UniCrypt-Portierung nach Javascript

Studiengang: BSc in Informatik | Vertiefung: IT-Security
Betreuer: Prof. Dr. Eric Dubuis, Prof. Philipp Locher
Experte: Han van der Kleij (Schweizerische Bundesbahnen SBB)

UniVote ist eine von der BFH entwickelte Internet-Abstimmungssoftware. Sie verwendet serverseitig UniCrypt, eine kryptographische Bibliothek. Um clientseitig auf den Rechnern der Wahlberechtigten die notwendigen Operationen ausführen zu können, muss UniCrypt nach JavaScript portiert werden.
Ziel der Bachelorarbeit ist es, die Grundlagen dazu zu untersuchen und eine Möglichkeit zur Portierung vorzuschlagen.

Einleitung

Wichtige Prinzipien der UniCrypt-Architektur sind Java spezifisch und lassen sich nicht ohne weiteres eins zu eins übernehmen. JavaScirpt ist eine prototypenbasierte Sprache und unterstützt klassische Java-Konstrukte wie Beispielsweise Vererbung, Typensicherheit oder Zugriffsmodifizierer nicht.
Wichtig für die Portierung sind die Komptabilität von UniCryptJS mit UniCrypt und die Wartbarkeit des JavaScript Codes.

Framework

Um die Java Sprach-Konstrukte in JavaScript abbilden zu können, habe ich ein Framework gebaut. Dieses ermöglicht eine ähnliche Syntax beibehalten zu können wie in Java, durch Abbildung von Konzepten wie der Vererbung oder des Überladens von Funktionen. Ohne das Framework müsste bei jeder übernommenen Klasse redundanter Code geschrieben werden, um durch die Prototypen Vererbung ein ähnliches Verhalten zu erreichen.

```
unicrypt.math.algebra.multiplicative.classes.GStarMod = Op.Class('GStarMod', {
    'extends': {
        'class' : unicrypt.math.algebra.multiplicative.abstracts.AbstractMultiplicativeCyclicGrou
        'generic' : ['GStarModElement', 'BigInteger']
    }
},{
    _modulus: null,
    _moduloFactorization: null,
    _orderFactorization: null,
    _superGroup: null,
    _init: function(moduloFactorization, orderFactorization) {
        this.$$super(u.BigInteger);
        this._modulus = moduloFactorization;
        this._moduloFactorization = moduloFactorization;
        this._orderFactorization = orderFactorization;
    }.paramType(['BigInteger', 'BigInteger']),
    getModulus: function() {
        return this._modulus;
    }.returnType('BigInteger'),
    getModuloFactorization: function() {
        return this._moduloFactorization;
    }.returnType('BigInteger'),
    getOrderFactorization: function() {
        return this._orderFactorization;
    }.returnType('BigInteger'),
    _defaultSelfApplyAlgorithm: function(element,posAmount) {
        return this._abstractGetElement(element.getValue()._modPow(posAmount, this._modulus));
    }.paramType(['GStarModElement','BigInteger']).returnType('GStarModElement'),
    _abstractContains: function(value) {
        return value.signum() > 0
                && value.compareTo(this._modulus) < 0
                && unicrypt.helper.math.MathUtil.areRelativelyPrime(value, this._modulus)
                && value.modPow(this.getOrder(), this._modulus).equals(u.BigInteger.ONE());
    }.paramType(['BigInteger']).returnType('boolean'),
    _abstractGetElement: function(value) {
        return new unicrypt.math.algebra.multiplicative.classes.GStarModElement(this, value);
    }.paramType(['BigInteger']).returnType('GStarModElement'),
```

GStarMod Klasse aus UniCryptJS

Pro:

- Übersichtlicher Code
- Ähnlichkeit zwischen UniCrypt und UniCryptJS
- Bei Änderungen der ECMAScript Spezifikation muss nur das Framework angepasst werden

Contra:

- Erhöhter Initialer Entwicklungsaufwand
- Komplexitätssteigerung
- Entwicklung nur für komplette Portierung von UniCrypt sinnvoll (war zeitlich aber nicht möglich)

Ergebnis

Konzepte von Java lassen sich mit der dynamischen Sprache JavaScript durch ein Framework abbilden:

- Vererbung
- Interfaces
- Typensicherheit
- Abstrakte Klassen
- Generische Klassen
- Statische Methoden
- Überladen von Methoden

In einem Proof of Concept konnte gezeigt werden, dass anhand des Beispiels von UniCrypt eine Portierung möglich ist. Kleine Teile der Bibliothek wurden in JavaScript umgesetzt. Mit reduziertem Aufwand liesse sich der Rest der Bibliothek mithilfe des Frameworks übernehmen.

Marcel Leroy Florian Portillo
marcel.portillo@gmx.ch