Benutzerdokumentation KryptoProjekt

Inhaltsverzeichnis

1. Basics

Über das „Basics“-Menü ist es möglich, sowohl verschiedene Datentypen zu erstellen, als auch Bausteine der Basisarithmetik und Matrizen aufzurufen.

1.1. Der Datentyp „Z“ (Reelle Zahlen mit Ganzzahl-Arithmetik)

Sie können hier den Wert direkt eintragen und den Baustein wie eine Zahl benutzen.

1.2. Der Datentyp „PrimeFieldElement“ (Element aus einem Primkörper)

Hier können der Wert und die Basis des Primkörpers eingetragen oder reelle Zahlen aus Z-Bausteinen hineingezogen werden.

Zur eingetragenen Basis kann über die Buttons „Addition Table“ und „Multiplication Table“ die jeweilige Arithmetiktabelle angezeigt werden.

1.3. Basisarithmetik

Auf den Basic Datentypen haben sie die die Möglichkeit mehrere Rechenoperationen auszuführen. Sie sind im Menü unter „Basics → Basic Arithmetik“ zu finden

Sie können die Datentypen mit den im Menü „Basic Arithmetic“ verfügbaren Operationen verknüpfen.

Mögliche Operationen sind:

- Addition
- Subtraktion
- Multiplikation
- Division
- Modulo-Rechnung
- Square-and-Multiply Algorithmus *
- Square-and-Multiply Algorithmus mit Modulo *
- Euklidscher Algorithmus *
- Erweiterter euklidscher Algorithmus *
- Phi Funktion *

* Über den „Extend“-Button haben Sie jeweils die Möglichkeit, sich die Zwischenergebnisse der Operation anzeigen zu lassen.

1.4. Matrizen

Die Matrixbausteine sind zu finden unter „`Basic → Matrices`“.

1.4.1. Neue Matrix (Z)

Die Eingabe einer Matrix sollte folgendermaßen aussehen:
Die einzelnen Zeilen der einzugebenden Matrix werden durch eine Pipe '`|`' getrennt, die Elemente innerhalb der Zeilen werden durch Komma '`,`' getrennt.

Die Anzahl der Elemente in jeder Zeile einer Matrix muss gleich sein (z.B. `3,2|2,1`).

1.4.2. Neue Matrix (Primefield)

Die Eingabe einer Matrix siehe Neue Matrix (Z)

Zusätzlich ist ein Parameter erforderlich, welcher das Primfeld dieser Matrix angibt, dies muss eine Primzahl sein.

1.4.3. Addition/Multiplikation einer Matrix

Bei diesen beiden Baukästen ist es möglich, Matrizen zu verlinken, eine Eingabe der Matrizen direkt in diesem Baukasten ist nicht möglich.

2. Primzahltests

Die Software unterstützt drei Primzahltests:

- Fermat-Test
- Miller-Rabin-Test
- Lucas-Test

Diese sind im Menü „`Primetests`“ eingebunden und werden im Folgenden kurz beschrieben.

2.1. Fermat-Test

Das Fermat-Test Fenster gliedert sich auf in zwei Textfelder zur Eingabe, einem „`settings`“-Auswahlmenü und einem „`Extend`“-Button.

In dem Textfeld „`check whether prime`“ wird die Zahl übergeben, welche auf Primzahleigenschaft überprüft werden soll (Modul). Es ist möglich, durch Kommata oder Leerzeichen getrennt, mehrere Zahlen und durch einen Gedankenstrich einen ganzen Zahlenbereich zu übergeben.

Beispiel:

•Eingabe

- `4` //Es wird überprüft ob die 4 eine Primzahl ist
- `3, 8, 101` oder `3 8 101` //Es wird überprüft, ob die Zahlen 3, 8 und 102 Primzahlen sind
- `4-20` //Es werden alle Zahlen zwischen (inklusive) 4 und 20 überprüft, ob diese Primzahlen sind.
  (Zwischen dem Gedankenstrich und den Zahlen darf kein Leerzeichen stehen!)
- `10.347.647` //Es wird überprüft, ob die Zahl 10347647 eine Primzahl ist.

Im Textfeld „`bases`“ werden die Basen eingegeben, welche für den Primzahltest verwendet werden sollen. Dabei gelten dieselben Formatierungen wie im oberen Textfeld.

Falls ein ungültiges Zeichen übergeben wurde, ändert sich die Schriftfarbe rot.

Unter „`settings`“ können zusätzliche Optionen eingestellt werden.
Zum Einen, ob die Wahrscheinlichkeit ausgegeben werden soll, ob es sich um eine Primzahl handelt oder nicht (Genauigkeit abhängig vom Primzahltest) und zum Anderen, ob zufällig Basen erzeugt werden sollen. Im Drehfeld (`spin box`) kann in Abhängigkeit zu den Primzahlen, die Anzahl der zu erstellenden Basen eingestellt werden. Es ist anzumerken das die Anzahl nicht zu groß eingestellt werden sollte, da dies zu einem unerwarteten Laufzeitverhalten führen kann.

Über den „`Extend`“-Button können die Zwischenergebnisse und die Wahrscheinlichkeit angezeigt werden.

Eine Kurzübersicht, ob es sich um eine Primzahl handelt oder nicht, wird zudem im „`Result`“-Fenster angezeigt.

In die Textboxen können von anderen Fenstern auch Verknüpfungen gezogen werden, um die Eingaben direkt aus anderen Fenstern zu erhalten.
Das Fermat-Test-Fenster kann wie die anderen Primzahl-Fenster auch, eine Liste von gefundenen Primzahlen weiterleiten. Dazu muss eine Verbindung vom Feld „`_primeFermat`“ zum Textfeld eines anderen Fensters gezogen werden.

2.2. Miller-Rabin-Test

Die Funktionalität dieses Fensters deckt sich mit dem vom Fermat-Test, nur das die Primzahlen mittels des Miller-Rabin-Algorithmus bestimmt werden.

2.3. Lucas-Test

In diesem Fenster stehen bisher nur zwei Textfelder zur Verfügung.

In das Textfeld „`prime factors`“ werden die Primfaktoren des Lucas-Term eingegeben. Die Faktoren werden mittels Multiplikationszeichen ‘`*`‘ verbunden. Potenzen werden durch ‘`^`‘ unterstützt.

Falsche Eingaben werden durch die Rotfärbung der Schrift hervorgehoben.

Der letzte Summand ist automatisch auf 1 gesetzt und kann nicht verändert werden.

In dem „`bases`“-Textfeld werden die Basen übergeben, mit denen die durch den Term dargestellte Zahl auf Primzahleigenschaft überprüft werden kann. In dem „`bases`“-Textfeld werden die gewohnten Formatierungen unterstützt.

3. Kodierung

Die verschiedenen Kodieralgorithmen sind im Menü unter „Coders“ zu finden.

3.1. Hamming-Code

3.1.1. Hamming-Code initialisieren

Um einen Hamming-Code zu erstellen, muss zunächst der Frame "Initialize Hammingcode" ausgewählt werden. Hier muss im ersten Textfeld das "source codeword" eingegeben werden.

Mit der Checkbox lässt sich auswählen, ob eine Matrix erzeugt wird oder ob der Benutzer selbst eine Matrix erzeugen will.
Ist dies der Fall, muss ein neuer Matrixbaustein ausgewählt werden und das Ergebnis in das zweite Textfeld gezogen werden.
Um mit dem Hamming-Code fortzufahren muss das Hammingcode-Element („_hcElem“) in den nächsten Baustein gezogen werden. Die anderen Zwischenergebnisse können gegebenenfalls anderweitig verwendet werden.

3.1.2. Hamming-Code enkodieren

Mit dem Baustein "Encode Hammingobject" lässt sich ein Hammingcode-Element encodieren. Dazu muss lediglich ein Hammingcode-Element in das Textfeld gezogen werden.

3.1.3. Syndrom berechnen

Hier ist gleich zu Verfahren wie beim Baustein Encode Hammingobject Das übergebene Hammingcode-Element muss zuvor jedoch encodiert worden sein, damit das Syndrom berechnet werden kann.

3.1.4. Fehler im enkodierten Codewort erzeugen

Mit folgendem Baustein ist es möglich Fehler in einem "encoded Word" zu erzeugen. Dafür muss das Hammingcode-Element vorher enkodiert worden sein.

Nun ist es möglich über das zweite Textfeld anzugeben, mit welcher Wahrscheinlichkeit Fehler erzeugt werden sollen. Die Wahrscheinlichkeit muss zwischen 0 und 1 liegen (bedeutet: zwischen 0% und 100%). Des Weiteren müssen Werte wie 0.xx immer durch einen Punkt und nicht durch ein Komma getrennt werden.

3.1.5. Hamming-Code dekodieren

Um ein enkodiertes Hammingcode-Element wieder zu dekodieren, muss dieses in das erste Textfeld gezogen werden.

Mit der Checkbox lässt sich weiterhin auswählen, ob das zu dekodierende Wort im Falle eines Fehlers, wenn möglich, korrigiert werden soll.

3.1.6. Hammingdistanz berechnen

Um diesen Baustein nutzen zu können, müssen zwei neue Matrix-Bausteine erzeugt werden. Diese werden dann in die beiden Textfelder gezogen. Danach kann die Distanz zwischen den beiden Matrixbausteinen berechnet werden.

Hier ist zu beachten, dass die Matrizen „einzeilig“ sein müssen, da bei der Hammingdistanz-berechnung nur Vektoren berücksichtigt werden können.

3.1.7. Vektorengewicht berechnen

Hierzu muss ebenfalls ein Matrixbaustein erzeugt und in das entsprechende Textfeld gezogen werden. Dann ist es möglich das Gewicht des angegebenen Vektors zu berechnen. Auch hier werden nur „einzeilige“ Matrizen bearbeitet/behandelt.