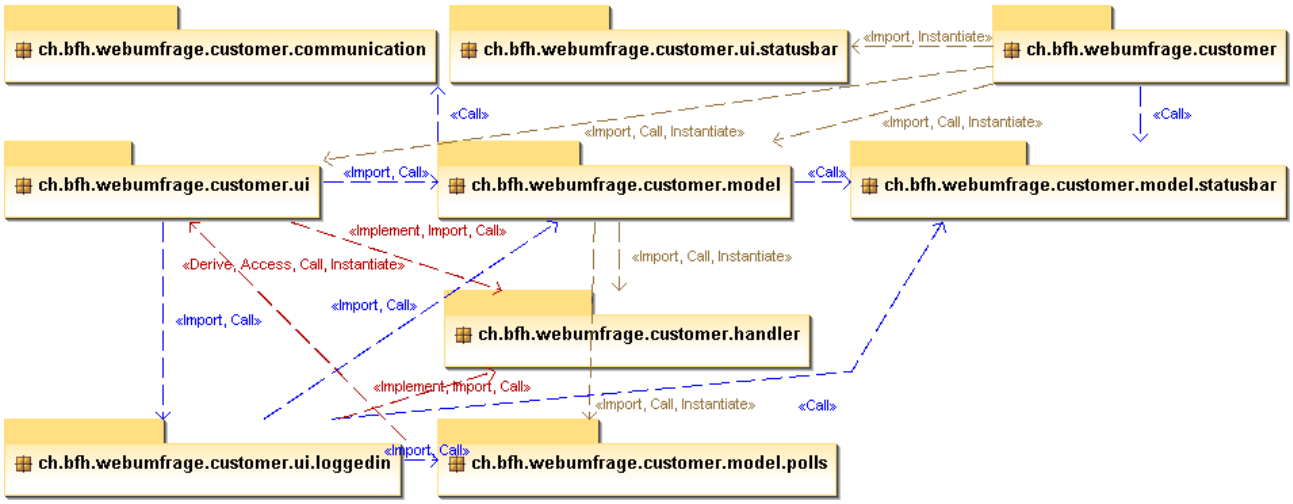


Abbildung 12: Packagediagramm ch.bfh.webumfrage.customer

  - **ch.bfh.webumfrage.poller**

    Die meisten Klassen in diesem Projekt werden verwendet, um dem Poller die verschiedenen Schritte zum Ausfüllen einer On-Line Meinungsumfrage im GUI darzustellen. Diese werden hier nicht beschrieben. Mit dem Package ch.bfh.webumfrage.poller.communication werden die Web Service Abfragen abgesetzt.

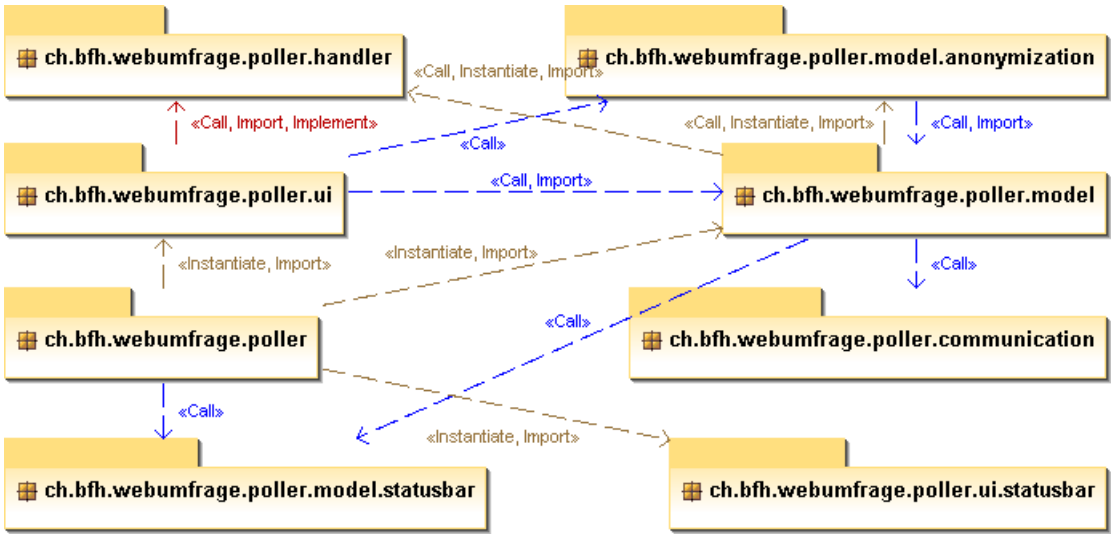


Abbildung 13: Packagediagramm ch.bfh.webumfrage.poller

Interessant ist das Package ch.bfh.webumfrage.poller.model.anonymization, wo die Pollermark generiert und versalzen wird. PollerMark bedient sich dabei der Klassen RandomNumber, um die Zufallszahl zu generieren die Teil der Ausgangszahl ist, Permutation, die aus der Ausgangszahl die Pollermark erstellt, und Salt, welche den Hash der Pollermark versalzt und die Signatur der Pollermark wieder entsatzt

Abbildung 14: Klassendiagramm Anonymisation

- **Web Services**

  Die Poller- und Customerapplikation verwenden Web Services, um mit den Agents zu kommunizieren.

  - **ch.bfh.webumfrage.provider**

    Der Provider stellt einen Web Service zur Verfügung, mit dem der Customer seine On-Line Meinungsumfragen verwalten, erstellen, mutieren und löschen kann.
    Alle Packages in diesem Projekt beginnen mit ch.bfh.webumfrage.provider, wegen des langen Namens werden die Klassen- und Package-Bezeichnungen nur mit provider.xyz bezeichnet. Die Klasse provider.MessageProcessorImpl nimmt die Web Service Anfragen an und verarbeitet sie gleich. security.SecurityManager fertigt die Anfragen für Zertifikate ab. provider.jcr.UtilJCR stellt eine Verbindung mit dem JCR her. Der Provider besitzt auch noch eine Scheduler, der minütlich den Delayer anfrägt, ob er Antworten zu einer On-Line Meinungsumfrage hat, deren Endtermin abgelaufen ist. Die Klasse provider.scheduler.FinishedPollerTrace setzte die Anfrage ab und verarbeitet und speichert die erhaltenen Antworten im JCR des Providers. Die Klasse provider.initialize.Initialize wird

beim Starten und Beenden vom Servlet-Container Tomcat aufgerufen und startet bzw. zerstört den einen Thread der Klasse FinishedPollerTrace. Damit Tomcat weiss, das er den Thread starten soll, muss

```
<listener><listener-class>
    ch.bfh.webumfrage.provider.initialize.Initialize
</listener-class></listener>
```

in das File [Tomcat folder]/webapps/axis/WEB-INF/web.xml eingefügt werden.

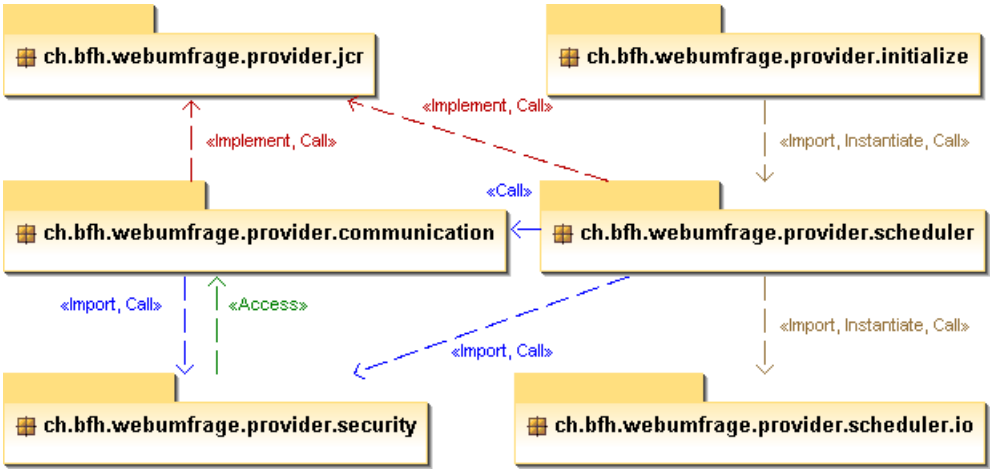


Abbildung 15: Packagediagramm ch.bfh.webumfrage.provider

- **ch.bfh.webumfrage.certifyingauthority**

  Enthält den WebService, der den Hashwert der Pollermark blind signiert. Zudem werden die Identität des Pollers und gewisse Daten abgespeichert, welche nach Beendigung der Umfrage publiziert werden.

  Das Projekt ch.bfh.webumfrage.certifyingauthority enthält nur sechs Klassen. Zwei davon sind Exeptions. InvalidIdException wird geworfen, wenn die Poller-ID nicht mit dem Distinguished Name des mitgeschickten Pollerzertifikats übereinstimmt oder wenn diese Poller-ID schon eine Pollermark signieren liess. VerifyException wird geworfen, wenn die Signatur des Pollerzertifikats oder die Signatur der versalzenen Pollermark nicht verifiziert werden kann. Im ersten Fall ist der Poller nicht authentisiert im zweiten Fall kann die Pollermark nicht signiert werden, da sie nicht korrekt ist. UtilJCR stellt die Verbindung zum JCR her. MessageProcessorImpl nimmt die Web Services auf und initiiert die weitere Verarbeitung. Der SecurityManager stellt den Schlüssel zum blinden signieren der Pollermark bereit. Die Klasse SignPollerMarkBlind enthält die eigentliche Logik und signiert die Pollermark.

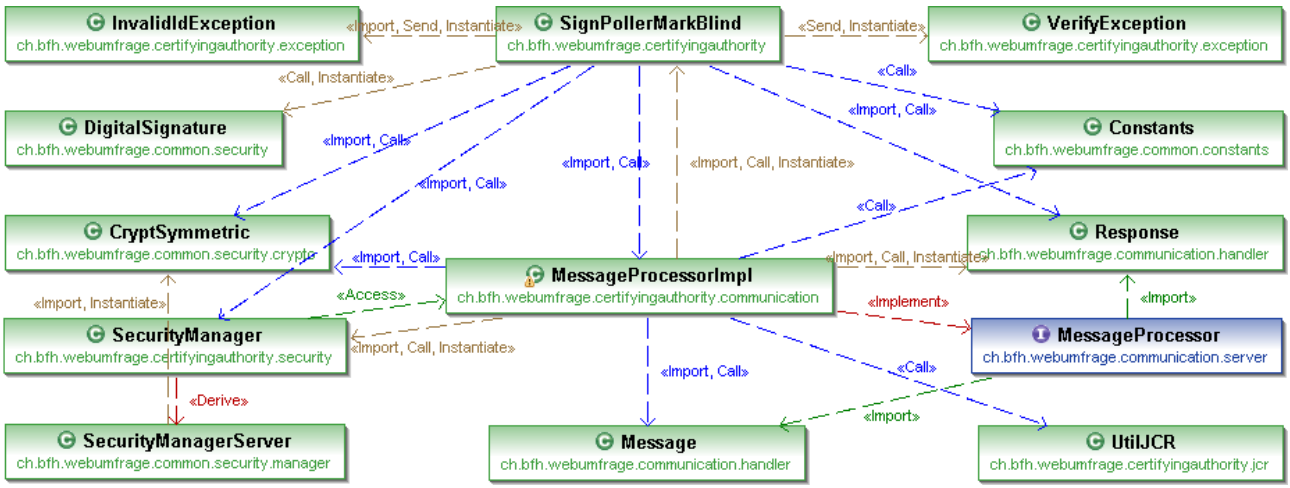


Abbildung 16: Klassendiagramm ch.bfh.webumfrage.certifyingauthority

- **ch.bfh.webumfrage.delayer**

  Die erste Aufgabe des Delayer ist, die empfangenen Antworten der Poller anzunehmen und nach Poll-IDs zu sortieren. Die zweite Aufgabe ist es, auf Verlangen des Providers alle Antworten einer bestimmten Poll-ID abzuliefern. Diese beiden Aufgeben erfüllt die Klasse MessageProcessorImpl. Die zweite Klasse im Projekt UtilJCR stellt die Verbindung zum JCR her.

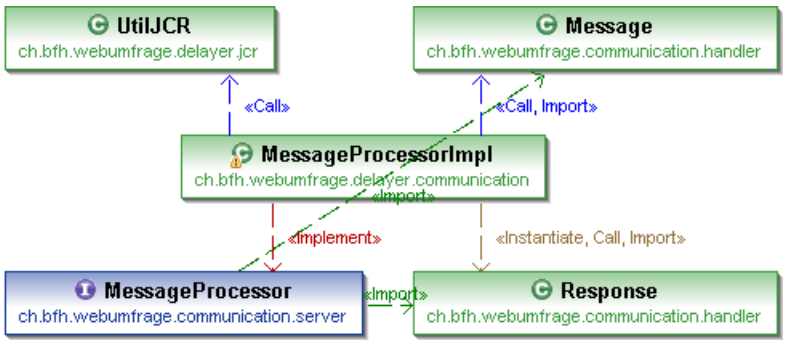


Abbildung 17: Klassendiagramm ch.bfh.webumfrage.delayer

- **Websites**

  Eine Website wird ganz am Anfang zum Registrieren eines Customers und ganz am Schluss zum Publizieren der Umfrageergebnisse verwendet.

  - **ch.bfh.webumfrage.registration**

    Die Registrierung eines Customers besteht aus zwei Seiten. Die erste Seite registration.jsp ist die Eingabemaske, wo der Customer seine Personalien angeben muss. Sind alle Eingaben plausibel (der Username und die E-Mail-Adresse müssen eindeutig sein) wird die Registrierung abgeschlossen, an die eingegebene E-Mail-Adresse eine E-Mail geschickt und auf die Seite successful.jsp navigiert.

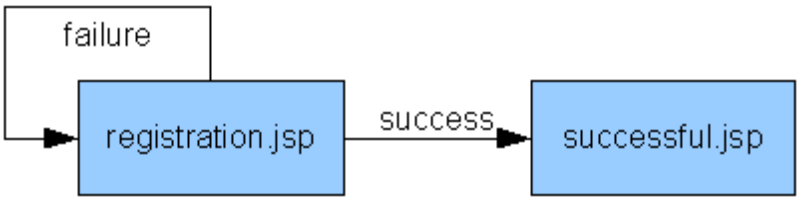


Abbildung 18: Navigation Registrierung

Die Klasse RegistrationHandler leitet die Registrierung und macht die Prüfungen. UtilJCR stellte die Verbindung zum JCR her und mit EmailWithAttachment wird die Bestätigungs-E-Mail versendet.

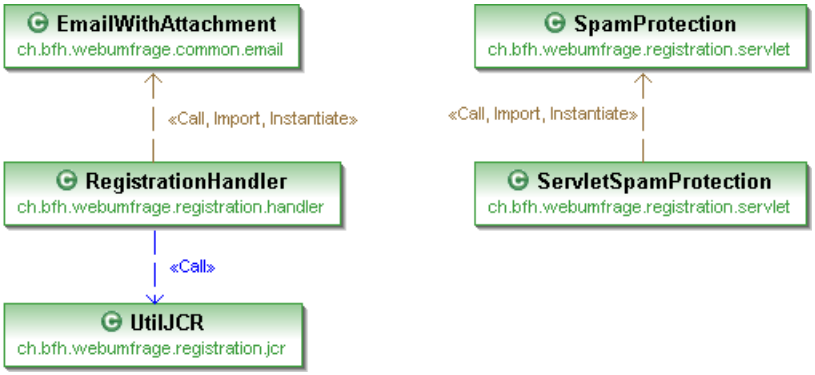


Abbildung 19: Klassendiagramm ch.bfh.webumfrage.registration

SpamProtection wird verwendet damit eine Applikation nicht Unmengen von Accounts eröffnen kann. Der Customer muss den Code in ein Feld eintragen.

Abbildung 20: Spam Protection

- **ch.bfh.webumfrage.publication**

  Dieses Projekt enthält alle Exceptions für die Publikationen, die am Ende der Umfrage veröffentlich werden. Dies sind alles Unterklassen der `PollException`.

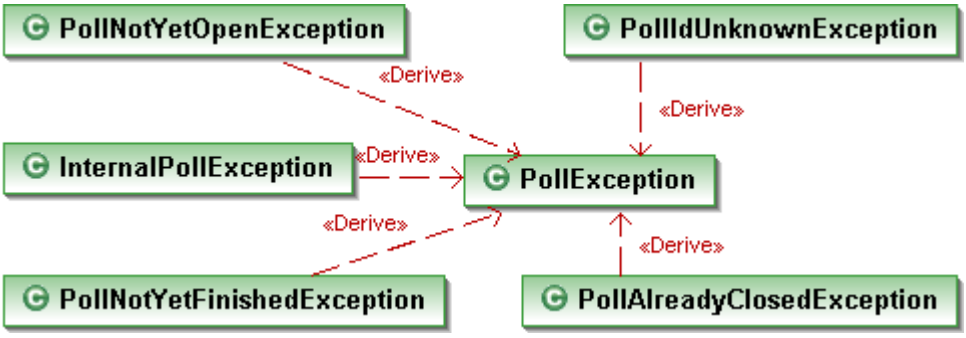


Abbildung 21: Klassendiagramm ch.bfh.webumfrage.publication

- **ch.bfh.webumfrage.publication.ca**

  Am Schluss der On-Line Meinungsumfrage veröffentlich die Certifying Authority die Daten der Poller mit deren Identifikationszeichen auf einer Website. Zum einloggen braucht der Poller die Poll-ID. Mit einer gültigen Poll-ID kommt er auf die Seite listingPollers.jsp, auf dieser Seite sind die Poller-ID (E-Mail-Adresse des Pollers) derjenigen Poller, die eine Pollermark signieren liessen. Durch auswählen einer Poller-ID werden auf der Seite pollerInfos.jsp die Informationen zum Pollen angezeigt.

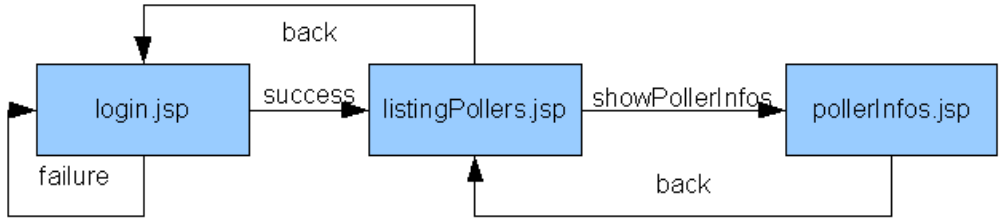


Abbildung 22: Navigation Certifying Authority

Alle Packages in diesem Projekt beginnen mit ch.bfh.webumfrage.publication.ca, wegen des langen Namens werden die Klassen- und Package-Bezeichnungen nur mit ca.xyz bezeichnet. Die Klasse PollerInfoHandler im Package ca.handler enthält alle Informationen der Poller und wickelt die Anfragen ab. Ein Poller wird durch die Klasse ca.Poller dargestellt. Das Package ca.jcr ist dafür zuständig, dass die Daten aus dem JCR gelesen werden und dem PollerInfoHandler zur Verfügung gestellt werden. Für das einloggen der Poller ist die Klasse ca.manager.PollerManager zuständig. Damit die Webseite schnell mehrsprachig gemacht werden kann, haben wir das System Internationalisation (I18n) verwendet. So braucht man den Quellcode nicht zu ändern, wenn es in eine andere Sprache übersetzt werden soll. Die Klasse ca.util.Util kann verwendet werden, wenn in einer Klasse ein Text in der gewünschten Sprache erscheinen soll. Im Code steht Util.getBundleText("Sprache") und im File Resources_xx.properties (xx steht für die Sprache) zum Beispiel "Sprache = Deutsch".

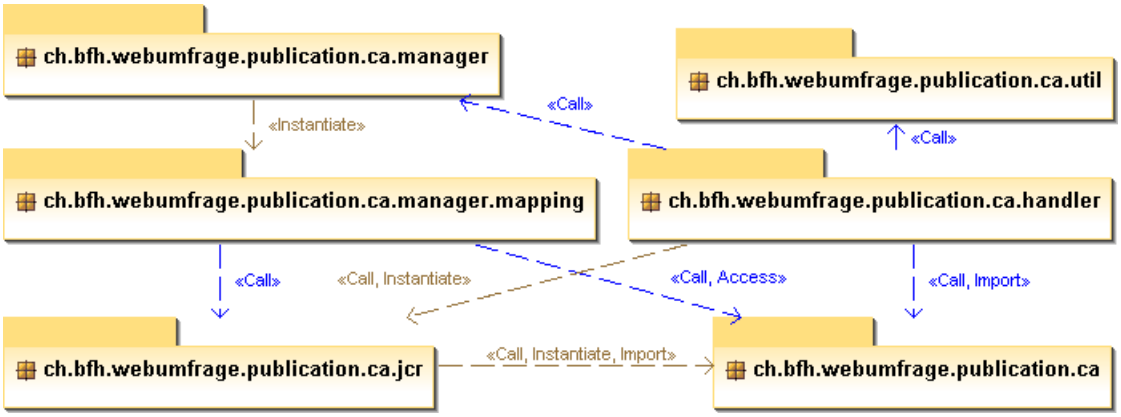


Abbildung 23: Packagediagramm ch.bfh.webumfrage.publication.ca

- **ch.bfh.webumfrage.publication.provider**

  Ebenfalls am Schluss der On-Line Meinungsumfrage veröffentlicht der Provider zu jeder

Pollermark die Signatur der dazugehörenden Antworten. sowie die Gesamtstatistiken der On-Line Meinungsumfrage. Das einloggen funktioniert wie bei der CA mit der Poll-ID. Auf der Seite availablePublications.jsp hat der Poller die Möglichkeit zu Prüfen, ob seine Antworten korrekt gespeichert wurden (listingSigns.jsp), oder die Antworten anzusehen (listingPolls.jsp). Die Seite listingSigns.jsp konnte noch nicht implementiert werden.In listingPolls.jsp sind alle Pollermark sowie das Gesamtergebnis aufgelistet. Durch klicken auf den Link werden die Antworten der Pollermark oder des Gesamtergebnis angezeigt.

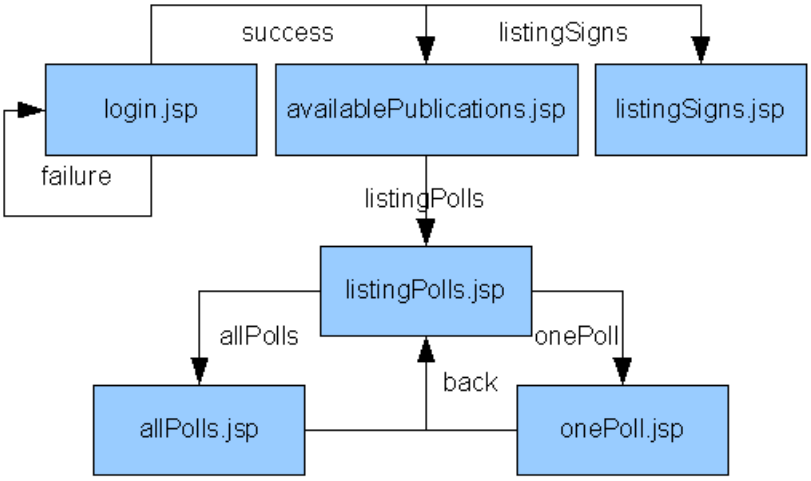


Abbildung 24: Navigation Provider

Alle Packages in diesem Projekt beginnen mit ch.bfh.webumfrage.publication.provider, wegen des langen Namens werden die Klassen- und Package-Bezeichnungen nur mit provider.xyz bezeichnet. Die Klasse PollerMarkHandler im Package provider.handler enthält alle Informationen der Poller und wickelt die Anfragen ab. Das Package provider.jcr ist dafür zuständig, eine Verbindung mit dem JCR aufzubauen. Damit die Webseite schnell mehrsprachig gemacht werden kann, haben wir das System Internationalisation (I18n) verwendet. So braucht man den Quellcode nicht zu ändern, wenn es in eine andere Sprache übersetzt werden soll. Die Klasse ca.util.Util kann verwendet werden, wenn in einer Klasse ein Text in der gewünschten Sprache erscheinen soll. Im Code steht Util.getBundleText(“Sprache“) und im File Resources_xx.properties (xx steht für die Sprache) zum Beispiel “Sprache = Deutsch“.

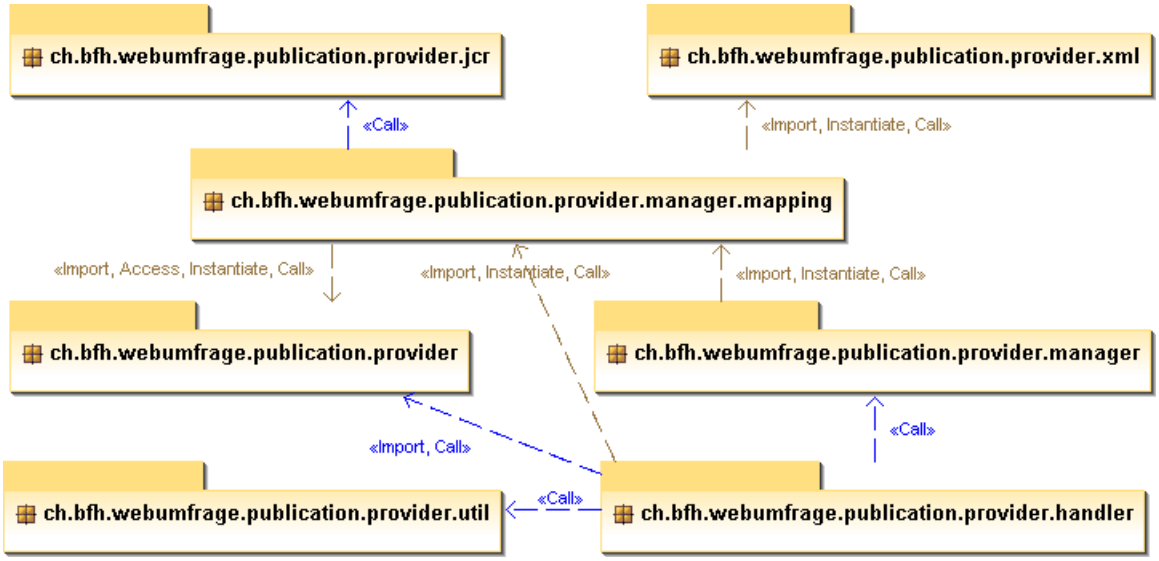


Abbildung 25: Packagediagramm ch.bfh.webumfrage.publication.provider

Mit der Klasse JCRPollerManagerMapping werden die Daten aus dem JCR gelesen und dem PollerMarkHandler zur Verfügung gestellt. Die Texte zu den Fragen und Antworten müssen aus der original On-Line Meinungsumfrage, die der Customer erstellt hat gelesen werden. Die Daten sind im XML Format gespeichert und werden mit der Klasse DOMTree gelesen. Eine Poll enthält alle Fragen und Antworten eines Pollers. Eine Question enthält den Titel der Frage und die SubQuestions, diese wiederum enthält die Frage – wenn vorhanden – und die Antworten. Die Antworten haben einen Counter für das Gesamtergebnis. Für die einzelnen Pollerergebnisse, kann nur eine Antwort den Wert 1 haben. Die anderen Antworten haben den Wert 0. Für das einloggen der Poller sind die Klassen LoginHandler und PollerManager zuständig.

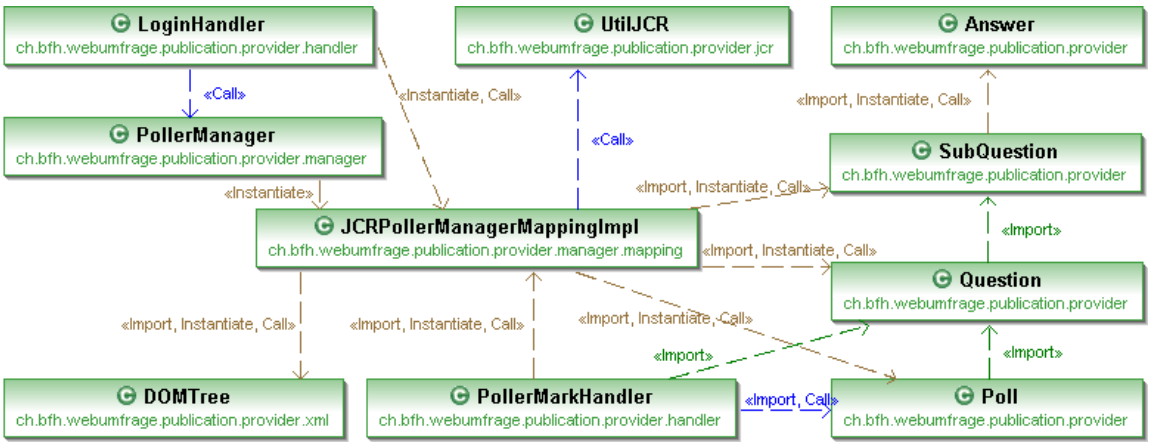


Abbildung 26: Klassendiagramm ch.bfh.webumfrage.publication.provider

9 Datenhaltung

9.1 JCR

Die Daten werden, wie schon erwähnt, mit JCR gespeichert. Das Konzept von JCR ist, dass die Daten hierarchisch als Baum dargestellt werden. Der Aufbau von JCR ähnelt sehr derjenigen von XML-Files. So kann jeder Knoten weiter Knoten oder Eigenschaften enthalten, wie ein XML-Element weitere Elemente oder Attribute enthalten kann. Die Syntax von XPath (bekannt um Abfragen in einem XML-File zu machen) kann auch verwendet werden um Daten im JCR zu suchen. Jeder Wurzelknoten heisst root.

9.1.1 *Provider*

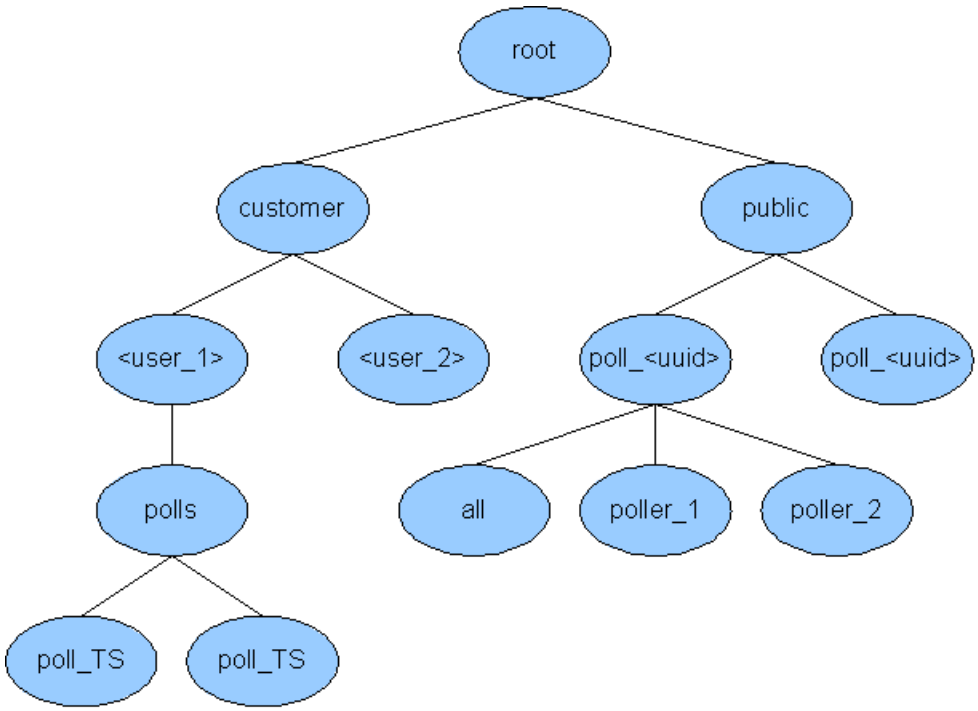


Abbildung 27: JCR Schema Provider

Der Provider muss zwei Arten von Daten speichern. Für jeden Customer werden die On-Line Meinungsumfragen gespeichert und nach Ablauf des Endtermines werden die Antworten der Poller abgespeichert. Dafür werden die Knoten customer und public verwendet. Jeder Customer hat unter dem Knoten customer einen eigenen Knoten mit seinem Usernamen, der die Personalien des Customer enthält (das Passwort ist als Hash gespeichert):

```
Name=aebya
 - email=aebya@bfh.ch
 - firstname=Adrian
 - street=Rüttistrasse 30
 - zip=1716
 - city=Plaffeien
 - lastname=Aeby
```

```
 - pwd=▼:G??&C???(ZQy??↔?>?☼???h?♬?u§*M→►x:?p)????J}?G↕?∟??↓RU?M ?
```

Unter einem weiteren Knoten polls, werden nun alle On-Line Meinungsumfragen eines Customers gespeichert. Der Name des Knoten ist poll_ gefolgt vom Timestamp. In diesem Knoten werden alle Informationen zur On-Line Meinungsumfrage als Eigenschaften gespeichert (jcr:uuid ist die Poll-ID):

```
Name=poll_1194517948000
 - end_date=08.11.2007 12:40
 - data=<?xml version="1.0"><poll>...</poll>
 - end_date_timestamp=1194522000000
 - description=Beschreibung der Testumfrage
 - start_date=08.11.2007 11:34
 - status=1
 - jcr:uuid=27ba1d01-7ac9-4a4a-b808-5b380dc554ac
 - title=Testumfrage
```

Unter public werden die Umfrageergebnisse aller Poller sortiert nach Poll-ID gespeichert. Der erste Knoten all enthält das Gesamtergebnis. Dieser Knoten enthält mehrere Unterknoten für die Fragen und Antworten ():

```
Name=all
 - end_date=08.11.2007 12:40
 - id=27ba1d01-7ac9-4a4a-b808-5b380dc554ac
 - description=Beschreibung der Testumfrage
 - title=Testumfrage
        Name=question_1
         - index=1
         - title=Titel der Frage
                Name=question1194517948037
                 - text=Ist das Ihre erste Umfrage?
                       Name=radiobutton1194517948038
                        - count=4
                        - text=Label
                       Name=radiobutton9845764632352
                        - count=5
                        - text=Label
```

radiobutton1194517948038 hat in diesem Fall den Wert Ja und radiobutton9845764632352 den Wert Nein. Das heisst, das Umfrageergebnis für die Frage Ist das Ihre erste Umfrage? lautet Ja: 4, Nein: 5.

9.1.2 *Certifying Authority*

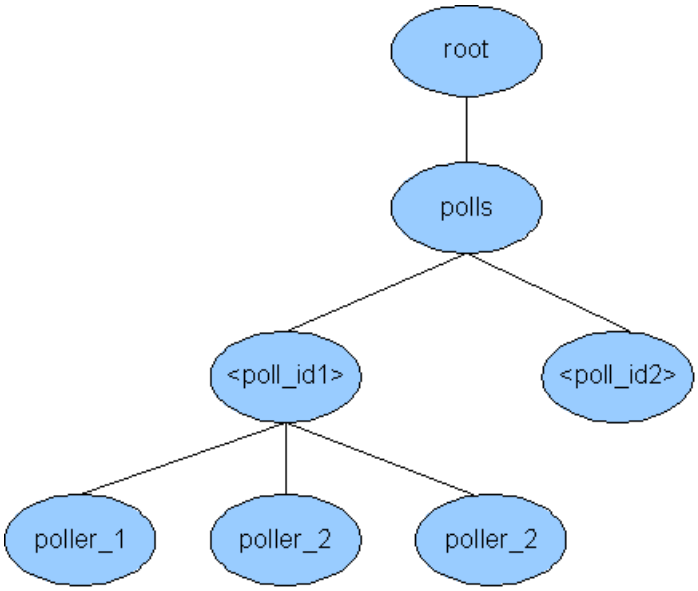


Abbildung 28: JCR Schema Certifying Authority

Die Certifying Authority speicher für jede Poll-ID das Zertifikat des Customers, mit welchem die Signatur des Pollerzertifikats verifiziert wird. Unter diesem Knoten speichert die CA für jeden Poller, der seine Pollermark signieren liess, gewisse Daten:

```
Name=poller_1194518781625
 - poller_id=aebya@bfh.ch
 - signed_by_poller_salted_hashed_poller_mark=424872...
 - salted_hashed_poller_mark=206123098479982576636672...
 - signed_by_ca_salted_hashed_poller_mark=2109915747...
```

9.1.3 *Delayer*

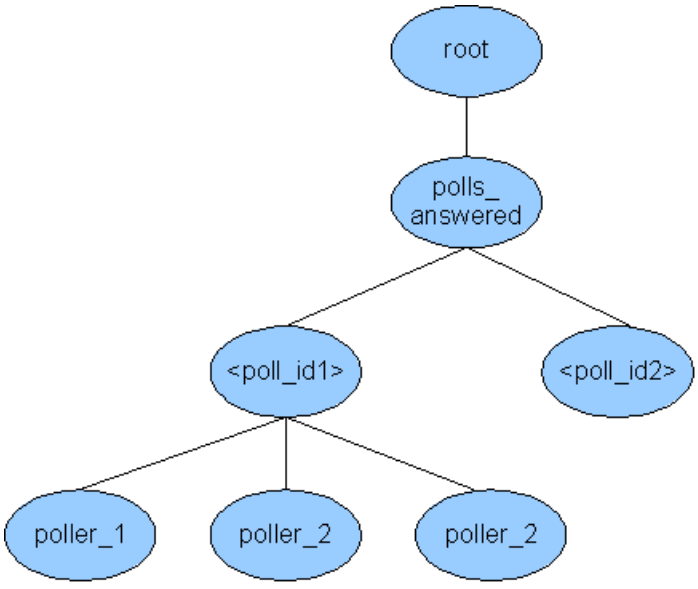


Abbildung 29: JCR Schema Delayer

Der Delayer erhält die Antworten von den Pollern. Anhand der Poll-ID werden diese abgespeichert (die Daten werden verschlüsselt verschickt und können nur vom Provider entschlüsselt werden):

```
Name=poller_1194518081234
 - timestamp=1194518081234
 - hashed_poll=LyuX3CU0wV5lrwVMECDBTEwyo/YQqMcIsxBEqNFlvNQ=
 - pollerMark=N2wHVJb2leEH6E/2GLB7MrNYExag0EekCRZ+7oU7w6TaawWY6...
 - key=e3TNpyhnL9YG3Ea1U2dFQxVznNUimr1ayzhpAZEDbSoCk8BL4EvZqOxE...
 - hashedBlindSignedPollerMark=TsOfzi+dTvA+1zLo+jzBy0wyo/YQqMcI...
 - poll=a.6@z?↔???<??L!o???v[??Q?"?+?Ij??z?K ??I?K↕???§C?—r?Aj?...
 - blindSignedPollerMark=059843FX8P5RvlEQYDb4TdTmkR0G9nzpH6KUXo...
```

9.2 XML

Die Fragen die der Poller vom Provider erhält und die Antworten die er an den Delayer verschickt sind im XML Format gespeichert.

9.2.1 *Fragen*

Im Zipfile, das der Poller erhält befindet sich ein XML-File mit de Fragen. Die Pollerapplikation liest dieses XML-File und stellt so die Fragen dar. Der Aufbau sieht folgendermassen aus:

```
<!DOCTYPE poll [
  <!ELEMENT poll (description, startdate, enddate, questions)>
  <!ATTLIST poll id ID #REQUIRED
                 title CDATA #REQUIRED>
  <!ELEMENT description (#PCDATA)>
  <!ELEMENT startdate (#PCDATA)>
  <!ELEMENT enddate (#PCDATA)>
  <!ELEMENT questions (question)*>
  <!ELEMENT question (elements)*>
  <!ATTLIST question id ID #REQUIRED
                    index CDATA #REQUIRED
                    title CDATA #REQUIRED>
  <!ELEMENT elements (element)*>
  <!ELEMENT element (position, special)>
  <!ATTLIST element id ID #REQUIRED
                    type CDATA #REQUIRED
                    group CDATA #IMPLIED
                    key CDATA #IMPLIED
                    value CDATA #IMPLIED>
  <!ELEMENT position EMPTY>
  <!ATTLIST position x CDATA #REQUIRED
                     y CDATA #REQUIRED>
  <!ELEMENT special (set)>
  <!ELEMENT set EMPTY>
  <!ATTLIST set name CDATA #REQUIRED
                value CDATA #REQUIRED>
]>
```

Das Tag <poll /> ist das Wurzelelement und enthält als Attribute die Poll-ID und den Titel der Umfrage.

Danach erscheinen die selbsterklärenden Tags <description />, <startdate /> und <enddate />. Das Tag <questions /> enthält nun eine beliebige Anzahl von <question /> Elementen. Jedes <question /> Tag enthält die Attribute id (Wert: question und eine Zufallszahl), index (aufsteigend nummeriert, beginnend bei 1) und title (Titel der Frage). Für jede Unterfrage in einer Frage enthält <question /> ein <elements />, welches wiederum für deses Label ein <element /> enthält. <element /> hat wieder ein id und ein type Attribut die obligatorisch sind. Ist der type = "label", entfallen die übrigen Attribute. Ist der type = "radiobutton", müssen noch die Attribute group, key und value gesetzt werden. <position /> wird zur Darstellung verwendet und <special /> enthält den Text des Labels.

Dies kann ein folgendes Fragen-XML-Dokument ergeben:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<poll id="ede44271-30ae-4ea8-86aa-a8dd4fc18f09" title="Testumfrage">
    <description>Beschreibung der Testumfrage</description>
    <startdate/>
    <enddate>08.11.2007 12:00</enddate>
    <questions>
        <question id="question1194553111818" index="1" title="Titel der Frage">
            <elements>
                <element id="label1194553111819" type="label">
                    <position x="100" y="100"/>
                    <special>
                        <set name="text" value="Ist das Ihre erste Umfrage?"/>
                    </special>
                </element>
                <element group="group" id="radiobutton1194553111820"
                          key="label1194553111819"
                    type="radiobutton" value="radiobutton1194553111820">
                    <position x="100" y="101"/>
                    <special>
                        <set name="text" value="Ja"/>
                    </special>
                </element>
                <element group="group" id="radiobutton1194553111821"
                          key="label1194553111819"
                    type="radiobutton" value="radiobutton1194553111821">
                    <position x="101" y="101"/>
                    <special>
                        <set name="text" value="Nein"/>
                    </special>
                </element>
            </elements>
        </question>
    </questions>
</poll>
```

9.2.2 *Antworten*

Hat der Poller die Fragen beantwortet, schickt er die Antworten an den Delayer. Diese Daten werden im XML-Format versendet. Der Aufbau sieht folgendermassen aus:

```
<!DOCTYPE poll [
  <!ELEMENT poll ( questions)>
  <!ATTLIST poll id ID #REQUIRED
                 title CDATA #REQUIRED>
  <!ELEMENT questions (question)*>
  <!ELEMENT question (answer)*>
  <!ATTLIST question index CDATA #REQUIRED>
  <!ELEMENT answer EMPTY>
  <!ATTLIST answer key CDATA #REQUIRED
                   value CDATA #REQUIRED>
]>
```

Das Antwort-XML ist etwas einfacher. Das Wurzelelement <poll /> enthält nur ein <questions /> Element. Die Attribute sind gleich wie im Fragen-XML. <questions /> enthält beliebig viele <question /> Elemente, welches nur ein Attribut index hat. Mit diesem index kann man den Title der Frage im Fragen-XML suchen. Das selbe gilt für das Element <answer />, bei welchem die Attribute key und value auf die Labels bzw. Radiobuttons im Fragen-XML verweisen.

Dies kann ein folgendes Antwort-XML-Dokument ergeben:

```
<?xml version="1.0" encoding="UTF-8"?>
<poll id="ede44271-30ae-4ea8-86aa-a8dd4fc18f09" title="Testumfrage">
    <questions>
        <question index="1">
            <answer key="question1194553111818" value="radiobutton1194553111820"/>
        </question>
    </questions>
</poll>
```

10 Wirtschaftlichkeit

Für die Durchführung der Diplomarbeit On-Line Meinungsumfrage wurden Softwaretechnologien verwendet, die kostenlos benutzt werden können. Zum Erstellen des Prototyps wurden die PC der Diplomanden verwendet. Zudem erhielten wir von der BFH-TI drei virtuelle Server um unsere Agents zu installieren.

Es sind keine Kosten für die Diplomarbeit angefallen. Trotzdem wollen wir aufzeigen, wie sich die aufgewendete Zeit auf die verschiedenen Arbeiten aufteilt und welcher Diplomand welchen Beitrag geleistet hat.

Der vorgesehene Zeitaufwand von 300 Stunden pro Studenten, erwies sich als gute Schätzung. Zusammen haben wir ca 615 Stunden für das Projekt aufgewendet. Für die Auswertung der Stunden wurden drei in die Kategorien Sitzungen, Dokumentation und Programmieren unterschieden:

	Adrian Aeby	Martin Wiget	Total
Sitzungen	4%	4%	4%
Dokumentation	38%	12%	25%
Programmieren	58%	84%	71%

Das ganze als Diagramm dargestellt:

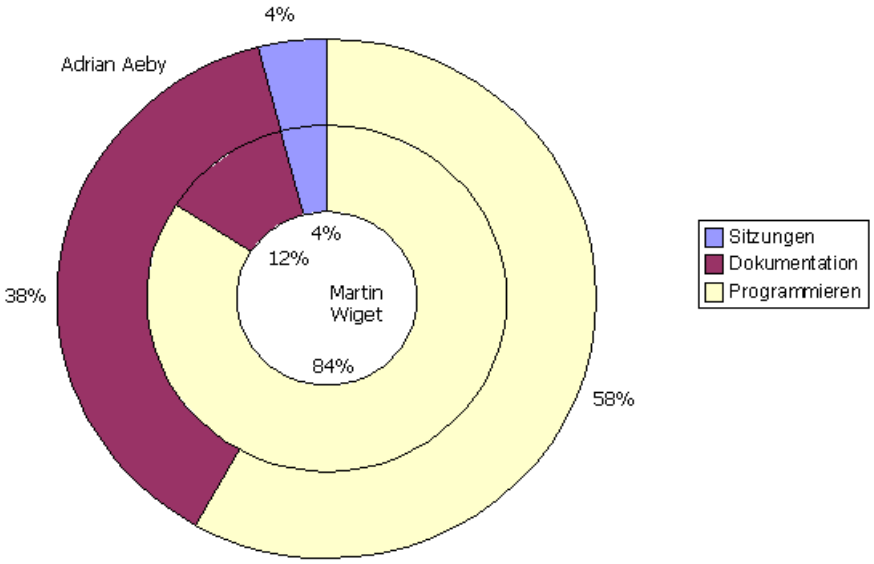


Abbildung 30: Arbeitsaufwand

Fast drei viertel des gesamten Arbeitsaufwands wurde für die Implementierung verwendet. Unter die Dokumentation fallen alle Dokumente die im Verlauf der Diplomarbeit erstellt wurden.

10.1 Aktivitäten

Es ist nicht so einfach, zu sagen wer welche Arbeiten gemacht hat, da viele Sachen gemeinsam erledigt wurden.

10.1.1 *Adrian Aeby*

10.1.1.1 *Dokumentation*

Wie aus dem Diagramm Arbeitsaufwand hervorgeht hat Adrian Aeby den grössten Teil der Dokumentation geschrieben. Dies sind Hauptsächlich die Dokumente, die zu Beginn der Diplomarbeit erstellt wurden und dieses Dokument.

10.1.1.2 *Programmieren*

Adrian Aeby hat:

- Viele Klassen im common Package kodiert. So für das Versenden der E-Mail, Ver- und Entschlüsseln von Daten, Hashen von Daten.
- Die ersten Versuche für Web Sevices.
- Das blinde Signieren der Pollermark (Poller- und CA-seitig).
- Publikationen der Agents.

10.1.2 *Matrin Wiget*

10.1.2.1 *Dokumentation*

Martin Wiget hat zu grossen Teilen die Benutzerdokumentation geschrieben und bei den anderen Dokumenten einige Kapitel geschrieben.

10.1.2.2 *Programmieren*

Martin Wiget hat:

- Das ganze GUI (Poller- und Customerapplikation) entwickelt.
- Die Kommunikation zwischen den Parteien erweitert und fertig gestellt.
- Die ganze Datenhaltung mit JCR definiert und implementiert.
- Die Agents Provider und Delayer implementiert.

11 Konsequenzen

11.1 Aussichten

Mit der Implementierung des Prototyps ist es uns gelungen, den Hauptteil des Protokolls umzusetzen. Wie erwähnt, kann eine On-Line Meinungsumfrage durchgeführt werden, wenn alles wie beschrieben korrekt installiert und befolgt wird. Um nun die On-Line Meinungsumfrage produktiv einsetzen zu können, muss die Praxistauglichkeit getestet werden. Dazu braucht es Erfahrungen von aussenstehenden Benutzern der OLM. Zudem muss weiter analysiert werden, was passiert, wenn verschiedene Parteien versuchen die On-Line Meinungsumfrage zu manipulieren oder zu sabotieren.

11.2 Weiteres Vorgehen

Über das weitere Vorgehen haben wir uns noch keine grossen Gedanken gemacht. Klar ist jedoch, dass zuerst die bekannten Fehler behoben werden müssen. Nach dem ist es wichtig, dass der Ablauf nach dem Sicherheitskonzept korrekt fertig implementiert wird. So muss jeder Agent die beschriebenen Informationen publizieren können und der Poller muss wissen, wie seine eigenen Antworten verifizieren kann. Zudem kommt, dass auf falsches Benutzen der Applikationen richtig reagiert werden muss, dies ist bis jetzt nur rudimentär implementiert. Weiter müssen die oben genannten Pendenzen analysiert und gelöst werden. Dies betrifft vor allem das Entdecken und Melden eines Betruges. Die Funktionalitäten der Customerapplikation steht sicher nicht im Zentrum der Arbeit, aber auch diese kann natürlich beliebig erweitert werden. So kann ein versierter User ohne Problem eine On-Line Meinungsumfrage erstellen, ein neuer User kann aber doch unbewusst einige Fehler begehen.

Dieser Prototyp kann also eine On-Line Meinungsumfrage erstellen und durchführen, aber damit es wirklich produktiv eingesetzt werden kann, muss noch viel Arbeit und Schweiss in das Projekt hinein fliessen.

11.3 Erfahrungen

Schon während der Semesterarbeit wurde uns klar, dass der Sicherheitsaspekt viel komplexer und zeitaufwendiger sein wird, als wir es uns vorgestellt haben. Dies war doch ein Risikofaktor, da wir beide wenig Erfahrungen mit Internetsicherheit hatten. So haben wir erkannt, dass das Erstellen eines Prototyps in der Semesterarbeit nicht realistisch ist und haben dies zum Ziel in der Diplomarbeit gemacht. In der Semesterarbeit haben wir das Sicherheitskonzept erarbeitet. Um so mehr freuten wir uns auf die Diplomarbeit, da wir nun mit der Implementierung beginnen konnten.

Es zeigte sich, dass es richtig war soviel Zeit in das Sicherheitskonzept zu investieren. Es konnte ohne grosse Anpassungen implementiert werden. In der Semesterarbeit hatten wir noch einige Leerläufe – so

hatten wir viel Zeit in die Entwicklung eines Editors auf Webbasis gesteckt, was dann nicht weiter verwendet werden konnte. Während der Diplomarbeit sind solche Leerläufe weitgehend ausgeblieben.

Für uns war es praktisch, dass wir viele Arbeiten unabhängig von anderen machen konnten. So war es auch nicht nötig, dass wir beide alles bis ins Detail kannten. In der Semesterarbeit konnten wir durch viele Diskussionen das Sicherheitskonzept erstellen. In der Diplomarbeit konnten wir die Zeit einzeln nutzen, um die Software zu erstellen. Wir hatten Angst, dass die Applikation für die Poller – wegen den Sicherheitsmassnahmen – unhandlich wird. Diese Angst war aber vergebens, mit der Pollerapplikation ist uns eine einfache, benutzerfreundliche Anwendung gelungen.

Zu Beginn der Endphase hatten wir das Gefühl, dass die Realisierung des Prototyps noch zu wenig weit war und wir in Zeitnot geraten würden. In den darauffolgenden Wochen konnten wir aber viele Probleme lösen und wurden zuversichtlich, einen lauffähigen Prototypen abgeben zu können.

Die Zusammenarbeit mit unter den Studenten war sehr positiv. Wir konnten das ganze Projekt abschliessen ohne wirkliche Auseinandersetzungen gehabt zu haben. Ich habe viel Zeit in die Erstellung der Dokumente investiert, so konnte sich Martin besser auf die Erstellung der Software konzentrieren, auf welche wir doch stolz sein können.

Auch nicht zu unterschätzen ist der Lerneffekt. Endlich konnten wir die im Unterricht gewonnenen theoretischen Kenntnisse in die Praxis umsetzen. Auch wenn wir schlussendlich nicht die Technologien verwendet haben, die wir ursprünglich wollten, so haben wir doch unseren Horizont erweitern können und gezeigt, dass wir flexibel genug sind, von unseren Vorstellungen abzuweichen, wenn dies erforderlich ist.

A Anhang

A Zertifikate erstellen

Um selbst signierte Zertifikate zu erstellen, kann OpenSSL [C.2] verwendet werden. Zudem wird noch eine PKCS#12Datei erstellt. PKCS#12 definiert ein Dateiformat, das dazu benutzt wird, private Schlüssel mit dem zugehörigen Zertifikat passwortgeschützt zu speichern. Dieser Vorgang wird in vier Schritten ausgeführt:

1. Erstellen des privaten Schlüssels

```
openssl genrsa -des3 -out privkey.pem 2048
```

2. Als Nächstes muss eine Zertifikatssignierungsanfrage erzeugt werden, welche die Daten des eigenen Zertifikats enthält

```
openssl req -new -x509 -nodes -sha512 -days 3650 -key privkey.pem > host.cer
```

3. Selbst signieren des Zertifikats

```
openssl x509 -signkey privkey.pem -in host.cer -out hostselfsigned.cer
```

4. Erstellen einer PKCS#12 Datei

```
openssl pkcs12 -inkey privkey.pem -in hostselfsigned.cer -export -out
   host.p12
```

Die Datei `hostselfsigned.cer` ist das eigentliche Zertifikat und wird anderen Benutzern zur Verfügung gestellt. `host.p12` dagegen wird nur vom Inhaber des Zertifikats zum Signieren oder Entschlüsseln von Daten verwendet.

B Generator-Modulus Paar erstellen

Um ein Generator-Modulus Paar zu erstellen, kann OpenSSL [A.4] verwendet werden. Der Modulus m muss „safe prime“ (m = 2p + 1, wobei m und p Primzahlen sind) und 1024 Bit lang sein. Für den Generator g wird normalerweise 2 verwendet, siehe RFC4419 [A.3].

1. Erstellen des Modulus

```
openssl.exe dhparam -text 1024
```

2. Output

```
Diffie-Hellman-Parameters: (1024 bit)
    prime:
        00:f8:5d:ce:4c:ee:d6:d1:b3:f7:96:75:8d:42:ed:
        d1:95:9b:a8:3c:29:29:05:ba:8d:b0:df:a9:18:af:
```

```
    55:92:e3:f6:e5:27:98:98:fd:7b:fd:ab:d0:0b:ff:
    13:4d:f0:37:4d:32:dc:80:50:c6:88:9c:46:f6:fc:
    9d:94:27:cf:f4:ff:3b:11:1f:9b:e7:e2:22:11:05:
    1a:00:d6:40:b2:02:fc:8b:83:52:ca:85:34:d2:3f:
    eb:fe:b2:88:0a:e5:07:df:7b:27:43:b5:6d:13:73:
    28:b8:84:43:39:28:95:fe:0e:51:83:a3:e3:07:42:
    2e:44:0d:ee:21:fa:51:4e:7b
generator: 2 (0x2)
```

B Glossar

Agent Sind die Kommunikatoinspartner einer Applikation, so der Provider, die Certifying Authority, der Delayer oder der Customer (dieser allerdings nur aus Sicht eines Pollers).

Ballot Distributor Ist im Original-Protokoll der Agent, der die Teilnehmer einlädt. In der On-Line Meinungsumfrage übernimmt der Customer die Rolle des Ballot Distributors.

BFH-TI Berner Fachhochschule – Technik und Informatik

Blind Signieren Stellt sicher, dass der Signierer den Inhalt des zu signierenden Wertes nicht kennt, der Auftraggeber aber trotzdem eine gültige Signatur erhält

Certificate Authority Ist eine Organisation, die digitale Zertifikate herausgibt.

Certifying Authority Kurz CA genannt, ermöglicht, dass ein Poller anonym abstimmen kann. Die CA stellt anhand des signierten Pollerzertifikats sicher, dass es sich um einen autorisierten Poller handelt und dieser zum ersten Mal eine Anfrage stellt. Dann signiert die CA die Pollermark des Pollers blind.

Customer Ein Kunde des Providers, der eine Meinungsumfrage erstellen und durchführen will.

Delayer Nimmt die Antworten der Poller entgegen. Dieser zusätzliche Agent stellt sicher, dass durch ein Zusammenarbeiten der CA und des Providers die Verbindung zwischen Pollermark und Polleridentifikation nicht hergestellt werden kann.

Diplomarbeit Abschlussarbeit des Diplomstudiengangs Ingenieur FH in Informatik.

Distinguished Name Eindeutige Bezeichnung des Eigentümers des öffentlichen Schlüssels, in unserer Applikation die E-Mail-Adresse des Inhabers.

GUI Das Graphical User Interface ist eine Softwarekomponente, die einem Computerbenutzer die Interaktion mit der Maschine über grafische Elemente unter Verwendung eines Zeigegerätes (wie einer Maus oder einer Tastatur) erlaubt.

JCR Java Content Repository API zum Speichern und Zugreifen von Daten.

JSF JavaServer Faces Framework zum Erstellen von Webanwendungen.

Öffentlicher Schlüssel In der Kryptologie Schlüssel, die jedem bekannt sein dürfen und z. B. zur Verschlüsselung eines Klartextes in einen Geheimtext genutzt werden können.

On-Line Meinungsumfrage Titel der Diplomarbeit. Als On-Line Meinungsumfrage (OLM) wird auch der Ablauf einer ganzen Umfrage beschrieben.

Partei Jede an der Umfrage beteiligte Person oder Agent, also Customer, Poller, Provider, Certifying Authority oder Delayer.

PKCS#12 Steht für Public Key Cryptography Standards Dokument 12. Definiert ein Dateiformat, das dazu benutzt wird, private Schlüssel mit dem zugehörigen Zertifikat passwortgeschützt zu speichern.

Poller Ein Teilnehmer, der eine Meinungsumfrage ausfüllen soll.

Pollermark Eindeutiges Identifikationskennzeichen für jeden Poller, welches jeder Poller selbst erstellt. Somit wird gewährleistet, dass nur der Poller eine Verbindung zwischen seiner Identität und der Pollermark herstellen kann. Beim Abgeben der Antworten wird die Pollermark anstatt der Poller-ID mitgegeben. So kann der Provider nicht feststellen, welcher Poller welche Antworten abgegeben hat.

Privater Schlüssel In der Kryptologie Schlüssel, die nur ihren legitimen Inhabern bekannt sein dürfen und diese in die Lage versetzen, einen Geheimtext in einen Klartext zu entschlüsseln oder zu einer Nachricht eine digitale Signatur zu erzeugen.

Protokoll Der gesamten Ablauf der On-Line Meinungsumfrage wird als Protokoll bezeichnet. An Anonymous Electronic Voting Protocol for Voting Over The Internet [A.1] dient als Vorlage für unser Protokoll und wird als Original-Protokoll bezeichnet.

Provider Agent, der die meisten Aufgaben an der On-Line Meinungsumfrage erledigt. Der Provider ermöglicht das Registrieren eines Customers, die Erstellung und Verwaltung von Meinungsumfragen und die Einladung der Poller. Nach Ablauf der Umfrage werden die Ergebnisse berechnet und publiziert.

Semesterarbeit Vorarbeit zur Diplomarbeit, die im letzten Semester an der BFH-TI erarbeitet wurde.

Teilnehmer Synonym zu Poller

URL Uniform Resource Locators identifizieren eine Ressource über das verwendete Netzwerkprotokoll (beispielsweise http oder ftp) und den Ort der Ressource in Computernetzwerken. Beispiel http://webumfrage.bfh.ch/

Versalzen Wird in unserem Kontext zum blinden Signieren verwendet. Der Wert, der signiert werden soll, wird zuerst versalzt von der CA signiert und anschliessend wieder entsalzt. Somit kennt die CA den Wert nicht, den sie signiert. Versalzen beschreibt dabei die mathematische Operation, die dies gewährleistet.

Vote Compiler Ist im original Protokoll der Agent, der die Antworten holt und die Gesamtresultate berechnet. In der On-Line Meinungsumfrage übernimmt der Provider die Rolle des Vote Compilers.

Web Service eine Software-Anwendung, die mit einem Uniform Resource Identifier (URI) eindeutig identifizierbar ist und deren Schnittstellen als XML-Artefakte definiert, beschrieben und gefunden werden können. Ein Webservice unterstützt die direkte Interaktion mit anderen Software-Agenten unter Verwendung XML-basierter Nachrichten durch den Austausch über internetbasierte Protokolle.

Zertifikat Strukturierte Daten, die den Eigentümer sowie weitere Eigenschaften eines öffentlichen Schlüssels bestätigen. Durch ein digitales Zertifikat können Nutzer eines asymmetrischen Kryptosystems den öffentlichen Schlüssel einer Identität (z. B. Agent, Customer oder Poller) zuordnen und seinen Geltungsbereich bestimmen. Damit ermöglichen digitale Zertifikate den Schutz der Vertraulichkeit,

Authentizität und Integrität von Daten durch die korrekte Anwendung der öffentlichen Schlüssel.

C Literaturverzeichnis

A Externe Dokumente

[A.1] An Anonymous Electronic Voting Protocol for Voting Over The Internet
http://projects.hti.bfh.ch/webumfrage/thesis/doc/AAEVPfVOTI.pdf

[A.2] Handbuch HERMES: Projekttyp Systementwicklung 2003, 5.3.53 Projekthandbuch
http://ehermes.swissforge.net/books/de/hermesse/index.htm

[A.3] RFC4419: Generator-Modulus Paar http://rfc.sunsite.dk/rfc/rfc4419.html

[A.4] OpenSSL Handbuch http://www.absolute-cool.de/dokus/ssl/openssl/

B Dokumente aus der Diplomarbeit

[B.1] Sicherheitskonzept http://projects.hti.bfh.ch/webumfrage/thesis/doc/Sicherheitskonzept.pdf

[B.2] Pflichtenheft http://projects.hti.bfh.ch/webumfrage/thesis/doc/Pflichtenheft.pdf

[B.3] Benutzerdokumentation http://projects.hti.bfh.ch/webumfrage/thesis/doc/Benutzerdokumentation.pdf

C Software

[C.1] Javamail http://java.sun.com/products/javamail/

[C.2] OpenSSL http://www.openssl.org/

[C.3] JCR Java Content Repository Apache Jackrabbit http://jackrabbit.apache.org/

[C.4] JCE Java Cryptography Extension http://java.sun.com/javase/technologies/security/

[C.5] Apache Tomcat http://tomcat.apache.org/

[C.6] Apache Axis http://ws.apache.org/axis/

[C.7] JSF Apache MyFaces http://myfaces.apache.org/