JuniorAkademie Adelsheim

17. Science Academy Baden-Württemberg 2019

Astronomie

Biologie

Informatik

Mathematik

Philosophie

TheoPrax

Regierungspräsidium Karlsruhe Abteilung 7 – Schule und Bildung

DIETER SCHWARZ STIFTUNG Stiftung Bildung und Jugend HOPP FOUNDATION for computer literacy & informatics

Dokumentation der JuniorAkademie Adelsheim 2019

17. Science Academy Baden-Württemberg

Veranstalter der JuniorAkademie Adelsheim 2019:
Regierungspräsidium Karlsruhe
Abteilung 7 –Schule und Bildung–
Hebelstr. 2
76133 Karlsruhe
Tel.: (0721) 926 4245
Fax.: (0721) 933 40270
www.scienceacademy.de
E-Mail: joerg.richter@scienceacademy.de
monika.jakob@scienceacademy.de
rico.lippold@scienceacademy.de

Die in dieser Dokumentation enthaltenen Texte wurden von der Kurs- und Akademieleitung sowie den Teilnehmerinnen und Teilnehmern der 17. JuniorAkademie Adelsheim 2019 erstellt. Anschließend wurde das Dokument mithilfe von LaTeX gesetzt.

Gesamtredaktion und Layout: Jörg Richter

Vorwort

Rund 100 verschiedene „Elemente“ versammelten sich im Juni 2019 am Landesschulungszentrum für Umwelterziehung in Adelsheim, die 17. Science Academy Baden-Württemberg konnte beginnen. Am Eröffnungswochenende lernten wir uns kennen: Teilnehmerinnen und Teilnehmer sowie das gesamte Leitungsteam. Während der Sommerakademie entstanden aus den unterschiedlichen Elementen immer neue Verbindungen, und so entwickelte sich eine einzigartige Atmosphäre. Mit dem Schreiben dieser Dokumentation hielten wir am Abschlusswochenende neben den fachlichen Ergebnissen auch alle unsere persönlichen Erlebnisse fest.

Anlässlich des diesjährigen Jahrs des Periodensystems stand die Akademie unter dem Motto „Elemente“. Das Motto gibt durch verschiedene Aktionen und Aufgaben immer wieder Anlass zum Nachdenken und Reflektieren über die sehr intensive gemeinsame Zeit mit vielen neuen Erkenntnissen und Eindrücken.

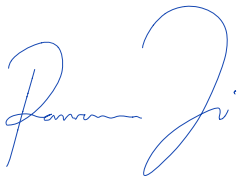
In den sechs Kursen beschäftigten sich die Teilnehmerinnen und Teilnehmer mit dem Mond, nachhaltigen Medikamenten, Verschlüsselungsmethoden, mathematischer Magie und Kältemaschinen. Dabei probierten sie viele neue Methoden aus, erhielten einen Einblick in das wissenschaftliche Arbeiten und trainierten persönliche Fähigkeiten wie Teamwork, Präsentieren, Projektmanagement und vieles mehr.

Allerdings bestand die gesamte Akademiezeit neben den Kursen auch aus verschiedenen anderen Elementen wie den kursübergreifenden Angeboten, dem Sportfest, dem Wandertag und noch vielen weiteren gemeinsamen Aktionen.

Insgesamt entstand so die einzigartige Akademieatmosphäre, welche für neue Freundschaften, aber auch den ein oder anderen Ohrwurm sorgte.

Wir wünschen Euch und Ihnen viel Spaß beim Lesen und Stöbern, viele schöne Einblicke in unsere Akademiezeit und hoffen, dass Ihr Euch noch lange an die einzigartige gemeinsame Zeit erinnert!

Eure/Ihre Akademieleitung



Ranran Ji (Assistenz)

Lorenz Löffler (Assistenz)

Dr. Monika Jakob

Jörg Richter

Inhaltsverzeichnis

Kurs 3 – Kryptographie

Vorwort

Henriette Neuschwander, Kevin Sommer, Michael Krüger

(EBG13) Rf vfg vzzre jvrqre refgnhayvpu, jvr zna zvg Arhtvre haq Syrvß va ahe mjrv Jbpura xbzcyrkr Gurzra fb irefgrura xnaa, qnff zna rva shaxgvbavreraqrf Fbsgjnercebqhxg qnenhf znpura xnaa. Hafre Xhef ung qvrfr Nhstnor zvg Oenibhe trzrvfgreg haq aroraure fbtne rva Cyüfpu-Enzn-Ynzn nhstrmbtra. Zvg Fgbym ceäfragvrera jve aha qvr Retroavffr hafrere Xhefgrvyaruzre.

Einleitung und Kursziele

Isabel

Viele Menschen nutzen das Internet, um Kleidung, Bücher oder andere Waren zu bestellen oder in Foren zu diskutieren. Dabei ist vielen gar nicht bewusst, wie unsicher Kommunikation im Internet eigentlich ist. Denn im Gegensatz zu der Annahme vieler wird eine Nachricht, die man über das Internet verschickt, nicht direkt an den Empfänger gesendet, sondern passiert auf dem Weg dorthin noch viele verschiedene Stationen. Jede dieser Stationen ist theoretisch in der Lage, diese Nachrichten abzufangen. Damit aber auch persönliche Daten sicher über das Internet verschickt werden können, müssen diese verschlüsselt werden. Mit dieser Problematik haben wir, der Informatikkurs der Science Academy 2019, uns beschäftigt.

Unser Ziel war es, einen Chat in Form einer Website zu programmieren, dessen Nachrichten mathematisch nachweisbar verschlüsselt sind, sodass nur Sender und Empfänger die Nachricht entschlüsseln können. Um überhaupt einen groben Überblick zu bekommen, was alles pro-

grammiert werden muss, teilten wir uns am Anfang in mehrere Gruppen auf und überlegten, welche Elemente wichtig für einen Chat sind. Nach Zusammentragen der Ergebnisse entschieden wir dann, dass unser Chat in Form einer Website mit Registrierung und Login Gestalt annehmen sollte. Da wir für unseren sicheren Chat aber nicht nur eine Benutzeroberfläche, sondern auch eine sichere Verschlüsselung benötigen, teilten wir uns für die Kurszeit in zwei Gruppen auf. Während die einen die Programmierung vertieften, beschäftigten sich die anderen mit der Kryptographie, der Wissenschaft der Verschlüsselung, und erarbeiteten sich die für unseren Chat verwendete komplizierte Mathematik hinter der Verschlüsselung.

Unser Kurs

Anna-Lena ist sehr mathebegeistert, sie konnte alle unsere Mathefragen beantworten. Durch ihre angenehme, ruhige Art hat sie die Gruppe positiv beeinflusst. Im Kurs hat sie sich zuvorkommend verhalten und und war stets bei der Sache und konzentriert. Kurz gesagt war sie immer motiviert und motivierend. Sie ist eine sehr kreative Person und teilt diese Kreativität auch gerne mit den anderen Teilnehmern. Außerhalb des Kurses ist sie sehr an Volleyball interessiert und freute sich immer, wenn auf dem Beachvolleyballfeld gespielt wurde. Letztendlich hat Anna-Lena den Kurs entscheidend vorangebracht.

Catharina stach besonders durch ihr außergewöhnlich ausgeprägtes Interesse an der Mathematik heraus. Daher nahm sie sehr aktiv an der Mathe-Gruppe teil und erklärte den anderen durch ihre schnelle Auffassungsgabe auch schwierigere mathematische Formeln und Beweise in ausführlichen Vorträgen. Ihre mathematische Begeisterung reichte sogar so weit, dass sie sich durch ihre gute Freundin Franka aus dem Mathe-Kurs über die Aktivitäten dort informierte. Außerdem konnte sie durch ihre Vorkenntnisse beim Programmieren mit Python auch oft den anderen Teilnehmern helfen. Ihr MINT-Interesse sticht stark hervor, was zu diversen Projekten bei Jugend forscht und dem Traumberuf als Astrophysikerin führt. Zusätzlich engagierte sie sich auch im Theater und spielt Klavier.

Henriette ist immer freundlich und hilfsbereit. Zudem ist sie immer motiviert, egal um was es geht. Mit ihrer taffen und powervollen Art, ihrer Energie und ihrer lauten Stimme kam sie in unserer Gruppe gut klar und sorgte für ein humorvolles Gruppenklima. Trotz unserer kurzen Aufteilung des Kurses in Informatik und Mathematik blieb sie beim Programmieren. Durch ihre Informatik-Vorkenntnisse konnte sie oft die Verantwortung übernehmen und unseren Kurs sehr unterstützen. Auch in der Freizeit war sie immer gelassen, locker und voller Energie.

Isabel steckte uns alle mit ihrer guten Laune an und überzeugte uns mit ihrer freundlichen Persönlichkeit. Sie hat eine schnelle Auffassungsgabe und lässt sich für fast alles begeistern: So nahm sie zum Beispiel auch an der Theater-KüA teil. Ihre Motivation sprang auch beim Präsentieren auf ihr Publikum über und sie trat immer souverän und überzeugend auf. Mit Valentin programmierte sie die Registrierung unseres Chats und bildete mit ihm ein wahres Dream-Team.

Jannik ist sehr sportlich, deshalb besuchte er meistens die Sport-KüA. Außerdem war er immer lustig und gut drauf, auch als er keine Socken hatte. Er machte jeden Spaß mit, war aber im Kurs konzentriert und konnte gut bei Fragen aushelfen. Er hat ein gutes Auffassungsvermögen und konnte schon nach kurzer Zeit programmieren. Er war der älteste im Kurs, deshalb auch meistens der Reifste. Aufgrund seines Nachnamens wurde ihm der Spitzname Bopp der Programmiermeister verliehen.

Lisa tat sich gleich am Eröffnungswochenende mit ihrer Begeisterung für Marvel-Filme hervor, weswegen ihr der Titel „Miss Marvel“ verliehen wurde. Im Kurs war sie mit ihren Informatik-Vorkenntnissen eine große Hilfe und motivierte uns immer mit ihrer positiven Grundeinstellung. Sie ist nicht nur sehr humorvoll und fröhlich, sondern

auch sehr unkompliziert, weshalb sie sich in ihrer Freizeit mit jedem gut verstand.

Moritz war für alle eine Bereicherung. Seine angenehme und zuvorkommende Persönlichkeit zeigt sich besonders im Gespräch mit ihm, und er ist immer für einen da. Man kann mit ihm über alle Themen reden. Im Kurs arbeitete er immer konzentriert mit und behielt auch bei schwierigen und verwirrenden mathematischen Aufgaben stets den Überblick. Außerdem erklärte er immer geduldig und konnte seine Themen auch in Präsentationen gut erklären. Er übernahm oft die Verantwortung, egal wie kompliziert seine Aufgabe war.

Nico hatte bereits Vorkenntnisse in Sachen Programmieren, weshalb er häufig nach Hilfe gefragt wurde. Dabei war er stets motiviert und vor allem denjenigen gegenüber hilfsbereit, die sich in Michaels Mathe-Kurs mit den mathematischen Hintergründen beschäftigten. Aber auch außerhalb des Kurses war Nico immer ein guter und sympathischer Gesprächspartner, der viel Humor zeigte. Außerdem unterstützte er Nathans Band in ihren knappen Probezeiten vor dem Hausmusikabend.

Tim arbeitete immer ruhig, konzentriert und mit Leidenschaft an seinen Projekten. Durch seine Begeisterung konnte er unseren Kurs sehr bereichern und unterstützen. Er ist sehr pünktlich und auch privat sehr großzügig, zum Beispiel bei dem Verleih seiner „Tuba“ (Tenorhorn), mit der er auch leidenschaftlich im Akademieorchester spielte. Außerdem ist er ein wahrer Schwabe, was sich vor allem an seinem Dialekt zeigt.

Timon hatte bereits Vorkenntnisse beim Programmieren mit Java. Er arbeitete mit großer Motivation, war hilfsbereit und zeigte großes Engagement bei der Zeitungs-KüA. Des Weiteren sorgte er oft für gute Stimmung und viel Entertainment. Er hatte gute Ideen und hielt die Gruppe am Laufen. Timon bot auch eine Scratch-KüA an, in der Programmieranfänger lernten, einfache Dinge zu programmieren.

Valentin sorgte mit seiner gelassenen und humorvollen Art für ein gutes Gruppenklima. Durch seine Vorkenntnisse in der Informatik und seinem riesigen Ordner voller Tipps und Tricks zur Programmierung konnte er unsere Gruppe sehr gut unterstützen. Außerdem hat er auch in Sachen Kleidung einen guten Geschmack. Egal, welches Wetter herrschte, konnte man sich darauf verlassen, dass er immer ein Hemd trägt.

Vilmos ist sehr sportlich und war deshalb etwas traurig, darüber, dass er nie an der Sport-KüA teilnehmen konnte. Stattdessen spielte er im Akademie-Orchester Klavier, wobei er, wie auch im Kurs, sehr viel Motivation zeigte. Er war Mitglied von Michaels Mathe-Kurs, der sich zwischenzeitlich mit den mathematischen Hintergründen beschäftigte, und konnte den „Programmierern“ stets neue Erkenntnisse liefern. Er half bei Fragen immer gut weiter und wusste sich auch bei Präsentationen gut auszudrücken.

Rama-Lama war ein unerwarteter Teil des Informatik-Kurses. In Anlehnung an das Lied „Rama Lama Ding Dong“, das jeden Tag nach dem Plenum gespielt wurde, wurde es von Kevin bestellt und war von diesem Zeitpunkt an unser treues Kursmaskottchen. Es unterstützte uns immer tatkräftig – soweit es konnte – und war eine sehr große Bereicherung für unseren Kurs, weil es unsere Gruppe noch mehr zusammenschweißte. Außerdem war es ein gutes Foto-Model für unsere Schülermentorin. Als Andenken an das Rama-Lama haben wir in unserem Chat, neben Michaels zwei GIFs, auch ein GIF von unserem geliebten Maskottchen erstellt.

Michael machte mit seiner humorvollen Art auf sich aufmerksam. Er lacht auch gerne über sich selbst, was sich darin zeigt, dass in unserem Chat zwei GIFs, in denen er tanzt, verschickt werden können. Als Lehrer konnte er uns alles anschaulich mit lustigen Beispielen erklären. Geduldig beantwortete er die scheinbar unendlich vielen Fragen von uns. Er gab auch nach zwei misslungenen Beispielen zur RSA-Verschlüsselung nicht auf. Darüber hinaus hatte er er ein gutes Auge, um die Fehler in unserem Code zu

entdecken. Doch er hat auch eine ernste Seite. Ihm liegt der Umwelt- und Klimaschutz sehr am Herzen. Das zeigt sich darin, dass er eine Klimaschutz-KüA anbot.

Kevin war als begabter Informatiker eine riesige Unterstützung, im Kurs, aber auch außerhalb. Während seiner Freizeit programmierte er den für unseren Chat wichtigen Server und scheute dabei keine Mühen. Er erfüllte uns alle Wünsche beim Programmieren und half uns, diese umzusetzen, dennoch ließ er uns unseren Freiraum, sodass wir selbst Hand anlegen konnten. Als wir unseren Kurs in Mathematik und Informatik aufteilten, bemühte er sich, den „Mathematikern“ die verpassten Programmiergrundlagen beizubringen. Bei den Abendveranstaltungen der Science Academy kümmerte er sich um die gesamte Technik, insbesondere um den Sound. Kevin sorgte für ein angenehmes und lustiges Gruppenklima, dies zeigte sich auch, als er unser Kursmaskottchen Rama-Lama kaufte, wofür wir alle sehr dankbar sind.

Henriette ist begeisterte Schülermentorin mit Herz und Seele. Sie sorgte für die immerwährende Gruppendynamik. Als rasende Reporterin und begabte Hobbyfotografin war sie sich nicht zu schade, alle Teilnehmer mal mehr und mal weniger gut abzulichten. Doch auch weitere wichtige Aufgaben außerhalb des Inhaltlichen wurden von ihr übernommen. Auf der einen Seite bestand ihre Zuständigkeit darin, den Kurs zu jeder Tages- und Nachtzeit mit Süßigkeiten zu verpflegen. Doch vor allem an den Tagen vor dem Sportfest achtete sie auf unsere Ernährung. Der gesamte Kurs wurde auf Diät gesetzt und mit von ihr geschnittenem Gemüse versorgt. Diese Maßnahme und ihre motivierenden Anfeuerungen während des Sportfestes brachten uns letztendlich den dritten Platz ein.

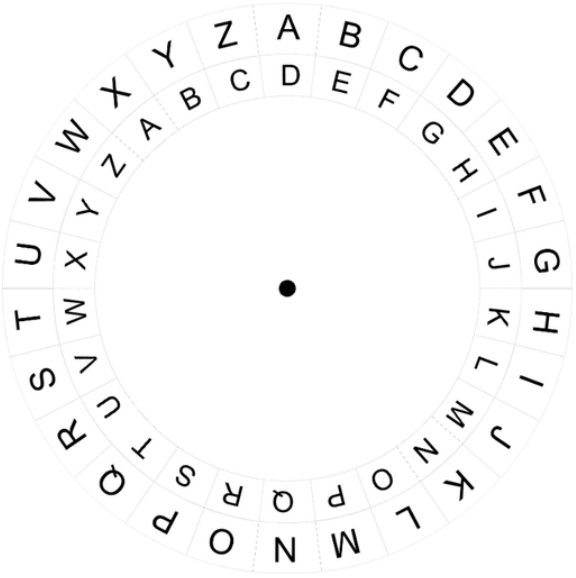
Grundlagen der Verschlüsselung

Valentin, Lisa, Isabel

Wir haben uns im Kurs mit verschiedenen Verschlüsselungstechniken beschäftigt. Verschlüsselung hat das Ziel, eine Nachricht nur für den gewünschten Kommunikationspartner lesbar zu machen. Somit ist es möglich, Nachrichten über einen nicht sicheren Kanal – beispielsweise einen Boten oder das Internet – zu transportieren.

Caesar-Verschlüsselung

Zum besseren Verständnis haben wir uns am Anfang unseres Kurses mit der sehr einfachen Caesar-Verschlüsselung beschäftigt. Bei diesem Verfahren verschieben wir das Alphabet um eine bis 25 Stellen. Zur besseren Veranschaulichung haben wir uns die Caesar-Scheibe angesehen:



Caesarscheibe zur Ver- und Entschlüsselung

Das äußere Alphabet ist das bekannte Klartextalphabet. Der Klartext ist der Text, den wir verschlüsseln wollen, also unsere Nachricht. Das innere Alphabet ist gedreht. Um wie viele Stellen es gedreht ist, ist geheim. Nur der Sender und Empfänger wissen es. In diesem Beispiel ist das innere Alphabet (Geheimalphabet) um drei Stellen verschoben, sodass A zu D wird.

Möchte man nun eine Nachricht verschlüsseln, so reduziert man als erstes die Nachricht auf Buchstaben des Alphabets. Andere Zeichen wie beispielsweise Satzzeichen, Umlaute, Leerzeichen und Zahlen werden weglassen oder ausgeschrieben.

Beispiel: *„Cäsar ist toll!“* wird zu *„CAESARISTTOLL“*

Anschließend verstellen wir unsere Scheibe um eine bestimmte Anzahl an Stellen. Jetzt steht unter dem A des Klartextalphabets (außen) das D des Geheimalphabets (innen). Nun verschlüsseln wir jeden Buchstaben des Klartextes mit dem zugehörigen Buchstaben des Geheimalphabets.

Beispiel: *„CAESARISTTOLL“* wird zu *„FDHVDULVWWROO“*

Jetzt kann die Chiffre, also die verschlüsselte Nachricht, verschickt werden.

Um die Nachricht zu entschlüsseln, stellt der Empfänger die Scheibe gleich ein. Dann wird der obige Prozess umgedreht. Das bedeutet, dass vom Geheimalphabet die Buchstaben abgelesen werden und jedem Geheimbuchstaben (innen) der passende Klartextbuchstabe (außen) zugeordnet wird.

Beispiel: *„FDHVDULVWWROO“* wird zu *„CAESARISTTOLL“*

Nun werden die Satzzeichen, Umlaute, Leerzeichen, etc. eingefügt.

Beispiel: *„CAESARISTTOLL“* wird zu *„Cäsar ist toll.“*

Allerdings gibt es einige Probleme. So wurde im obigen Beispiel beispielsweise ein Ausrufezeichen zu einem Punkt, da dieses nicht übermittelt werden konnte. Das hängt damit zusammen, dass wir mit der Caesar-Verschlüsselung nur Buchstaben verschlüsseln können.

Das größte Problem ist allerdings die Sicherheit. Da wir das Alphabet um 0 bis 25 Stellen verschieben können, gibt es nur 26 verschiedene Schlüssel, wovon einer (Verschieben um 0 Stellen) gar nichts verändert und einfach den Klartext wiedergibt. Das heißt, dass selbst ein Mensch durch einfaches Ausprobieren innerhalb kurzer Zeit auf den Schlüssel kommen könnte.

Ein weiterer Schwachpunkt ist die Häufigkeitsanalyse. Wenn man die Sprache kennt, in der die Nachricht verfasst ist, kann man besonders bei langen Texten über die Häufigkeit des Auftretens einzelner Buchstaben Wahrscheinlichkeitsaussagen über den Buchstaben machen. So kann ein häufiges Auftreten von H im deutschen Geheimtext auf eine A-zu-D-Verschlüsselung hinweisen, da der Buchstabe E in deutschen Texten im Durchschnitt am häufigsten auftritt und bei der A-zu-D-Verschlüsselung durch ein H ersetzt wird.

Symmetrische und Asymmetrische Verschlüsselung

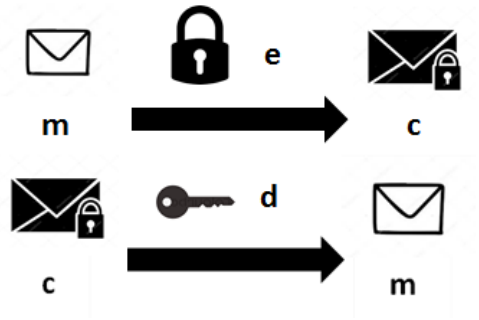
Bei einer symmetrischen Verschlüsselung wie zum Beispiel der Caesar-Verschlüsselung benötigt man nur einen Schlüssel. Dieser wird zum Ver- und Entschlüsseln benutzt. Das klingt sehr praktisch, trotzdem bringt es ein Problem mit sich: Wie kann man den Schlüssel sicher an den Empfänger weiterleiten?

Früher wurden diese Schlüssel meist persönlich über einen Boten übergeben, doch die weiten Entfernungen zwischen Sender und Empfänger machen uns einen Strich durch die Rechnung. Deshalb wird heute meist eine asymmetrische Verschlüsselung angewendet, um den Schlüssel sicher übertragen zu können.

Bei der asymmetrischen Verschlüsselung, wie zum Beispiel der RSA-Verschlüsselung, benötigt man zwei Schlüssel. Einen privaten Schlüssel d (für Englisch: decrypt), auch Private Key genannt, und einen öffentlichen Schlüssel e (für Englisch: encrypt), auch Public Key. Diese Schlüssel heben einander auf. Man kann sie sich vorstellen wie ein Schloss und den passenden Schlüssel dazu.

Hierzu haben wir eine passende Demonstration erstellt: Man möchte mit jemand anderem Nachrichten austauschen. Dafür muss der Nachrichtenempfänger erst den eigenen öffentlichen Schlüssel e, also in unserer Demo das Schloss, durch das Internet versenden, in unserem Beispiel eine Sitzreihe, durch die das Schloss gegeben wird.

Nun kann der Nachrichtenversender einen Brief mit einer Nachricht m (für Englisch: message) versenden. Diese Nachricht wird mit dem Schloss, welches er zuvor erhalten hat, verschlossen. Für unsere Verschlüsselung bedeutet dies, dass die Nachricht mit dem öffentlichen Schlüssel e des Empfängers verschlüsselt wird. Danach wird der verschlossene Brief c (für Chiffre) über das Internet (Sitzreihe im Klassenzimmer)



Prinzip asymmetrische Verschlüsselung

an den Empfänger weitergegeben, und dieser kann ihn mit seinem privaten Schlüssel d, den er geheim gehalten hat, entschlüsseln.

Dennoch birgt sich eine Gefahr hinter dieser asymmetrischen Verschlüsselung. Der Empfänger darf seinen privaten Schlüssel auf keinen Fall an jemand anderen weitergeben, denn dieser könnte dann die Nachrichten entschlüsseln. Außerdem darf der private Schlüssel d nicht aus dem öffentlichen Schlüssel e ableitbar sein. Wie wir diese Probleme gelöst haben, sieht man bei der Registrierung (Seite 65) und bei der RSA-Verschlüsselung (Seite 60).

Grundlagen Mathematik

Nico, Tim, Catharina, Moritz, Vilmos, Anna-Lena

Neben den Grundlagen der Verschlüsselung benötigen wir für unseren Chat auch eine Reihe an mathematischen Grundkenntnissen, um unser Kursziel – einen Chat mit sicherer Verschlüsselung – zu erreichen.

Primzahlen

Primzahlen sind natürliche Zahlen, die nur durch eins und sich selbst teilbar sind. Dies ist eine sehr wichtige Eigenschaft für unsere Verschlüsselung. Für unsere Kryptographie benötigen wir sehr große Primzahlen.

Ein einfacher Weg, Primzahlen zu finden, ist das Sieb des Eratosthenes. Dazu fertigt man eine Liste der Zahlen von 2 bis n an.

	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

Dann beginnt man mit der 2 und streicht alle Vielfachen von ihr weg.

	2	3	~~4~~	5	~~6~~	7	~~8~~
9	~~10~~	11	~~12~~	13	~~14~~	15	~~16~~
17	~~18~~	19	~~20~~	21	~~22~~	23	~~24~~
25	~~26~~	27	~~28~~	29	~~30~~	31	~~32~~
33	~~34~~	35	~~36~~	37	~~38~~	39	~~40~~
41	~~42~~	43	~~44~~	45	~~46~~	47	~~48~~

Anschließend geht man zur nächsten nicht durchgestrichenen Zahl und streicht wiederum alle Vielfachen von ihr weg. Dies wird so lange wiederholt, bis keine weitere nächstgrößere Zahl nicht durchgestrichen ist. Alle Zahlen, die nun nicht durchgestrichen wurden, sind prim.

	$\boxed{2}$	$\boxed{3}$	~~4~~	$\boxed{5}$	~~6~~	$\boxed{7}$	~~8~~
~~9~~	~~10~~	$\boxed{11}$	~~12~~	$\boxed{13}$	~~14~~	~~15~~	~~16~~
$\boxed{17}$	~~18~~	$\boxed{19}$	~~20~~	~~21~~	~~22~~	$\boxed{23}$	~~24~~
~~25~~	~~26~~	~~27~~	~~28~~	$\boxed{29}$	~~30~~	$\boxed{31}$	~~32~~
~~33~~	~~34~~	~~35~~	~~36~~	$\boxed{37}$	~~38~~	~~39~~	~~40~~
$\boxed{41}$	~~42~~	$\boxed{43}$	~~44~~	~~45~~	~~46~~	$\boxed{47}$	~~48~~

Alle umrandeten Zahlen sind also Primzahlen. Wie man aber sieht, ist das Sieb des Eratosthenes nicht für größere Zahlen geeignet. Ein für diese Zahlen geeignetes Verfahren ist der Miller-Rabin-Test, Seite 10.

Größter gemeinsamer Teiler

Der größte gemeinsame Teiler zweier Zahlen setzt sich aus ihren gemeinsamen Primfaktoren zusammen. Ein Primfaktor ist ein Teiler einer natürlichen Zahl. Dabei ist jeder Primfaktor selbst eine Primzahl. Beispielsweise ist der $\mathrm{ggT}(60, 75) = 15$.

Zunächst zerlegen wir die Zahlen, deren ggT gesucht ist, in ihre Primfaktoren: Die Primfaktoren von 60 sind

$$2 \cdot 2 \cdot 3 \cdot 5 = 60$$

und die Primfaktoren von 75 sind

$$3 \cdot 5 \cdot 5 = 75$$

Die gemeinsamen Primfaktoren sind 3 und 5. Daraus folgt

$$\text{ggT}(60, 75) = 3 \cdot 5 = 15.$$

Euklidischer Algorithmus

Alternativ zur Primfaktorzerlegung beider Zahlen kann der ggT(a,b) von zwei Zahlen a und b auch mit dem euklidischen Algorithmus berechnet werden. Zunächst schreibt man die zwei Zahlen a und b, wobei $a > b$ gilt, in zwei Spalten nebeneinander, sodass a in der linken Spalte steht. Dann rechnet man $\frac{a}{b}$ und schreibt das Ergebnis als ganze Zahl in eine dritte Spalte. Den Rest der Division schreibt man im nächsten Schritt in die b-Spalte und überträgt den b-Wert aus Schritt 1 in die a-Spalte.

Der Vorgang wird wiederholt, bis der Rest 0 beträgt. Die letzte Zahl in der b-Spalte über 0 ist der größte gemeinsame Teiler von a und b. Beim ersten Beispiel erfahren wir, dass der größte gemeinsame Teiler aus 75 und 60 die Zahl 15 ist. Das zweite Beispiel berechnet ggT(143, 117)

Schritt	a	b	$q = \frac{a}{b}$
1	75	60	1
2	60	15	4
3		0	

Beispiel 1: Der ggT aus 75 und 60 ist 15.

Schritt	a	b	$q = \frac{a}{b}$
1	143	117	1
2	117	23	5
3	23	2	11
4	2	1	2
5		0	

Beispiel 2: Der ggT aus 143 und 117 ist 1.

Erweiterter euklidischer Algorithmus

Der erweiterte euklidische Algorithmus ergänzt den euklidischen Algorithmus, indem sich hierbei neben dem ggT von zwei Zahlen a und b auch noch zwei weitere Zahlen x und y berechnen lassen, für die $a \cdot x + b \cdot y = ggT(a, b)$ mit $x, y \in \mathbb{Z}$ gilt.

Beim erweiterten euklidischen Algorithmus werden die beiden Zahlen a und b (im Beispiel $a = 60$ und $b = 13$) in der Spalte r untereinander geschrieben. Anschließend wird a durch b geteilt und das ganzzahlige Ergebnis in die Spalte q geschrieben. Der verbleibende Rest wird in die nächste Zeile der Spalte r geschrieben.

In dem Beispiel sieht das so aus:

1. Man rechnet zuerst $\frac{60}{13} = 4$ Rest 8.
2. Die 4 wird in Spalte q geschrieben und die 8 unter die 13.
3. Das wiederholt man mit allen Zahlen in Spalte r, bis dort eine 0 steht. Das r in der Zeile über der 0 ist $ggT(a, b)$. Nun wollen wir aber x und y berechnen.
4. Für Zeilen in den Spalten x und y schauen wir, welche Zahlen wir in die Gleichung einsetzen müssen, damit die ggT-Linearkombination erfüllt ist. Dabei geht man wie folgt vor: Die zweite Zahl der Spalte x wird mit der ersten Zahl der Spalte q (Zeile 2) multipliziert, und das Ergebnis wird von der ersten Zahl der Spalte x abgezogen, im Beispiel also $1 - (0 \cdot 4) = 1$.
5. Das Ergebnis wird in die nächste Zeile geschrieben.
6. Der Vorgang wird bis zur letzten Zahl der Spalte q fortgeführt.
7. Die Spalte y wird analog zur Spalte x berechnet.
8. Die untersten Zahlen der Spalten x und y erfüllen die ggT-Linearkombination. Es gilt: $60 \cdot 5 + 13 \cdot (-23) = 1 = \text{ggT}(60, 13)$

Schritt	r	q	x	y
1	60	-	1	0
2	13	4	0	1
3	8	1	1	-4
4	5	1	-1	5
5	3	1	2	-9
6	2	1	-3	14
7	1	2	5	-23
8	0	-	-	-

Erweiterter euklidischer Algorithmus

Modulo

$$5 : 3 = 1 \text{ Rest } 2$$

Dieses Teilen mit Rest, mit dem viele von uns das Dividieren lernten, führen wir beim Modulorechnen weiter. Nur interessiert uns beim Rechnen mit Divisionsrest nur der übriggebliebene Rest (im obigen Beispiel die 2). Mathematisch geschrieben bedeutet dies: 5 (mod 3) ist kongruent zu 2.

Ein alltägliches Beispiel für eine Modulo-Rechnung ist die Uhr: 17 Uhr = 5 Uhr. Hierbei rechnen wir genau genommen $17 \equiv 5 \pmod{12}$.

Weiteres Beispiel:

$$15 \pmod 7 \equiv 1$$

Modulo-Inverse

Die modulo-inverse Zahl a^{-1} zu a zur Basis n ist so beschaffen, dass

$$a^{-1} \cdot a \equiv 1 \pmod n$$

gilt. Die Inverse a^{-1} existiert nur dann, wenn $\text{ggT}(a, n) = 1$, also a und n teilerfremd sind. a^{-1} lässt sich mithilfe des erweiterten euklidischen Algorithmus bestimmen: Dazu findet man die Linearkombination von $\text{ggT}(a, n) = 1$. Stellt man diese anschließend um, erhält man

$$a \cdot x = y \cdot n + 1$$

nimmt man das ganze nun $\pmod n$, erhält man

$$a \cdot x \equiv y \cdot n + 1 \equiv 1 \pmod n$$

Also muss x die gesuchte inverse Zahl a^{-1} sein.

Schnelle Exponentiation

Für kleine Exponenten ist Potenzieren mit Modulo noch kein Problem. Da wir bei der RSA-Verschlüsselung jedoch gezwungen sind, mit großen Exponenten und Basen zu rechnen, müssen wir unsere Exponentiationsmethode effizienter machen. Dies erklären wir an einem Beispiel:

Für die Rechnung 31^{345} müssten wir bereits 345 Multiplikationen ausführen. Später benötigen wir jedoch Exponenten mit mehr als 100 Dezimalstellen, was unsere Verschlüsselung zu langsam machen würde.

Daher nutzen wir ein Verfahren zur schnellen Exponentation Modulo n. Dabei teilt der Computer große Exponenten in Zweierpotenzen auf, zum Beispiel:

$$31^{345} = 31^{256} \cdot 31^{64} \cdot 31^{16} \cdot 31^{8} \cdot 31$$

Dann quadriert der Computer die Zahl entsprechend oft und rechnet Modulo n. Die Ergebnisse werden in Modulo miteinander multipliziert, um auf das Endergebnis zu kommen.

Das hat den großen Vorteil, dass der Computer nur mehrfach quadrieren muss und dann die entsprechenden Zwischenergebnisse miteinander multipliziert. Eine Quadrierung entspricht nur einer einzigen Multiplikation und einer Modulo-Division und ist deutlich effizienter. Unsere Verschlüsselung läuft deutlich schneller.

Als Veranschaulichung wird $31^{345} \pmod{1000}$ berechnet:

$$\begin{aligned}
31^{2} &\equiv 961 \pmod{1000} \\
31^{4} \equiv (31^{2})^{2} &\equiv 961^{2} \equiv 521 \pmod{1000} \\
31^{8} \equiv (31^{4})^{2} &\equiv 521^{2} \equiv 441 \pmod{1000} \\
31^{16} \equiv (31^{8})^{2} &\equiv 441^{2} \equiv 481 \pmod{1000} \\
31^{32} \equiv (31^{16})^{2} &\equiv 481^{2} \equiv 361 \pmod{1000} \\
31^{64} \equiv (31^{32})^{2} &\equiv 361^{2} \equiv 321 \pmod{1000} \\
31^{128} \equiv (31^{64})^{2} &\equiv 321^{2} \equiv 041 \pmod{1000} \\
31^{256} \equiv (31^{128})^{2} &\equiv 041^{2} \equiv 681 \pmod{1000}
\end{aligned}$$

$$\begin{aligned}
31^{345} &\equiv 31^{256} \cdot 31^{64} \cdot 31^{16} \cdot 31^{8} \cdot 31^{1} \\
&\equiv 681 \cdot 321 \cdot 481 \cdot 441 \cdot 31 \\
&\equiv 81 \cdot 441 \cdot 31 \\
&\equiv 351 \pmod{1000}
\end{aligned}$$

In unserem Beispiel müssen statt der 345 Multiplikationen nur 12 Multiplikationen und Modulorechnungen ausgeführt werden. Selbst die Berechnung von $x^{1000000000}$ benötigt mit dieser Methode maximal 64 Schritte (statt 1 Milliarde).

Die Euler'sche φ-Funktion

Die Euler'sche φ-Funktion gibt die Anzahl der zu n teilerfremden Zahlen x an mit $x < n$ an. Die Schreibweise ist: $\varphi(n) = x$.

Beispielsweise gilt: $\varphi(9) = 6$, weil

x	ggT$(9, x)$
1	1
2	1
3	3
4	1
5	1
6	3
7	1
8	1

6 Zahlen x mit $x < 9$ erfüllen also die Bedingung ggT$(9, x) = 1$. Teilerfremd zu 9 sind demnach: 1, 2, 4, 5, 7 und 8.

Einige weitere Beispiele:

n	$\varphi(n)$	Zu n teilerfremde Zahlen
9	6	1, 2, 4, 5, 7, 8 (6 Stück)
10	4	1, 3, 7, 9 (4 Stück)
11	10	1, 2, 3, ..., 10 (10 Stück)
12	4	1, 5, 7, 11 (4 Stück)
13	12	1, 2, 3, ..., 12 (12 Stück)
14	6	1, 3, 5, 9, 11, 13 (6 Stück)
...	...	...

Wenn man die Euler'sche φ-Funktion einer Primzahl p bestimmt, gilt: $\varphi(p) = p-1$, da eine Primzahl nur die 1 und sich selbst als Teiler hat und somit alle Zahlen z mit $z < p$, teilerfremd zu p sind.

Es gilt: $\varphi(ab) = \varphi(a) \cdot \varphi(b)$, wenn a und b teilerfremd sind.

Der Satz von Euler

Der Satz von Euler besagt, dass $a^{\varphi(n)} \equiv 1 \pmod{n}$, wenn a und n teilerfremd sind.
Beweis: Sei R die Menge aller natürlichen Zahlen kleiner n, die teilerfremd zu n sind:

$$R = \{r_1, r_2, \cdots, r_{\varphi(n)}\}$$

Dabei enthält r genau $\varphi(n)$ Elemente (siehe Definiton φ-Funktion).

Sei a nun eine beliebige ganze Zahl, die teilerfremd zu n ist. Dann sei A die Menge

$$A = \{r_1 \cdot a, r_2 \cdot a, \cdots, r_{\varphi(n)} \cdot a\}$$

Da $r_i \cdot a \equiv r_j \cdot a \pmod{n}$ nur dann gilt, wenn $r_i \equiv r_j \pmod{n}$ ist, und alle Elemente aus R paarweise verschieden sind (kein Element ist doppelt), müssen alle Elemente aus A ebenfalls paarweise verschieden sein. Weil nun a und alle Elemente aus R teilerfremd zu n sind, müssen auch alle Elemente in A teilerfremd zu n sein. Das heißt, dass in A genau $\varphi(n)$ zu n teilerfremde Zahlen enthalten sind, die alle paarweise verschieden sind. Somit enthalten A und R dieselben Elemente! Es ist also

$$\Pi_{i=1}^{\varphi(n)} r_i \equiv \Pi_{i=1}^{\varphi(n)} r_i \cdot a \equiv a^{\varphi(n)} \Pi_{i=1}^{\varphi(n)} r_i \pmod{n}$$

beziehungsweise $a^{\varphi(n)} \equiv 1 \pmod{n}$

Primzahltests

Wie findet man heraus, ob eine Zahl prim ist?

Ein möglicher Primzahltest ist der Fermat-Test, der auf dem kleinen Satz von Fermat beruht. Der kleine Satz des Fermat lautet

$$a^{p-1} \equiv 1 \pmod{p} \text{ mit } p \in \mathbb{P},\ \text{ggT}(a, p) = 1$$

und folgt direkt aus dem Satz von Euler. Dieser besagt bekanntlich $a^{\varphi(n)} \equiv 1 \pmod{n}$. Falls p nun prim ist, ist $\varphi(p) = p - 1$. Setzen wir nun ein, erhalten wir direkt den kleinen Satz des Fermat.

Um eine Zahl zu testen, kann man im Umkehrschluss einen Zeugen z wählen, der teilerfremd zu der Zahl x ist, die diesem Test unterzogen werden soll. Anschließend prüft man, ob

$$z^{x-1} \equiv 1 \pmod{x}$$

erfüllt ist. Erfüllen x und z diese Beziehung nicht, ist x auf jeden Fall nicht prim, ansonsten kann x prim sein, muss es aber nicht. Denn nicht alle Zahlen, die die Voraussetzung erfüllen, sind prim, aber alle Zahlen, die die Voraussetzung nicht erfüllen, sind nicht prim.

Beispiel: Testen wir nun mithilfe des Fermat-Test, ob 4 eine Primzahl ist. Wählen wir als Zeugen $z = 3$, erhalten wir

$$3^3 \equiv 27 \equiv 3 \pmod{4}.$$

Da 3 nicht kongruent zu 1 (mod 4) ist, kann 4 nicht prim sein.

Zahlen, die für mindestens einen Zeugen den Fermat-Test bestehen, heißen Pseudoprimzahlen. Es gibt jedoch auch besondere Pseudoprimzahlen, sogenannte Carmichaelzahlen. Diese bestehen den Fermattest für *jeden* Zeugen. Die kleinste Carmichaelzahl ist die $561 = 3 \cdot 11 \cdot 17$. Diese Carmichaelzahlen stellen ein Problem dar, weil mit dem Fermat-Test nicht mit Sicherheit festgestellt werden kann, ob es sich bei einer Zahl um eine Primzahl handelt – selbst wenn man alle möglichen Zeugen testet.

Wir benötigen also einen weiteren Primzahltest: den Miller-Rabin-Test.

Um den Test durchzuführen, schreiben wir unsere zu testende Zahl x als

$$x - 1 = 2^s t \text{ mit } t \text{ ungerade}$$

Nun wählen wir einen Zeugen z mit $\mathrm{ggT}(x, z) = 1$. Ist x nun prim, ist eine der folgenden Bedingungen erfüllt: (a) entweder $z^t \equiv 1 \pmod{x}$, (b) oder $z^{2^i t} \equiv -1 \pmod{x}$

Falls keine der beiden Bedingungen zutrifft, ist die zu testende Zahl mit Sicherheit nicht prim. Mit dem Miller-Rabin-Test existieren keine starken Pseudoprimzahlen oder Ähnliches mehr, sodass dieser Test für alle Zahlen geeignet ist.

Allerdings gibt der Miller-Rabin-Test seine Ergebnisse für einen einzelnen Zeugen mit einer Fehlerquote von ca. 25 % an. Darum führen wir den Test mehrmals hintereinander mit unterschiedlichen Zeugen durch, um eine Wahrscheinlichkeit von annähernd 100 % zu erhalten: Führen wir den Test beispielsweise 50 Mal durch, liegt die Wahrscheinlichkeit für ein falsches Ergebnis nur noch bei gerade mal $0.25^{50} \approx 7.9 \cdot 10^{-31} \approx 0.00000000000000000000000000079\,\%$. Will man das noch weiter reduzieren, erhöht man einfach die Testanzahl entsprechend.

RSA

Henriette, Jannik, Catharina, Moritz

Wir verwenden für unseren Chat die sichere RSA-Verschlüsselung. Sie wurde nach ihren drei Erfindern Rivest, Shamir und Adleman benannt. Die RSA-Verschlüsselung oder davon abgeleitete Verfahren finden in fast allen Bereichen der modernen Informationsübertragung Anwendung.

Schlüsselerzeugung

Bei der RSA-Verschlüsselung handelt es sich um eine asymmetrische Verschlüsselung. Es werden also zwei verschiedene Schlüssel zur Ver- und Entschlüsselung einer Nachricht benötigt. Dabei handelt es sich um einen öffentlichen Schlüssel, der zur Verschlüsselung dient und einen privaten Schlüssel, den nur der Empfänger einer Nachricht kennt und der zum Entschlüsseln benötigt wird. Der Modulus n ist ebenfalls öffentlich und wird zum Ver- und Entschlüsseln benötigt.

Beide Schlüssel werden innerhalb des folgenden Vorgangs erzeugt:

1. Im ersten Schritt werden zwei Zufallszahlen p und q gewählt, die prim sein müssen. Zunächst wird geprüft, ob die Zufallszahlen bereits prim sind. Wenn dies nicht der Fall ist, werden die Zahlen so lange erhöht, bis die nächstgrößeren Primzahlen vorliegen. Die beiden Primzahlen p und q sollen die gleiche Größenordnung haben, aber nicht zu dicht beieinander liegen.
2. Anschließend wird das gemeinsame Produkt n, der sogenannte RSA-Modulus, aus den beiden Primzahlen gebildet:

   $$n = p \cdot q$$

3. Im dritten Schritt wird $\varphi(n)$ berechnet. Da n das Produkt aus den Primzahlen p und q ist, können wir $\varphi(n)$ wie folgt berechnen:

   $$\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$$

4. Danach wird eine zu φ (n) teilerfremde Zahl e gewählt, die größer als eins, aber kleiner als $\varphi(n)$ ist. Diese Wahl darf zufällig sein, kann aber auch fest gewählt werden.
5. Im letzten Schritt wird die Modulo-Inverse d von e bezüglich des Moduls

$\varphi(n)$ mithilfe des erweiterten euklidischen Algorithmus berechnet.

Es soll also die folgende Kongruenz gelten: $e \cdot d \equiv 1 \mod \varphi(n)$. Die Berechnung erfolgt mit dem erweiterten euklidischen Algorithmus (Seite 57). Dabei wird $a = e$, $x = d$, $b = \varphi(n)$ und $y = -k$. Damit ist die ggT-Linearkombination mit den Variablen der RSA-Verschlüsselung

$$e \cdot d - \varphi(n) \cdot k = ggT(e, \varphi(n))$$

Damit keine fremde Person den privaten Schlüssel d berechnen kann und somit niemand außer dem vorgesehenen Empfänger die Nachricht entschlüsseln kann, werden die Werte für p, q und $\varphi(n)$ nach dem Beenden der Schlüsselerzeugung gelöscht. Anderenfalls könnte man durch den gemeinsamen Modulus n und $\varphi(n)$ sehr leicht auf d schließen und ungewollten Zugriff auf fremde Nachrichten bekommen.

Beispiel zur RSA-Schlüsselerzeugung

1. Wir wählen $p = 8$ und $q = 12$. Die beiden Zahlen werden so lange erhöht, bis sie prim sind. In diesem Fall ist das bei $p = 11$ und $q = 13$ der Fall. Jetzt sind sowohl p als auch q Primzahlen.
2. Hier ist $n = 143$, da $n = p \cdot q = 11 \cdot 13 = 143$
3. In diesem Beispiel ist $\varphi(n) = 120$, da $\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1) = (11-1) \cdot (13-1) = 120$
4. Beispielsweise kann für $e = 17$ gewählt werden, da $e = 17$ und $\varphi(n) = 120$ teilerfremd sind.
5. d wird nun mit dem erweiterten euklidischen Algorithmus berechnet. Hier ist $d = 113$ (Seite 57)
6. Als letzter Schritt werden $p = 11$, $q = 13$ und $\varphi(n) = 120$ vernichtet, damit nicht auf $d = 113$ zurückgeschlossen werden kann.

Somit ergeben sich in diesem Beispiel die folgenden beiden Schlüssel:

- öffentlicher Schlüssel: $n = 143, e = 17$
- privater Schlüssel: $n = 143, e = 113$

Ver- und Endschlüsselung

Der öffentliche Schlüssel e kann bedenkenlos über öffentlich Kanäle an den Sender geschickt. Der private Schlüssel d bleibt beim Empfänger und darf unter keinen Umständen weitergegeben werden.

Jetzt wird unsere Nachricht m mit dem öffentlichen Schlüssel des Empfängers vom Sender verschlüsselt:

$$m^e \equiv c \pmod{n}$$

Damit erhält der Sender die Chiffre c. Diese kann problemlos an den Empfänger gesendet werden, da sie nur mit dem privaten Schlüssel wieder in die Nachricht übersetzt werden kann. Der Empfänger entschlüsselt mit seinem privaten Schlüssel die Nachricht folgendermaßen:

$$c^d \equiv m \pmod{n}$$

Der Empfänger erhält somit die Nachricht m.

Beispiel mit Schlüsseln aus der Schlüsselerzeugung

$m = 3$

Verschlüsseln:

$3^{17} \equiv 9 \pmod{143}$

Entschlüsseln:

$9^{113} \equiv 3 \pmod{143}$

Beweis

Im Folgenden zeigen wir, warum $c^d \equiv m \pmod{n}$ gilt. Laut dem Satz von Euler gilt:

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

Wir haben d mit Hilfe des erweiterten Euklidischen Algorithmus (Seite 57) so berechnet, dass folgendes gilt:

$$1 = e \cdot d - k \cdot \varphi(n)$$

also ist

$$e \cdot d = 1 + k \cdot \varphi(n)$$

Anders ausgedrückt: d ist die Modulo-Inverse zu $e \pmod{\varphi(n)}$.

Für beliebige $m, n, k \in \mathbb{N}$, mit $\text{ggT}(m, n) = 1$ (also teilerfremden m und n), gilt:

$$\begin{aligned} & m^{\varphi(n)\cdot k+1} \pmod n \\ \equiv & m^{\varphi(n)\cdot k} \cdot m^1 \pmod n \\ \equiv & (m^{\varphi(n)})^k \cdot m^1 \pmod n \\ \equiv & 1^k \cdot m \pmod n \\ \equiv & m \pmod n \end{aligned}$$

In Kurzform gilt also

$$m^{e\cdot d} \equiv m^{\varphi(n)\cdot k+1} \equiv m \pmod n.$$

Damit haben wir gezeigt, dass man seine ursprüngliche Nachricht wieder erhält, wenn man sie erst mit e und anschließend mit d potenziert (modulo n). Es ist auch möglich, die Zahlen e und d zu vertauschen: mit e verschlüsseln und anschließend wieder mit d entschlüsseln.

Sicherheit

Die RSA-Verschlüsselung ist nur so lange sicher, wie die beiden Primzahlen p und q unbekannt sind. Anderenfalls kann man alle oben genannte Schritte der Schlüsselerzeugung (Seite 60) durchführen, um den privaten Schlüssel d zu bestimmen. Ungünstige Beispiele für p und q sind alle Primzahlen, bei deren Produkt nur wenige Kombinationen von zwei Primzahlen in Frage kommen – also sind zu kleine Zahlen schlecht. Damit das Bestimmen von p und q mit Hilfe der Primfaktorzerlegung nach heutigem Stand auch mit sehr leistungsfähigen Computern praktisch nicht zu erreichen ist, benötigt man sehr große Primzahlen. Es muss außerdem gelten: $p \neq q$, da man sonst einfach die Wurzel aus n ziehen könnte und somit sowohl p als auch q berechnet hätte.

Obwohl es in der Vergangenheit bereits oft gelang, Zahlen mit mehreren hundert Dezimalstellen zu faktorisieren, stellt die wachsende Rechenleistung moderner Computer für die RSA-Verschlüsselung bisher kein Problem dar. Die Bundesnetzagentur empfiehlt bis Ende 2020 Schlüssel mit einer Länge von 2048 Bit, was einer Zahl mit 617 Dezimalstellen entspricht. Damit müssen p und q je etwa 1024 Bit oder 309 Dezimalstellen lang sein.

Die zwei wesentlichen Probleme der Kryptoanalyse der RSA-Verschlüsselung sind:

1. RSA-Problem: Gegeben sind der öffentliche Schlüssel (n, e) sowie der Geheimtext c. Gesucht wird der Klartext m, wobei gilt:

   $$m^e \equiv c \pmod n.$$

   Das Problem liegt hier in der Schwierigkeit, e-te Wurzeln modulo n zu ziehen, was zur Bestimmung der Nachricht m notwendig ist.

2. RSA-Schlüsselproblem: Gegeben ist der öffentliche Schlüssel (n, e). Gesucht wird der geheime Schlüssel d wobei gilt:

   $$e \cdot d \equiv 1 \pmod{\varphi(n)}.$$

   Das Problem liegt hier in der Schwierigkeit, die Eulersche φ-Funktion von n ohne Kenntnis der Faktoren p und q zu berechnen.

Solange diese Probleme ungelöst sind, ist die RSA-Verschlüsselung sicher.

Programmiergrundlagen

Lisa, Timon, Vilmos, Moritz, Anna-Lena

Neben den mathematischen Grundlagen benötigten wir auch einige Programmiergrundlagen, um unseren Chat zu programmieren. Da wir den Chat als Website mit Angular programmiert haben, benötigten wir drei Programmiersprachen, die wir hier kurz vorstellen: HTML, CSS und TypeScript.

HTML

HTML steht für Hypertext Markup Language. Es wird für die Strukturierung und den Inhalt von Websites genutzt. Damit kann man zum Beispiel Überschriften (headings), Textfelder (inputs) oder Knöpfe (buttons) erzeugen.

Doch wenn man einen Button drückt oder etwas in ein Textfeld eingibt, würde nichts passieren, denn dafür braucht man eine andere Sprache, nämlich TypeScript. Mit dem Befehl [ngModel] verbinden wir HTML mit TypeScript. Gleichzeitig geben wir den Eingabefeldern Namen, damit wir auch sie in TypeScript verwenden können. Um Elemente in CSS zu bearbeiten, müssen wir beispielsweise den Knopf (button) mit id=„beispiel“ versehen.

Ein simpler Beispielcode:

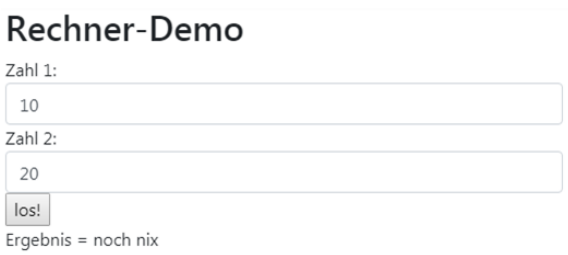
```
<h2>Rechner-Demo</h2>

Zahl 1:
<input name="zahl1"
    [(ngModel)]="zahl1" />

Zahl 2:
<input name="zahl2"
    [(ngModel)]="zahl2" />

<button id="beispiel"
    (click)="rechne()">los!
</button>
<br />

Ergebnis = {{ergebnis}}
```



Website zum HTML-Quellcode

CSS

CSS steht für Cascading Style Sheets. Es wird benutzt, um die Websites zu designen. Beispielsweise kann man die Hintergrundfarbe (background-color) oder die Textfarbe (color) und Textart (font-family) ändern. Zudem kann man Objekte wie Bilder oder Knöpfe transformieren (zum Beispiel drehen „transform: rotate“). Indem wir einzelnen Elementen in HTML IDs gegeben haben (id=„beispiel“), können wir sie mit CSS verändern.

Ein Beispielcode:

```
body
{
    background-color: hotpink;
    color: white;
    font-family: Verdana;
}
#beispiel
{
    color: hotpink;
    transform: rotate(-15deg);
}
```

CSS-Beispiel

TypeScript

Mit HTML und CSS wird das Grundgerüst einer Website gebaut. Für die Interaktion mit dem Nutzer ist TypeScript verantwortlich. Hierfür werden Inputs (Textfelder) und Buttons mit TypeScript verknüpft. So kann man beispielsweise wie im untenstehenden Code zwei Zahlen addieren. Hierbei erstellen wir jeweils eine Variable, welche den HTML Eingabefeldern zugeordnet ist. Wenn wir nun den Knopf drücken, wird die Funktion „rechne()“ ausgeführt und die beiden Zahlen werden addiert. Am Schluss lassen wir uns noch das Ergebnis über die Variable *ergebnis* in HTML ausgeben.

```
ergebnis = 'noch nix';
zahl1 = 10;
zahl2 = 20;

rechne() {
    this.ergebnis =
        this.zahl1 + this.zahl2;
}
```

BigInteger

Normalerweise rundet TypeScript große Zahlen ab einer bestimmten Größe. Da unsere RSA-Verschlüsselung aber nur mit sehr großen Zahlen sicher ist, nutzen wir einen speziellen Datentyp *BigInteger*. Er lässt uns auch mit großen Zahlen rechnen, die zum Teil 100 oder mehr Stellen haben, ohne sie zu runden oder zu verändern. TypeScript läuft meistens im Hintergrund, sodass man von all dem nichts mitbekommt.

ASCII/Unicode

Da die RSA-Verschlüsselung nur mit Zahlen funktioniert, muss ein eingegebener Text in Zahlen umgewandelt werden. Dafür gibt es verschiedene Umrechnungstabellen, zum Beispiel ASCII und Unicode, die verschiedenen Zeichen (Zahlen, Buchstaben, Emojis etc.) jeweils einen Zahlencode zuordnen. So ist der Code für das Zeichen „A“ 65, für „B“ 66, sowie für „¤“ 226 130 172. Mit diesen Zahlencodes kann ein Text dann auch verschlüsselt werden.

Text-Zu-Zahl / Zahl-Zu-Text

Möchte man beispielsweise das Wort *Hallo* versenden, so muss dieses zunächst in eine Zahl umgewandelt werden. Dabei wird jedem Buchstaben ein bestimmter, zweistelliger Wert zugeordnet, der in der Unicodetabelle niedergeschrieben ist. Ist die Umwandlung der Buchstaben in die entsprechenden Codes erfolgt, so werden diese nicht addiert, da man sonst einen unbrauchbaren Wert bekommen würde, den man nicht in das ursprüngliche Wort zurück umwandeln könnte.

Stattdessen werden die Codes in der Reihenfolge der Buchstaben des jeweiligen Wortes aneinandergehängt. Damit der Computer weiß, an welcher Stelle später die einzelnen Codes wieder voneinander getrennt werden müssen, erfolgt die Aneinanderreihung durch Multiplikation mit der Hunderterpotenz der Stelle, an der sie sich befinden. Da die erste Stelle die 0-te Position einnimmt, muss von dem Exponenten die Zahl 1 subtrahiert werden. Im folgenden Beispiel wird das Wort „Hallo“ folgendermaßen codiert:

Beispiel: Umwandlung von Text zu Zahl

$$72 \cdot 100^4 + 65 \cdot 100^3 + 76 \cdot 100^2 \cdot + 76 \cdot 100^1 + 79 \cdot 100^0$$

Um den Code anschließend wieder in das ursprüngliche Wort, in unserem Fall „Hallo“ zu decodieren, muss man diese Potenzierung rückgängig machen. Dies geschieht dadurch, dass man den hintersten zweistelligen Code mod 100 rechnet und danach durch hundert teilt, damit der nächste Code an die hinterste Stelle rückt. Hier fallen die entstandenen Kommazahlen weg, es wird auch nicht gerundet. Damit werden alle Codes nacheinander wieder voneinander getrennt, allerdings sind sie in umgedrehter Reihenfolge angeordnet. Nun muss jedem Code der entsprechende Buchstabe zugeordnet werden und die entstehende Buchstabenfolge umgedreht werden.

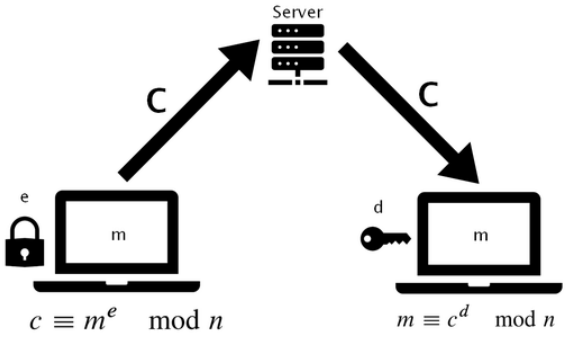
$$
\begin{aligned}
&7265767679 \pmod{100} = 79\\
&7265767679/100 = 72657676\\
&72657676 \pmod{100} = 76\\
&72657676/100 = 726576\\
&726576 \pmod{100} = 76\\
&726576/100 = 7265\\
&7265 \pmod{100} = 65\\
&7265/100 = 72\\
&72 \pmod{100} = 72
\end{aligned}
$$

Der umgedrehte, decodierte Code ergibt nun wieder das Wort „HALLO“.

Server-Kommunikation

Damit nicht beide Chatpartner gleichzeitig online sein müssen und man sich auch von unterschiedlichen Geräten in den Chat einloggen kann, versenden wir unsere Nachrichten nicht direkt von PC zu PC. Stattdessen verwenden wir einen Server. Auf dem Server werden die gesamten Daten sicher gespeichert, damit wir von überall auf unseren Chat zugreifen können.

Die gesendeten Nachrichten werden so lange auf dem Server gespeichert, bis sich der Empfänger erneut einloggt und die Nachrichten liest. Die gesamte Ver- und Entschlüsselung findet nur auf dem jeweiligen Endgerät statt, somit erhält der Server nur verschlüsselte Nachrichten vom Sender. Auf dem Server werden nur verschlüsselte Daten gespeichert, so können wir uns sicher sein, dass niemand unsere Daten lesen kann. Sind wir online, fragen wir den Server alle zwei Sekunden nach neuen Nachrichten.



Serverkommunikation

Chat

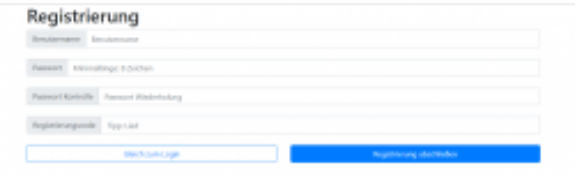
Isabel, Valentin, Timon, Nico, Tim

Nachdem nun die Grundlagen bekannt sind, können wir endlich mit der Programmierung des Chats beginnen.

Registrierung

Damit man überhaupt chatten kann, muss man sich zunächst bei der Registrierung ein Profil anlegen.

Dafür legt der Benutzer sich zuerst einen Benutzernamen mit einer Länge zwischen 3 und 22 Zeichen an und gibt dann ein Passwort ein,



Registrierungsseite

das mehr als 8 Zeichen umfassen muss. Damit das Passwort auf jeden Fall richtig ist, muss das Passwort in einem weiteren Feld wiederholt werden. Zudem wollten wir sicherstellen, dass sich nur Teilnehmerinnen und Teilnehmer der Akademie registrieren können, indem wir ein Registrierungswort festgelegt haben.

Drückt man nun den „Registrierung abschließen“-Button, wird zunächst überprüft, ob das Registrierungswort richtig ist. Ist dies nicht der Fall, wird direkt unter dem Eingabefeld die Meldung „Fehler“ angezeigt. So ist es auch bei dem Passwort, das mit dem wiederholten Passwort verglichen wird: Stimmen die beiden Eingaben der Passwörter nicht überein, wird unter dem Passwortfeld die Meldung „Fehler“ angezeigt.

Gibt es keine Fehler, findet die RSA-Schlüsselerzeugung (Seite 60) statt.

Dafür erstellen wir erst einmal p und q, zwei 160-stellige, zufällige Primzahlen, indem wir eine Zufallszahl zwischen 0 und 1 durch den TypeScript-Befehl Math.Random erzeugen. Diese Zahl hat 16 Nachkommastellen. Damit wir eine natürliche Zahl erhalten, multiplizieren wir diese mit 10^{16}.

Damit die Zahl überhaupt größer wird, multiplizieren wir die Zahl erneut mit 10^{16} und addieren eine zweite zufällige Zahl dazu.

Nach zehnmaligem Wiederholen des Vorgangs erhalten wir so eine 160-stellige Zahl. Die Zahl, die dabei entsteht, ist aber noch nicht automatisch eine Primzahl, weshalb wir mit verschiedenen Primzahltests (Seite 56) die nächstgrößere Primzahl bestimmen.

Dieser Vorgang wird zweimal durchgeführt: einmal für p und einmal für q. Ein Problem in der Zufälligkeit unserer Primzahlen ist, dass Primzahlen, bei denen der Abstand zur vorherigen Primzahl größer ist, eine höhere Wahrscheinlichkeit besitzen, ausgewählt zu werden.

Dies fällt bei der möglichen Anzahl an Primzahlen aber kaum ins Gewicht. So würde ein normaler Computer immer noch länger als die Lebensspanne des Universums durchprobieren müssen, um von den anderen Zahlen der RSA-Verschlüsselung auf unser p und q zu schließen.

Dass p und q gleich sind, ist aufgrund der hohen Anzahl der möglichen Primzahlen so unwahrscheinlich, dass wir diesen Fall nicht weiter beachtet haben.

Wir wollen nicht an einen bestimmten Rechner für die Verwendung unseres Chats gebunden sein, weshalb wir Zugriff auf unseren privaten Schlüssel d von anderen Rechnern brauchen. Damit wir unser d wieder von einem anderen Rechner abrufen können, verschlüsseln wir d mithilfe der symmetrischen (Seite 55), sicheren AES-Verschlüsselung und unserem Passwort als Schlüssel. Das verschlüsselte d wird nun an den Server geschickt und dort gespeichert.

Wir hashen (Seite 66) unser Passwort. Das tun wir, damit wir den Passworthash an den Server senden können. So wird der Hashwert bei der Anmeldung von einem anderen Rechner abgeglichen. Ist dieser gleich, kann d mithilfe des nicht gehashten Passworts entschlüsselt werden. Gleichzeitig werden der Benutzername, sowie die Schlüssel e, n und das verschlüsselte d an den Server weitergeleitet.

Falls der Benutzer schon registriert ist, kann er über den Login-Button zum Login gelangen.

Hash-Funktionen

Eine Hash-Funktion ist ein Algorithmus, der für eine Eingabe beliebiger Länge eine Ausgabe fester Länge berechnet. Dabei sind von der Ausgabe keine Rückschlüsse auf die Eingabe zu machen. Eine Hash-Funktion ist somit keine Verschlüsselung, sondern eine sogenannte Einweg-Funktion. Eine für uns sehr wichtige Eigenschaft ist, dass für die gleiche Eingabe immer die gleiche Ausgabe (der gleiche Hash) erzeugt wird.

Hash-Funktionen können beispielsweise zum Speichern von Passwörtern auf einem Server verwendet werden. Dabei wird nicht direkt das Passwort gespeichert, sondern nur der Hash des Passworts. So kann kein Hacker auf dem Server die Klartext-Passwörter lesen. Beim Login wird das eingegebene Passwort wieder in den Hash-Wert umgerechnet und mit dem auf dem Server gespeicherten, gehashten Passwort vergleichen.

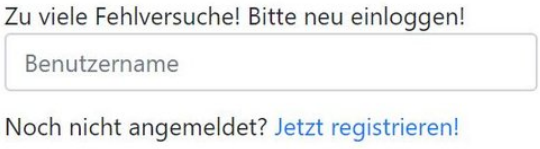
Beispiel: Aus der Eingabe *Passwort* wird durch den MD5-Hash-Algorithmus die Ausgabe *3e45af4ca27ea2b03fc6183af40ea112*

Login

Wenn man unseren Chat (www.sabwchat.eu) aufruft, wird man zum Login weitergeleitet. Hier muss man seine Anmeldedaten eingeben.

Login-Seite

Wenn man noch keinen Account hat, ist hier auch der Link zur Registrierung hinterlegt. Die eigentliche Anmeldung geschieht in Schritten, aufgeteilt auf zwei Seiten: Auf der einen gibt man seinen Benutzernamen an, auf der anderen sein Passwort. Sobald man auf der zweiten Seite den Button „Login“ anklickt, geschehen im Hintergrund mehrere Dinge: Zuerst wird das Passwort gehasht und zum Server weitergeleitet. Hier wird der Benutzername und der Passworthash mit den Anmeldedaten, die bereits bei der Registrierung auf dem Server abgespeichert wurden, abgeglichen. Bei erfolgreichem

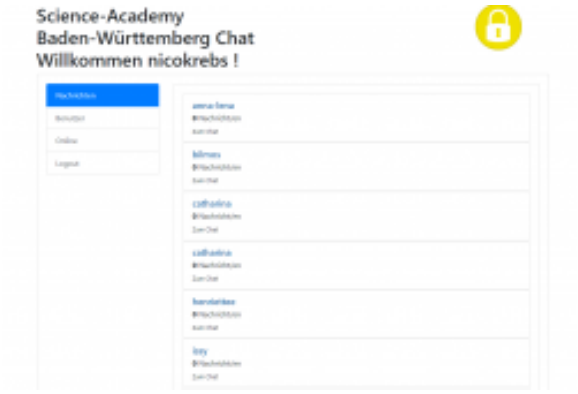


Fehlermeldung

Abgleich mit den Daten auf dem Server wird noch das beim Login mit AES verschlüsselte d entschlüsselt, danach ist man erfolgreich angemeldet.

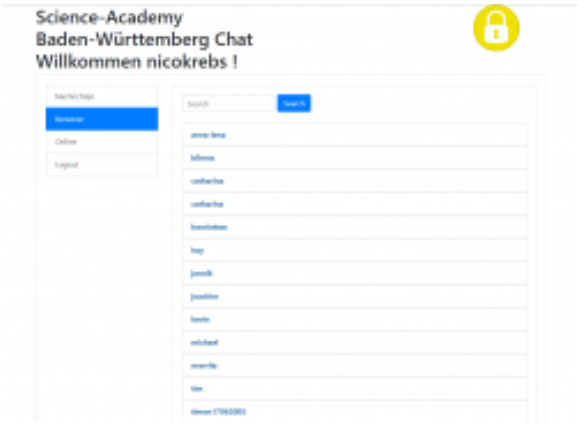
Benutzerliste

Vom Login wird man zur Benutzerliste weitergeleitet. Diese ist so aufgebaut, dass es links ein Auswahlmenü gibt, mit dem man die verschiedenen Funktionen der Benutzerliste aufrufen kann. Darüber wird man mit einem „Willkommen“ begrüßt, bei dem der angemeldete Benutzername eingefügt ist. Die erste Auswahlmöglichkeit ist die Nachrichtenliste, bei der dem Benutzer angezeigt wird, wie viele neue Nachrichten er von wem bekommen hat.



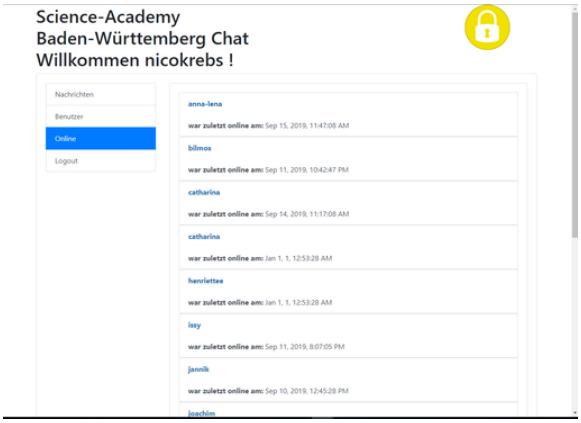
Chatübersicht

Klickt man auf die jeweiligen Namen, so wird man zum Einzelchat mit dieser Person geführt. Der nächste Eintrag im Menü mit der Aufschrift *Benutzer* öffnet eine Liste mit allen registrierten Benutzern. Direkt darüber befindet



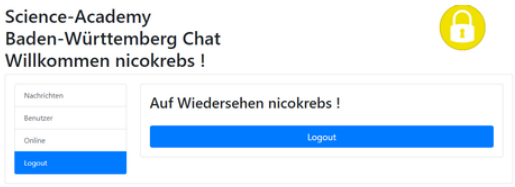
Übersicht der Benutzer

sich ein Suchfeld, in welches man den Teil eines gesuchten Benutzernamens eingeben kann. Die Benutzernamen, welche diese Buchstabenkombination enthalten, werden darunter angezeigt. Die nächste Auswahlmöglichkeit zeigt an, wann die anderen Benutzer zuletzt online waren, oder ob sie gerade online sind. Der unterste Eintrag



Übersicht der Nutzung

bringt den angemeldeten Benutzer zum Logout. Hierbei werden alle wichtigen Benutzerinforma-



Logout

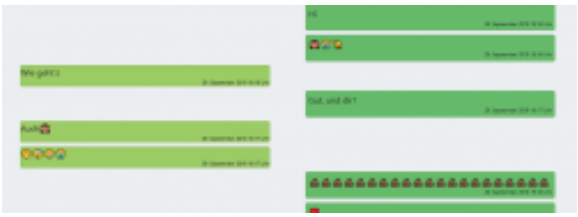
tionen wie Benutzername, Passworthash und die Schlüssel d und n vom Computer gelöscht. Diese Funktion ist nötig, damit niemand über den gleichen PC auf diese Informationen zugreifen kann. Sie werden beim erneuten Einloggen wieder aufgerufen.

Einzelchat

Wenn man eingeloggt ist und einen Chat ausgewählt hat, wird man in einen Bereich weitergeleitet, in dem man mit der ausgewählten Person in einem Einzelchat chatten kann. Ganz oben links in diesem Bereich sieht man unser Kurslogo. Wenn man mit dem Curser der Maus über dieses fährt, wird der Cursor zu einem Zeigefinger, und wenn man dann auf das Logo klickt, gelangt man wieder zur Benutzerliste.

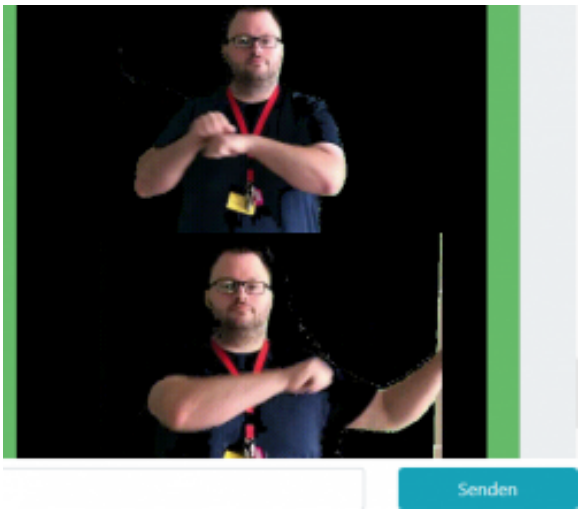
Kurslogo

Daneben wird angezeigt, mit wem gechattet wird. Darunter befindet sich der Bereich, in welchem die Nachrichten angezeigt werden. Dort werden Nachrichten, die man selbst und



Einzelchat

der Chatpartner schreibt, in unterschiedlichen Grüntönen angezeigt. Unter dem Chatfenster befinden sich ein Textfeld und ein Button. In das Textfeld gibt man die Nachricht ein, die man senden will. In diese Nachricht kann man auch mithilfe von Buttons unter dem Textfeld Emojis einfügen. Außerdem verfügt der Chat über eine GIF-Unterstützung, sodass man drei verschiedene GIFs versenden kann. Der Button



GIF von einem unserer Kursleiter

rechts vom Textfeld ist mit dem Text „Senden“ versehen. Wenn dieser Button gedrückt wird, wird der Text des Textfeldes mit Unicode in eine Zahl umgewandelt. Diese Zahl wird zunächst mit dem öffentlichen Schlüssel e des Partners verschlüsselt, zum Server geschickt und beim Partner mit dem privaten Schlüssel d wieder entschlüsselt. Zusätzlich wird die Nachricht jedoch auch mit dem eigenen e verschlüsselt und zum Server geschickt, um zu ermöglichen, dass man auf die geschriebenen Nachrichten auch von einem anderen Gerät aus zugreifen kann. Da die Website alle zwei Sekunden überprüft, ob es neue Nachrichten auf dem Server gibt, werden diese direkt bei sich und beim Partner angezeigt und es wird automatisch runtergescrollt.

Zusammenfassung

In unserem Kurs haben wir einen Chat programmiert. Dafür mussten wir uns zuerst sowohl mathematische als auch informatische Grundlagen aneignen. Zu den mathematischen Grundlagen gehörte die Rechnung mit Modulo, die φ-Funktion, Primzahlen und Primzahltests, der Satz von Euler, der euklidische und erweiterte euklidische Algorithmus, die Modulo-Inverse und der größte gemeinsame Teiler. Außerdem beschäftigten wir uns zur Einführung in Verschlüsselungsmethoden mit der Caesar-Verschlüsselung, dem Unterschied zwischen symmetrischer und asymmetrischer Verschlüsselung, den Hash-Funktionen und mit der RSA-Verschlüsselung, welche wir letztendlich für unseren Chat benutzten.

Zu den informatischen Grundlagen gehörte die Programmierung mit Angular, also mit HTML, CSS und TypeScript. Außerdem beschäftigten wir uns mit ASCII, Unicode, Text-zu-Zahl und Zahl-zu-Text. Für unseren Chat haben wir uns auch ein mit der Serverkommunikation beschäftigt, wobei Kevin uns an dieser Stelle sehr viel geholfen hat und den Server größtenteils einrichtete. Bei unserem Chat haben wir in Kleingruppen Registrierung, Login, Benutzerliste und Einzelchat programmiert.

Insgesamt hat uns der Kurs sehr gut gefallen, und gegen Ende der Akademie waren wir als Gruppe ein sehr eingespieltes Team, wir haben sogar unser Kursmaskottchen fest als 13. Kurs-

teilnehmer integriert. Wir sind alle sehr gut miteinander ausgekommen, und als Informatikkurs haben wir unsere Kursziele mit tatkräftiger Unterstützung von Kevin, Michael und Henriette gut umgesetzt.

Zitate und Kurssprüche

- „Das dauert noch ein Weilchen. Die lagen auf dem Physiktisch – kann ich die noch essen?“
- „KCB UX5!“
- „BigInt – Wir sind die Größten!“
- „Fluchtweg: Raus – Treppe runter!“
- „Musterbeutelklammern“
- „Das ist KONGRUENT, nicht gleich!“
- „didaktische Reduktion“
- „Wir haben da mal was vorbereitet . . . “
- „AMÖBE! AMÖBE!“
- „Mensch Kevin, was soll das?“
- „Määäh“ „Lamas mähen nicht! Sie spucken!“ „Ist es dir also lieber, wenn das Lama dich anspuckt?“
- „Habt ihr das verstanden?“ „Ähm . . . “
- „Du kannst nicht einfach andere Leute nerven, wenn du noch eine Aufgabe hast! Entweder du kümmerst dich um deine Aufgabe, oder du gibst sie weiter. Ansonsten kriegen wir es am Ende nicht zusammen!“
- „kleine Füße“
- „Ich werde mal so groß, wenn ich groß bin! (3m)“
- „Alle Elemente können PCs zerstören: Wasserschaden, Überhitzung, Steinschlag, vom Wind vom Tisch gepustet – die Möglichkeiten sind vielfältig“
- „Michael?“ „Was?“ „Nichts.“
- „Nieder mit UTF 16!“
- „Henriette, kannst du bitte nochmal das Buch holen? Es ist aus dem Fenster nach unten gefallen“
- „Das schaffen wir zeitlich nicht“

Danksagung

Wir möchten uns an dieser Stelle bei denjenigen herzlich bedanken, die die 17. JuniorAkademie Adelsheim / Science Academy Baden-Württemberg überhaupt möglich gemacht haben.

Finanziell wurde die Akademie in erster Linie durch die Stiftung Bildung und Jugend, die Schwarz-Stiftung, die Hopp-Foundation, den Förderverein der Science Academy sowie durch den Fonds der Chemischen Industrie unterstützt. Dafür möchten wir allen Unterstützern ganz herzlich danken.

Die Science Academy Baden-Württemberg ist ein Projekt des Regierungspräsidiums Karlsruhe, das im Auftrag des Ministeriums für Kultus, Jugend und Sport Baden-Württemberg für Jugendliche aus dem ganzen Bundesland realisiert wird. Wir danken daher Frau Anja Bauer, Abteilungspräsidentin der Abteilung 7 – Schule und Bildung des Regierungspräsidiums Karlsruhe, der Leiterin des Referats 75 – allgemein bildende Gymnasien, Frau Leitende Regierungsschuldirektorin Dagmar Ruder-Aichelin und Herrn Jan Wohlgemuth vom Ministerium für Kultus, Jugend und Sport Baden-Württemberg. Koordiniert und unterstützt werden die JuniorAkademien von der Bildung & Begabung gGmbH in Bonn, hier gilt unser Dank dem scheidenden Koordinator der Deutschen Schüler- und JuniorAkademien, Herrn Volker Brandt, seiner Nachfolgerin Ulrike Leithof, der Referentin für die Akademien Dorothea Brandt sowie dem gesamten Team.

Wie in jedem Jahr fanden die etwas über einhundert Gäste sowohl während des Eröffnungswochenendes und des Dokumentationswochenendes als auch während der zwei Wochen im Sommer eine liebevolle Rundumversorgung am Eckenberg-Gymnasium mit dem Landesschulzentrum für Umwelterziehung (LSZU) in Adelsheim. Stellvertretend für alle Mitarbeiterinnen und Mitarbeiter möchten wir uns für die Mühen, den freundlichen Empfang und den offenen Umgang mit allen bei dem zum Zeitpunkt des Drucks dieser Dokumentation schon ehemaligen Schulleiter des Eckenberg-Gymnasiums, Herrn Oberstudiendirektor Meinolf Stendebach, und seinem Nachfolger, Herrn Studiendirektor Martin Klaiber, besonders bedanken.

Ein herzliches Dankeschön geht auch an Frau Oberstudiendirektorin Dr. Andrea Merger vom Hölderlin-Gymnasium in Heidelberg, wo wir bei vielfältiger Gelegenheit zu Gast sein durften.

Zuletzt sind aber auch die Kurs- und KüA-Leiter gemeinsam mit den Schülermentoren und der Assistenz des Leitungsteams diejenigen, die mit ihrer hingebungsvollen Arbeit das Fundament der Akademie bilden.

Diejenigen aber, die die Akademie in jedem Jahr einzigartig werden lassen und die sie zum Leben erwecken, sind die Teilnehmerinnen und Teilnehmer. Deshalb möchten wir uns bei ihnen und ihren Eltern für ihr Engagement und Vertrauen ganz herzlich bedanken.

Bildnachweis

Seite 11, Abbildung Sonnenfinsternis-Schema:
https://commons.wikimedia.org/wiki/File:Sonnenfinsternis-schema.svg
Wikimedia-User Юкатан
CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

Alle anderen Abbildungen sind entweder gemeinfrei oder eigene Werke.