FACHHOCHSCHULE ERFURT UNIVERSITY OF APPLIED SCIENCES
Angewandte Informatik

Rechnen mit elliptischen Kurven

Projektdokumentation

André Wagenknecht
28. März 2018

Master – IT Projekt
in der Angewandten Informatik

Dipl.-Math. Anja Haußen

Inhaltsverzeichnis

1 Grundlagen

1.1 Elliptische Kurven

Eine elliptische Kurve ist definiert als:

> *„die Menge $\varphi = \varphi(K)$ der Lösungen $(x, y) \epsilon K^2$ einer kubischen Gleichung in zwei Variablen x und y mit einer auf der Kurve definierten Addition“* [1]

Zu diesen Kurven gehört die sogenannte „Weierstraß-Kurven“, welche sich mit der Gleichung:

$$y^2 = x^3 + ax + b$$

Abbilden lassen. Weitere elliptische Kurven sind die *„Edwards-Kurven“* und die *„Montgomery-Kurven“.*

1.1.1 Beispielrechnung

Gegeben sei die elliptische Kurve mit der Gleichung:

$$y^2 = x^3 + ax + b$$

Mit $a = b = 1$; $p = 5$ und einer affinen Ebene über $K = \mathbb{Z}_p$.
Hierbei gilt: $a, b \epsilon \mathbb{Z}_p$ und $4a^3 + 27b^2 \neq 0 \; mod \; p$.

Zur Bestimmung der elliptischen Kurve kann wie folgt vorgegangen werden:

1. Erzeugung einer Tabelle von y^2 der Elemente y von $\mathbb{Z}_p$

Tabelle 1: Elemente $\boldsymbol{y}$ von $\mathbb{Z}_{\boldsymbol{p}}$

y	0	1	2	3	4
$y^2 \; (mod \; 5)$	0	1	4	4	1

2. Erzeugung einer weiteren Tabelle mit den Werten von $f(x) = x^3 + 1 * x + 1$ für jedes $x \epsilon \mathbb{Z}_p$

Tabelle 2: Werten von $\boldsymbol{f(x) = x^3 + 1 * x + 1}$ für jedes $\boldsymbol{x \epsilon \mathbb{Z}_p}$

x	0	1	2	3	4
$x^3 + x + 1 \; (mod \; 5)$	1	3	1	1	1

3. Bestimmung der Paare (x, y) mittels vergleich der Tabellenwerte, für die y^2 und $f(x)$ übereinstimmen

$$\varphi_5 = \{(0,1),(0,4),(2,1),(2,4),(3,1),(3,4),(4,2),(4,3)\}.$$

1.2 HTML5 Web Workers

JavaScript selbst ist eine Single-Threaded-Umgebung, kann also nicht mehrere Skripte parallel ausführen. Um trotz dieser Einschränkung Leistungsintensive Berechnungen ausführen zu können, kann JavaScript mittels Web-Woker „Multi-Threadfähig" gemacht werden. Somit können auch die rechenintensiven Aufgaben in der Umgebung abgewickelt werden, ohne dabei die Benutzeroberfläche oder andere Scripte zu blockieren.

Eine Vollständige Dokumentation kann unteranderen auf der „W3schools" Webseite [2] eingesehen werden.

1.3 HTML5 Canvas

Ein Canvas ist ein mit HTML 5 eingeführter Grafikcontainer, der via JavaScript während der Laufzeit verschiedenen Grafischen Elementen gefüllt werden kann. Gleichzeitig enthält der Canvascontainer die Funktionen eines *„noScript-Containers"*. D.h. dass die in den Canvas-Tag eingeschlossenen Elemente nur angezeigt werden, wenn der Browser die Canvas-Funktion nicht unterstützt bzw. JavaScript deaktiviert wurde.

Eine vollständige Dokumentation kann auf der von Mozilla bereitgestellten Developer Dokumentation [3] eingesehen werden.

2 Anforderungen

2.1 Projektanforderungen

Das gesamte Projekt basiert auf der Kombination von JavaScript, HTML5 und CSS und ist somit komplett Plattformunabhängig. Zum Ausführen des Projektes ist die JavaScript engine „Node.js" [4] von Nöten. Für die Implementierung wurde die LTS Version 8.10 verwendet, jedoch ist das Projekt prinzipiell auch mit neueren Versionen kompatibel.

2.2 Buildanforderungen

Die zwei beigefügten Builds wurden direkt für Windows-Geräte mit 64bit kompiliert, jedoch ist es Problemlos möglich andere, Plattformübergreifende Builds zu erstellen. Die Anwendung selbst ist für eine minimale Auflösung von 1280x720 Pixeln konzipiert, jedoch auch bei geringeren Auflösungen bzw. Seitenverhältnissen ausführbar. Jedoch ist hier mit Suboptimaler Gestaltung sowie Scrollbalken zurechnen.

3 Verwendete Bibliotheken & Frameworks

3.1 Electron

Electron ist ein Open-Source Framework, das zur Erstellung von plattformübergreifenden Desktopanwendungen mittels HTML, CSS und JavaScript entwickelt wurde. Electron vollbringt dies durch die Kombination des Chromium-Browsers und Node.js zur Erstellung von Apps für Mac, Linux oder Windows. [5]

Im diesen Projekt kommt die Version 1.7.10 zum Einsatz.

3.2 Bootstrap

Bootstrap ist ein anpassungsfähiges und für Mobilgeräte optimiertes Framework, mit welchen sich viele übliche HTML-Elemente einfach gestalten lassen. Ebenso enthält dieses Framework viele nützliche Plugins, mit denen sich Webanwendungn einfach und modern gestalten lassen. [6]

Im diesen Projekt kommt die Version 3.3.7 zum Einsatz.

3.3 Bootstrap Colorpicker

Bootstrap Colorpicker ist ein Einfaches, auf JQuery basierendes Plugin, zur Bereitstellung eines mittels Bootstrap angepassten Colorpickers.

Eine Dokumentation der verfügbaren Funktionen kann auf der Website [7] eingesehen werden.

3.4 Electron-packager

Hierbei handelt es sich um ein Kommandozeilentool, das eine Nod.js-Bibliothek enthält. Mittels dieses Tools können Node.js-Anwendungen gepackt und uns für verschiedene Plattformen kompiliert werden, ohne dass zuvor Node.js auf dem auszuführenden System installiert werden muss.

Der, für den Buildvorgang benötigte Befehl ist im Abschnitt 3.8 Benötigte Konsolenanweisungen aufgeführt. Eine ausführliche Dokumentation, sowie alle möglichen Optionen sind auf der Github-Seite des Tools [8] einsehbar.

3.5 Sweetalert

Sweetalert ist ein einfaches Plugin zur dynamischen Erstellung von Popups, die mittels verschiedener Bootstrap Vorlagen gestylt werden können. Diese Alerts können wie normale JavaScript -Alerts verwendet werden.

Eine Ausführliche Dokumentation sowie weitere Verwendungszwecke ist auf der Website von sweetalert.js [9] zu lesen

3.6 JCanvas

JCanvas ist eine JavaScript-Bibliothek die mit und für JQuery geschrieben wurde. Sie umschließt dabei die Standart-Canvas-API und fügt viele anpassbare Funktionen hinzu, welche durch weitere, auf der Website [10] erhältliche Plugins, erweiterbar sind.

3.7 Better Round

Better Round, ist eine Erweiterung der in JavaScript enthaltenen Funktionen zum Runden von Zahlen, nach ihrer Kommastelle. Das Script hierzu ist etwas versteckt in der Mozilla Dokumentation [11] eben dieser Funktionen zu finden.

3.8 Benötigte Konsolenanweisungen

Im diesen Abschnitt wird kurz auf die benötigten Konsolenbefehle eingegangen, die während der Entwicklung oder des Buildvorganges benötigt werden. Eine ausführliche Dokumentation ist auf der Website des electron-Frameworks [5] zu finden.

1. Installieren der hinterlegten npm-Pakete

```
1.  npm install
```

2. Ausführen des Projektes

```
1.  electron .
```

3. Buildvorgang

```
1.  electron-packager <sourcepath> <appname>
    --platform=<platform> --arch=<arch>
2.
3.  platform options:
4.  Windows: „win32“
5.  MacOS: „darwin“
6.  Linux: „linux“
7.
8.  arch options:
9.  32 Bits OS: „x86“
10. 64 Bits OS: „x64“
11. ARM: „armv71“ (linux only)
```

4 Implementierung

4.1 Main.js

Die Main.js ist das Herzstück dieser Anwendung. Hier werden die allgemeinen Konfigurationen der Anwendung vorgenommen. Die wichtigste Funktion hierbei ist die „createWindow“, auf die nun näher eingegangen wird:

```
function createWindow () {

  mainWindow = new BrowserWindow({
      width: 1280,
      height: 720,
      frame: false,
      titleBarStyle: "customButtonsOnHover",
  });

  mainWindow.loadURL(url.format({
    pathname: path.join(__dirname, 'index.html'),
    protocol: 'file:',
    slashes: true
  }));

  mainWindow.webContents.openDevTools();


  mainWindow.on('closed', function () {
        mainWindow = null
  })
}
```

- *konstruktor BrowserWindow()*: hier werden die Grundeinstellungen des App-Fensters festgelegt. **`Width`** und **`height`** bestimmen die Standardgröße im normalen Fenstermodus, ist Auflösung des Bildschirms geringer als diese Einstellung wird das Fenster automatisch an die maximale Größe Angepasst. **`Frame`** bestimmt ob die App den Standartframe der jeweiligen Plattform erhalten soll, ist dieser Wert wie in dieser Anwendung auf „false“ gesetzt muss eine Custom-Titelbar eingefügt werden, da ansonsten keine Standartfunktionen wie minimieren, maximieren oder Schließen des Fensters möglich sind. Diese Custom-Titelbar wird mit dem Wert **`titleBarStyle`** initialisiert.

- *loadUrl():* bestimmt welches HTML-Template als Startseite geladen werden soll.

- *webContents.openDevTools():* ist diese Funktion nicht Auskommentiert öffnet die App in einen Debugmodus, in den Entwickler-Tools verfügbar sind. Diese Tools lassen sich durch externe Plugins [12] ergänzen.

- *on():* mit dieser Funktion können verschiedene events abgefangen, und spezialisiert werden.

4.2 Screen.js

Die Screen.js befasst sich mit den grundlegenden Funktionen des Fensters, die durch die vier Buttons im Header / der Custom-Titelbar angesprochen werden können. Sowie eine Funktion die stetig die aktuelle Größe des Fensters zurückgeben kann.

4.3 Canvas.js

Die Canvas.js ist eins der wichtigsten Bestandteile dieses Projektes. In dieser Datei werden sämtliche Funktionen, die das Zeichnen auf dem Canvas-Element betreffen, per JCanvas abgehandelt. Ebenso ist hier die allgemeine Konfiguration zu finden. Die Konfiguration enthält folgende Elemente:

- **`color:`** hier werden alle benötigten Farbcodes definiert
- **`font:`** Festlegung der Schriftart+ Schriftgrößen
- **`distance:`** Festlegung verschiedener benötigter Distanzen wie:
  - **`padding:`** allgemeiner Abstand der Zeichnungen zum Rand, damit keine Bestandteile abgeschnitten werden können.
  - **`stroke:`** länge der „Anstriche“
  - **`axes:`** zusätzliche Distanzen bei den Achsen um genügend Raum für Beschriftungen zu ermöglichen
  - **`arrow:`** länge des Achs-Pfeiles
  - **`beauty:`** Distanz, mit reiner kosmetischer Funktion
  - **`text:`** nötig zur Bestimmung des Beschriftungshintergrunds
  - **`corner:`** Rundungen des Beschriftungshintergrunds
- **`width:`** enthält verschiedene „Strichstärken“
  - **`axes:`** breite der Achsen
  - **`grid:`** breite des Grids
  - **`text:`** textbreite
- **`arrow:`** Konfigurationsgrößen der Achs-Pfeile
- **`point:`** Konfiguration der eingezeichneten Punkte

Ebenfalls enthält die Konfiguration Funktionen, mittels der während der Laufzeit sämtliche relevanten Farben sowie der Abstand des Gitternetzes verändert werden können.

Eine detaillierte Dokumentation, wie per JCanvas Elemente erzeugt werden können ist auf der Zugehörigen Website [10] sehr gut Dokumentiert, jedoch fehlt es an einigen Erklärungen, die den Umgang mit diesen Plugin enorm erleichtern. Diese werden im folgenden Abschnitt kurz erfasst.

Ein Punkt, der ein vor der Verwendung von Canvas-Elementen klar sein muss ist, dass jedes Element eine eigene Ebene besitzt. Ein neues Element wird also immer über ein altes gezeichnet. Diese Ebenen werden im einen Array gesichert, welches mittels JCanvas leicht zu manipulieren ist. Dazu muss nur im jeden Element die **`layer`**–variable auf „*true*“ gesetzt werden.

Ein weitere zu beachtende Eigenart ist die integrierte „Kantenglättung“ die alle Elemte mit einem Schlag sehr Pixelig erscheinen lassen. Diese setzt ein, wenn einen Element nicht ganzzahlige Werte übergeben werden und lässt sich lauf Dokumentation auch nicht deaktivieren. Der sicherste Weg diese Eigenschaft nicht zu aktivieren ist es jeden übergebenen Wert zu prüfen und gegeben falls per *parseInt()* Funktion diesen zu begradigen. Da dabei Pixel verloren gehen können, ist in der Konfiguration eine **`beauty`**-distanz hinterlegt, die diese, dabei entstehenden Lücken, wieder schließt.

4.4 CustomTechniques

Diese Datei enthält einige Funktionen, die ständig in Verwendung sind und einige Globale Variablen.

Preparer:

Der Preparer enthält einige Funktionen, die bereits zum Beginn der Anwendung verfügbar sein müssen. Ebenfalls enthalten ist die *redraw()* – Funktion, die mittels eines JavaScript-Workers (siehe 1.2 HTML5 Web Workers) einen neuen Thread startet, in den die Punkte der elliptischen Kurve berechnet werden, ohne das die komplette Anwendung einfriert.

UI:

Die UI enthält Funktionen, die durch z.B. Buttons der Benutzerfläche aktiv werden, diese validieren ggf. die benötigten variablen und reichen diese an die bearbeitenden Funktionen weiter.

Progress:

Progress enthält das aktivieren und deaktivieren der Ladeanimation während aufwendigen Rechenarbeiten.

Addition:

In der Addition werden die zwei vom User ausgewählten Punkte Addiert und das Ergebnis ausgegeben. Hierzu sind drei Helfer-funktionen notwendig, die die Berechnung stark vereinfachen:

1. mod(): diese Funktion wird in der folgenden Berechnung den Standard JavaScript-Modulo, mit dem es schnell zu Fehlebrechnungen kommen kann.

2. getInversArray(): diese Funktion sucht bereits vor der Berechnung alle inversen, die mit der übergebenen Primzahl infrage kommen.

3. getM(): in dieser Funktion wird die Steigung der nachfolgenden Rechnung bestimmt, hier wird ebenfalls die benötige Berechnung ausgewählt falls Q und P übereinstimmen.

4.5 Primes

Diese Datei enthält benötigten Funktionen, um Primzahlen zu bestimmen. Die Funktionen *„testAndGetNextPrime()“* und *„testAndGetLastPrime()“* endfangen jeweils eine Zahl, prüfen ob es sich dabei um eine Primzahl handelt und geben falls nötig die nächste bzw. vorherige berechenbare Primzahl aus.
Die Funktion *„testNumber()“* testet nur ob es sich bei der übergebenen Zahl um eine Primzahl handelt.

4.6 rationalPoints-Worker

Hierbei handelt es sich um den JavaScript-Worker, der innerhalb der redraw()-Funktion, die in der CustomTechniques.js zu finden ist. In dieser Funktion werden, wie in den Grundlagen (siehe 1.1.1 Beispielrechnung) beschrieben. Zusätzlich werden die höchsten x und y Koordinaten ermittelt, mit denen sich das zu zeichnende Grid leichter berechnen lässt.

5 Literaturverzeichnis

[1] R.-H. Schulz, H. Witten und B. Esslinger, „fu-berlin.de," 2015. [Online]. Available: http://page.mi.fu-berlin.de/rhschulz/Artikel_LogIn/Endfassung_LOG_IN_Elliptische_Kurven.PDF. [Zugriff am 26 März 2018].

[2] w3schools, „w3schools.com/," [Online]. Available: https://www.w3schools.com/html/html5_webworkers.asp. [Zugriff am 28 März 2018].

[3] mozilla, „developer.mozilla.org," [Online]. Available: https://developer.mozilla.org/de/docs/Web/Guide/HTML/Canvas_Tutorial/Grundlagen. [Zugriff am 28 März 2018].

[4] Node.JS, „nodejs.org," [Online]. Available: https://nodejs.org/en/.

[5] electron, „Electron," [Online]. Available: https://electronjs.org/docs.

[6] Bootstrap, [Online]. Available: https://getbootstrap.com/.

[7] Bootstrap Colorpicker, [Online]. Available: https://farbelous.github.io/bootstrap-colorpicker/index.html.

[8] electron-packager, [Online]. Available: https://github.com/electron-userland/electron-packager.

[9] sweetalert.js, [Online]. Available: https://sweetalert.js.org/.

[10] JCanvas, [Online]. Available: https://projects.calebevans.me/jcanvas/.

[11] Mozilla Dokumentation Ceil , [Online]. Available: https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global_Objects/Math/ceil.

[12] Devtools, [Online]. Available: https://electronjs.org/docs/tutorial/devtools-extension.