Pflichtenheft

Cryptographics

24. November 2013

Phase	**Verantwortlicher**	**Email**
Pflichtenheft	Matthias Jaenicke	matthias.jaenicke@student.kit.edu
Entwurf	Matthias Plappert Julien Duman	undkc@student.kit.edu uncyc@student.kit.edu
Implementierung	Christian Dreher	uaeef@student.kit.edu
Qualitätssicherung	Wasilij Beskorovajnov	uajkm@student.kit.edu
Präsentation	Aydin Tekin	aydin.tekin@student.kit.edu

Inhaltsverzeichnis

1 Zielbestimmung

Im Rahmen der PSE-Veranstaltung soll für das Kryptologikum des Instituts für Kryptographie und Sicherheit die Software Cryptographics zur Demonstration kryptographischer Verfahren erstellt werden.

Das Programm soll im Laufe der Ausstellung Kryptologikum das Interesse der Besucher wecken, sich mit Verschlüsselung zu befassen und schließlich auch ausgewählte Inhalte der Kryptographie näher bringen.

Wichtig ist, dass die Visualisierungen ansprechend und verständlich gestaltet sind. Die Nutzung von Cryptographics soll Spaß machen.

Jedes Verfahren wird in drei Schritten vorgestellt. Diese sind eine automatisch ablaufende Demonstration, ein interaktiver Versuch und ein Wiki-artiger Artikel für weitere Informationen. In der Demonstration wird dem Nutzer der Ablauf der Verschlüsselung möglichst anschaulich demonstriert. Darauf folgend kann im Eigenversuch auf der gleichen Oberfläche die Verschlüsselung selbst ausprobiert werden. Dem Nutzer wird hierbei eine größtmögliche Freiheit zur Wahl sämtlicher Komponenten (wie Klartext, Primzahl, etc.) gelassen. Unter "Mehr Wissen" sollen sich interessierte Benutzer schließlich tiefgehender über das Verfahren informieren können, über seine Entstehung, Einsatz und Nachfolger. Auf geeignete weiterführende Literatur kann mit QR-Codes verwiesen werden.

1.1 Musskriterien

- Visualisierung Caesar-Chiffre
- Visualisierung Vigenère-Chiffre
- Visualisierung Diffie-Hellmann
- Zugriff auf Krypto-Verfahren über Zeitleiste
- Benutzerführung über Phasen
- Krypto-Algorithmen lassen sich schrittweise ausführen, um den Prozess langsam zu verdeutlichen
- Angeleiter Selbstversuch bei komplexen Verfahren
- Praxisbezug auf moderne Kryptoverfahren
- Time-Out bei fehlender Nutzerinteraktion, Rückkehr zum Startbildschirm
- Mögliche Rückkehr zur Startoberfläche per Button zu jedem Ausführungszeitpunkt
- Interaktion des Benutzers über den Touchscreen

- Intuitive Benutzerführung
- Einfache, übersichtliche UI, die sich gut auf Touchscreen bedienen lässt
- Einheitliche optische Darstellung der verschiedenen Verfahren
- Robuste Programmierung
- Schnelle Reaktionszeiten
- Robustheit gegen außergewöhnliche Interaktion
- Beenden des Programms / Wechsel zum Desktop nur per angeschlossener Tastatur, nicht über Touch-Eingabe
- Vermeidung undefinierter Zustände des Programms
- Vermeidung von Sackgassen im Programm, die einen Neustart erfordern
- Lauffähigkeit des Programms in jedem vollständigen Java RTE

1.2 Wunschkriterien

- Visualisierung Public Key – Infrastruktur
- Visualisierung Shamir Secret Sharing
- Visualisierung Passwortsicherheit
- Visualisierung One-Time-Pad
- Visualisierung Blockchiffre (DES, AES)
- Visualisierung Hashfunktionen
- Visualisierung RSA
- Modulares Austauschen / Hinzufügen von weiteren kryptografischen Verfahren
- Animation auf Startbildschirm als Blickfang
- Namen / Icons für Verfahren in der Zeitleiste für bessere Übersichtlichkeit
- Literaturhinweise (auch weiterführende Literatur) mittels QR-Codes
- Optisch ansprechende Visualisierungen durch Nutzung von Grafiken und Animationen
- Verbesserte Verständlichkeit der Visualisierungen durch Verwendung von z.B. Analogien
- Visualisierung von Angriffen auf Verfahren
- Erfassung von grundlegenden Nutzungsstatistiken zur potentiellen externen Analyse des Nutzerverhaltens im Hinblick auf eventuelle zukünftige Optimierung des Programms

1.3 Abgrenzungskriterien

- Implementierung sämtlicher kryptologischer Verfahren nur zu Vorführungszwecken; somit keine festgelegt sichere Implementierung
- Keine optimierte und 100% standardkonforme Implementierung von Krypto-Algorithmen, sondern Fokus auf schrittweiser Ausführbarkeit
- Keine formale Korrektheit bei Erklärungen sondern Ansatz ohne nötige Vorkenntnisse um breiter Masse zugänglich zu sein
- Nicht zur tatsächlichen Verschlüsselung von Texten gedacht

2 Produkteinsatz

2.1 Anwendungsbereiche

Cryptographics soll in erster Linie als Ausstellungsstück für das Kryptologikum des Instituts für Kryptographie und Sicherheit (IKS) am Karlsruher Institut für Technologie (KIT) dienen.

Besuchern der Ausstellung soll das Funktionsprinzip und die Verwendung historischer, sowie aktueller, kryptographischer Verfahren nähergebracht werden. Diese sollen anhand von vereinfachten und beispielhaften Szenarien aus dem Alltag vermittelt werden, mit dem Ziel, ein größeres Interesse an der Materie zu wecken.

Cryptographics soll primär auf dem eeePC (siehe Produktumgebung) im Kryptologikum eingesetzt werden.

2.2 Zielgruppen

Das Programm richtet sich insbesondere an Kinder, Jugendliche und Erwachsene mit grundlegenden Kenntnissen der Mathematik. Die Zielgruppe ist demnach ein anonymes Publikum, das sich potenziell aus fachfremden Personen zusammensetzt. Deshalb dürfen für die Verwendung von Cryptographics keine Vorkenntnisse in der Kryptographie vorausgesetzt werden.

2.3 Betriebsbedingungen

Cryptographics soll im Dauerbetrieb problemlos laufen. Dementsprechend dürfen keine Ausnahmen auftreten, die den Betrieb des Programms behindern, oder gar einen Absturz auslösen, der den Benutzer zur Betriebssystemebene führt. Insbesondere muss darauf geachtet werden, dass schnelle, unkontrollierte Benutzereingaben zu keinem undefinierten Zustand des Programms führen. Auch ein reguläres Beenden von Cryptographics soll durch den Benutzer nicht möglich sein, und nur durch eine angeschlossene Hardwaretastatur erreicht werden können.

Der Betrieb soll für die Hardware des eeePC (siehe Produktumgebung) optimiert sein, der während der Ausstellung über keine Internetverbindung verfügt. Inhalte zu weiterführenden Informationen müssen also lokal in Form von leicht aktualisierbaren und austauschbaren Textdateien vorliegen.

3 Produktumgebung

Cryptographics wird nicht ausschließlich, aber primär für folgende Spezifikationen entwickelt.

3.1 Software

- Windows XP Home edition
- Java Version 7 Update 45

3.2 Hardware

- ASUS Touchscreen EeeTop Tablet
- 1.6 GHz Intel Atom CPU
- 1GB RAM
- 160GB HDD
- 15.6 Zoll LCD
- Integrierter Grafikchip
- Resistiver Touchscreen

4 Produktfunktionen

4.1 Funktionale Anforderungen

4.1.1 Grundfunktionen

/FA100/ Darstellung eines Startbildschirm mit Zeitleiste

/FA101/ Darstellung aller Visualisierungen auf einer Zeitleiste

/FA102/ Interaktion mit der Zeitleiste, um eine einzelne Visualisierung auswählen zu können

/FA103/ Darstellung einer Übersichtsseite zur gewählten Visualisierung

/FA104/ Start der gewählten Visualisierung

/FA105/ Abbruch einer gewählten Visualisierung, um zum Startbildschirm zurückzukehren

/FA106/ Soforthilfe zu einer gewählten Visualisierung

/FA107/ Darstellung weiterführender Information nach erfolgreichem Beenden einer Visualisierung

/FA100/ Prozess: Visualiserung des Verfahrens Cäsar.

Ziel: Benutzer versteht das Verfahren und kann selbstständig beliebigen Klartext verschlüsseln und beliebigen Chiffrat, der mit Cäsar erzeugt wurde, entschlüsseln.

Aktuere: Benutzer.

Beschreibung:

Phase1: Demonstration.

1. Auf dem Bildschirm erscheinen Animationen, die das Verfahren erklären.
2. Der Benutzer sieht die Animation schrittweise und kann sie jederzeit anhalten.
3. Der Benutzer kann in der Animation einen Schritt vorwärts gehen.
4. Der Benutzer kann in der Animation einen Schritt rückwärts gehen.
5. Am Ende der Demonstration wechselt die Visualisierung des Verfahrens in die zweite Phase, dem Selbstversuch.

Phase2: Selbstversuch.

Schritt 1:

1. Der Benutzer wird mit einem kleinen Eingabefenster aufgefordert eine kurze Eingabe, die aus einer kleinen Zeichenfolge besteht, zu tätigen.
2. Benutzer hat die Möglichkeit sich die Eingabe durch einen Button generieren zu lassen.
3. Eingabe ist nun in der Mitte des Bildschirms zu sehen und unterhalb ist das Alphabet abgebildet, wobei jeder Buchstabe nummeriert ist.
4. Nun leuchtet jeder Buchstabe der Eingabe nacheinander auf.
5. Der Benuter wählt das verschlüsselte Gegenstück zum leuchtenden Zeichen in der Eingabe aus dem Alphabet aus.
6. Bei jeder Auswahl des Buchstaben aus dem Alphabet taucht dieser unter der Eingabe auf.

Benutzer wiederholt alles solange, bis alle Buchstaben der Eingabe abgearbeitet sind und das Chiffrat vollständig unter der Eingabe dargestellt ist.

7. Nun entschlüsselt das Programm das Chiffrat selbst.
8. Programm zeigt die Ausgabe unterhalb des Chiffrats an.

Schritt 2:

1. Die alte Bildschirmanzeige verschwindet.
2. Für den Benuter wird vom Programm ein Chiffrat generiert.

   Auf diesem Chiffrat wird dann analog gearbeitet, wie auf der Eingabe aus Schritt 1. Die Funktionalen Anforderungen sind identisch. Die Unterschiede liegen darin, dass der Benutzer nun entschlüsselt.
3. Nach erfolgreichem Entschlüsseln wechselt die Visualisierung des Verfahrens in die Phase 3, Wiki.

Phase 3: Wiki.

1. Sammlung an Artikeln über das Verfahren Cäsar.
2. Nach dem Durchlesen ist die Visualisierung des Verfahrens Cäsar abgeschlossen. Der Benuter wird zum nächsten Verfahren weitergeleitet.

/FA100/ Prozess: Visualiserung des Verfahrens Vigenere.

Ziel: Benutzer versteht das Verfahren und kann selbstständig beliebigen Klartext verschlüsseln und beliebigen Chiffrat, der mit Vigenere erzeugt wurde, entschlüsseln.

Akteure: Benutzer.

Beschreibung:

Phase1: Demonstration.

Identisch mit funktionalen Anforderungen in Caesar Phase 1, Demonstration.

Phase2: Selbstversuch.

Schritt 1:

1. Auf dem Bildschirm erscheinen Animationen, die das Verfahren erklären.
2. Der Benutzer kann die Animationen schrittweise vollziehen.
3. Der Benutzer kann in den Animationen auch schrittweise rückwärts gehen.
4. Am Ende der Animation wechselt die Visualisierung des Verfahrens in die zweite Phase, dem Selbstversuch.

Schritt 2:

1. Die alte Bildschirmanzeige verschwindet.
2. Für den Benuter wird vom Programm ein Chiffrat mit einem Codewort generiert.

   Auf diesem wird dann analog gearbeitet, wie auf der Eingabe aus Schritt 1. Die Funktionalen Anforderungen sind identisch. Die Unterschiede liegen darin, dass der Benutzer nun entschlüsselt. Der Klartext in Schritt 1 ist nun das Chiffrat und der Schlüssel hat dieselbe Rolle. Das Chiffrat wird auf dieselbe Weise entschlüsselt, wie der Klartext verschlüsselt.
3. Programm prüft die entschlüsselte Zeichenfolge auf Richtigkeit.
4. Nach erfolgreichem Entschlüsseln wechselt die Visualisierung des Verfahrens in die Phase 3, Wiki.

Phase 3: Wiki.

Identisch mit den funktionalen Anforderungen aus Cäsar, Phase 3: Wiki.

/FA100/ Prozess: Diffie-Hellman Schlüsselaustausch Demonstration

Ziel: Vermittlung des Diffie-Hellman Schlüsselaustauschs mit Farben-Analogie

Vorbedingung: keine

Nachbedingung (Erfolg): Alice und Bob haben sich auf eine gemeinsame, geheime Farbe geeinigt die Eve nicht kennt

Nachbedingung (Fehlschlag): Gibt es nicht, da nur eine Demonstration ausgeführt wird

Auslösendes Ereignis: Ausgewählt in der Zeitleiste

Beschreibung:

1. Erkläre das Ziel sich auf ein gemeinsames Geheimnis zu einigen, durch Nutzung eines unsicheren Übertragungskanals bei dem Eve lauschen kann, ohne dass Eve das Geheimnis erfährt
2. Demonstriere das Prinzip der Einwegfunktion, anhand Farben die man mischen kann. Farben zu mischen ist einfach, zur einer gemischten Farbe herauszufinden welche Farben verwendet wurden hingegen ist schwer
3. Alice und Bob einigen sich auf eine gemeinsame, nicht geheime Farbe A
4. Alice wählt eine geheime Farbe X und mischt sie mit A zur Farbe AX
5. Bob wählt eine geheime Farbe Y und mischt sie mit A zur Farbe AY
6. Alice schickt AX zu Bob u. Bob schickt AY zu Alice

7. Alice mischt X mit AY zu AYX
8. Bob mischt Y mit AX zu AXY
9. Wenn Eve weder X noch Y erfahren hat, kennt Eve nicht AXY

/FA100/ Prozess: Diffie-Hellman Schlüsselaustausch Selbstversuch

Ziel: Der Nutzer hat einen erfolgreichen Diffie-Hellman Schlüsselaustausch (mit Farben-Analogie) mit Bob durchgeführt

Vorbedingung: keine

Nachbedingung (Erfolg): Alice und Bob haben sich auf eine gemeinsame, geheime Farbe geeinigt, die Eve nicht kennt

Nachbedingung (Fehlschlag): Alice und Bob haben eine gemeinsame, nicht geheime Farbe oder sich auf gar keine Farbe geeinigt

Auslösendes Ereignis: Ende der Demonstration des Diffie-Hellman Schlüsselaustauschs mit Farben, oder die Demonstration wurde Übersprungen

Beschreibung:

1. Der Nutzer übernimmt die Rolle von Alice
2. Benenne die Aufgabe, dass der Nutzer (Alice) sich auf ein gemeinsames Geheimnis mit Bob einigt, wobei ein öffentlicher Kanal genutzt wird auf dem Eve lauscht, ohne dass Eve das Geheimnis erfährt.
3. Der Nutzer (Alice) wird aufgefordert eine Farbe A zu wählen, wurde diese gewählt, wird der Nutzer aufgefordert die nicht geheime Farbe an Bob zu schicken.
4. Der Nutzer soll nun eine geheime Farbe X != A wählen, diese soll er an Bob schicken
5. Bob wählt eine geheime Farbe Y und mischt sie mit A zur Farbe AY
6. Der Nutzer kann nun eine Farbe zu Bob schicken, wenn er die private Farbe X schickt hat er verloren, wenn er A schickt hat er ebenso verloren, wenn er AX schickt geht es weiter
7. Der Nutzer wird aufgefordert nun die richtigen Farben zu mischen, wenn er X mit AY mischt geht es weiter, wenn nicht hat er verloren.
8. Bob mischt Y mit AX zu AXY
9. Wenn Eve weder X noch Y erfahren hat, kennt Eve nicht AXY, der Nutzer kann beglückwünscht werden

4.1.2 Erweiterte Funktionen

/FA500/ Public Key Infrastruktur (PKI)

Prozess: Erläuterung der Vorgänge bei der Verteilung von öffentlichen Schlüsseln.

Ziel: Dem Benutzer ist klar ersichtlich, wie die Verteilung vonstatten geht und welche Instanzen in diesem Rahmen interagieren.

Akteure: Benutzer.

Beschreibung:

Phase 1:Demonstration.

1-4 Die ersten 4 funktionalen Anforderungen sind identisch mit den Anforderungen aus Cäsar und Vigenère.

5 An die Demo anschließend springt das Programm in die Phase 3, Wiki. Phase 2 wird übersprungen, da sich die Überlegungen zu einem Selbstversuch in PKI keine vernünftigen Ergebnisse liefern. Ein Selbstversuch kann aber jederzeit eingebaut werden

Phase 3: Wiki.

1-2 Identisch mit den funktionalen Anforderung aus Cäsar und Vigenère

.

/FA600/ Hashing

Prozess: Hashing Demonstration

Ziel: Dem Benutzer ist die Funktionsweise des Hashings bewusst und erhält einen Einblick in den praktischen Nutzen.

Akteure: Benutzer.

Beschreibung:

Phase 1: Einleitung.

1. Erkläre was Hashes sind.
2. Zeige deren praktischen Nutzen anhand von Beispielen in der Informationstechnologie.

Phase 2: Animation.

1. Erkläre das Ziel "Fingerabdrücke"von Daten zu machen.
2. Demonstriere eine einfache Merkle-Damgard-Konstruktion.

3. Nehme einen festgelegten String und führe diese durch die Konstruktion.
4. Jeder Schritt wird einzeln visualisiert dargestellt, so entsteht Schritt für Schritt der "Fingerabdruck".
5. Gebe den Fingerabdruck dieses Strings an. Speichere das Ergebnis in der Maske, damit dieser später eingesehen werden kann.
6. Nun ersetze einen Buchstaben des vorherigen Strings und führe diese erneut durch die Konstruktion.
7. Nach dem Erhalten des zweiten Hashes, zeige Nutzer dass die Hashes sich gravierend unterscheiden, also dass Hashes “echte Fingerabdrücke” sind.

Phase 3: Weiterführende Informationen.

1. Der Benutzer erhält Links zu weiterführenden Quellen.

/FA700/ Blockchiffren

Prozess: Blockchiffren Demonstration

Ziel: Dem Benutzer ist die Funktionsweise von Blockchiffren bewusst und erhält einen Einblick in die verschiedenen Modí.

Akteure: Benutzer.

Beschreibung:

Phase 1: Einleitung.

1. Erkläre was Blockchiffren sind.
2. Zeige deren praktischen Nutzen anhand von Beispielen in der Informationstechnologie.

Phase 2: Animation ECB.

1. Erkläre den ECB Modus.
2. Zeige den Aufbau vom ECB Modus.
3. Nehme einen festgelegten String und führe diese durch die Konstruktion.
4. Jeder Schritt wird einzeln visualisiert dargestellt, so entsteht Schritt für Schritt das Chiffrat.
5. Gebe das Chiffrat dieses Strings an. Speichere das Ergebnis in der Maske, damit dieser später eingesehen werden kann.
6. Nun ersetze einen Buchstaben des vorherigen Strings und führe diese erneut durch die Konstruktion.

7. Nach dem Erhalten des zweiten Chiffrates, zeige Nutzer dass der ECB Modus homomorph ist.
8. Zeige Nutzer das Problem das dadurch entsteht anhand einem im ECB Modus chiffrierten Bildes.
9. Dechiffriere das erste Chiffrat.

Phase 3: Animation CBC.

1. Erkläre den CBC Modus.
2. Zeige den Aufbau vom CBC Modus.
3. Nehme einen festgelegten String und führe diese durch die Konstruktion.
4. Jeder Schritt wird einzeln visualisiert dargestellt, so entsteht Schritt für Schritt das Chiffrat.
5. Gebe das Chiffrat dieses Strings an. Speichere das Ergebnis in der Maske, damit dieser später eingesehen werden kann.
6. Nun ersetze einen Buchstaben des vorherigen Strings und führe diese erneut durch die Konstruktion.
7. Nach dem Erhalten des zweiten Chiffrates, zeige Nutzer dass die Homomorphie nicht mehr gegeben ist.

Phase 4: Weiterführende Informationen.

1. Der Benutzer erhält Links zu weiterführenden Quellen.

/FA500/ Prozess: Visualisierung des Verfahrens One Time Pad.

Ziel: Benutzer versteht das Verfahren und kann selbstständig beliebigen Klartext verschlüsseln und beliebigen Chiffrat, der mit dem One Time Pad erzeugt wurde, entschlüsseln.

Akteure: Benutzer.

Beschreibung:

Phase 1: Demonstration.

1-5 Die funktionalen Anforderungen and diese Phase sind identisch mit den funktionalen Anforderung in Cäsar, Vigenere und PKI.

Phase2: Selbstversuch.

Schritt 1:

1-6 Die Funktionalen Anforderungen und der Aufbau des ersten Schrittes des Selbstversuches ist analog zu Vigenere, Phase 2: Selbstversuch, Schritt 1. Die Unterschiede sind logischer Natur, wie die Länge des Schlüssels zum Beispiel. Der Benutzer verschlüsselt also mit einem Vigenere, wobei die Schlüssellänge gleich der Länge des Klartextes ist.

Schritt 2:

1-6 Die Funktionalen Anforderungen und der Aufbau des zweiten Schrittes des Selbstversuches ist analog zu Vigenere, Phase 2: Selbstversuch, Schritt 2. Die Unterschiede sind ebenfalls logischer und nicht funktionaler Natur.

Phase 3: Wiki.

1-2 Identisch mit den funktionalen Anforderung aus Cäsar und Vigenere

- Funktionen: Willkommensbildschirm, Algo/Visualisierung wählen, Algo bearbeiten, Soforthilfe passend zum Algo falls man nicht weiter weiß, Algo beenden, ggf. Kontrollfragen oder selbstständiges Anwenden des Gelernten am Ende (z.B. ein Wort mit Caesar verschlüsseln)

RSA und Public-Key Verschlüsselung anhand von Pad-Locks als öffentliche Schlüssel und den Schlüssel dazu als privaten, vielleicht mit Komplementärfarben visualisieren ähnlich wie Diffie-Hellman

/FA100/ Prozess: Shamir Secret Sharing Demonstration

Ziel: Führe Shamir Secret Sharing aus, wobei die Koeffizienten des Polynoms aus den rationalen Zahlen kommen. (Mit rationalen Zahlen ist SSS nicht sicher, allerdings schön um das Prinzip zu vermitteln)

Vorbedingung: keine

Nachbedingung (Erfolg): Shamir Secret Sharing wurde ausgeführt, der Graph des Polynoms wird auf dem Bildschirm gezeichnet, und die Y-Achse sowie der Punkt (0, D) hervorgehoben

Nachbedingung (Fehlschlag): Sollte nicht auftreten, da dies nur eine Demonstration ist, kein Selbstversuch

Auslösendes Ereignis: Ausgewählt in der Zeitleiste

Anmerkung: Es werden nur Punkte (x, y) gewählt, in denen $x > 0$. Außer für das Geheimnis (0, D)

Beschreibung:

1. Das Programm erklärt das Ziel, ein Geheimnis auf N=4 Personen aufzuteilen, wobei K=2 Personen mindestens notwendig sein sollen um das Geheimnis zu rekonstruieren
2. Erkläre dass das Geheimnis der Wert D im Punkt (0, D) ist
3. Das Programm wählt ein zufälliges lineares Polynom und zeichnet dieses, hebt dabei hervor dass die Wahl zufällig sein muss

4. Der geheime Punkt (0, D) wird hervorgehoben und als geheim gekennzeichnet

5. 4 unterschiedliche Punkte werden ausgewählt und markiert

6. Die 4 Punkte werden an 4 Personen verteilt, welche als Figuren dargestellt werden

7. Demonstriere wie man nicht erfährt welche Gerade die richtige ist, wenn man nur einen Punkt kennt, indem ein Punkt gewählt wird und man unterschiedliche mögliche Geraden nacheinander zeichnet. Somit erfährt man auch nicht das Geheimnis D

8. Zeige nun wie 2 Personen durch ihre 2 Punkte das Geheimnis rekonstruieren, indem sie die Gerade wiederherstellen und D ablesen

9. Erkläre das man dies auf weitere Polynome verallgemeinern kann

10. Erkläre das der Grad des Polynoms abhängig von der Zahl K ist, welche beschreibt wie viele Personen nötig sind um das Geheimnis zu rekonstruieren

11. Demonstriere das ganze nochmal mit N=4 und K=3 und somit einer quadratischen Funktion, zeige aber diesmal das 2 Punkte nicht ausreichen um D zu rekonstruieren

/FA100/ Prozess: Shamir Secret Sharing Selbstversuch

Ziel: Der Nutzer erstellt ein Geheimnis D mit Hilfe von Shamir Secret Sharing. Dieses Rekonstruiert er, indem er das Polynom mit K Punkten interpoliert und das Geheimnis abliest

Vorbedingung: Der Nutzer hat sich die zugehörige Demonstration angesehen

Nachbedingung (Erfolg): Shamir Secret Sharing wurde ausgeführt, der Graph des Polynoms wird auf dem Bildschirm gezeichnet, und die Y-Achse sowie der Punkt (0, D) hervorgehoben

Nachbedingung (Fehlschlag): -

Auslösendes Ereignis: Ende der SSS Demonstration, oder Demo übersprungen

Beschreibung:

1. Der Nutzer wählt die Zahl N, für die 2 <= N <= 6 gilt, welche angibt wie viele Personen sich das Geheimnis D teilen

2. Der Nutzer wählt die Zahl K, für die 1 <= K <= N gilt, welche angibt wie viele Personen nötig sind, um das Geheimnis D zu rekonstruieren

3. Der Nutzer wählt das Geheimnis D
4. Nun wird ein zufälliges Polynom vom Grad K-1 konstruiert, welches den Punk (0, D) enthält
5. Der Nutzer darf nun verschiedene Punkte auswählen, welche auf die Personen verteilt werden
6. Nun kann er versuchen mit einer unterschiedlichen Anzahl an Mitwissern versuchen (0, D) herauszufinden, er selektiert die Mitwisser und wenn er meint er hat genug kann er auf einen Button drücken, welches versucht (0, D) zu rekonstruieren
7. Wenn die Anzahl kleiner ist als K, zeige mögliche Polynome als Graphen an, und demonstriere somit das man (0, D) nicht rekonstruieren kann, gehe zum vorherigen Schritt
8. Ist die Anzahl der ausgesuchten Mitwisser größer gleich K, so interpoliere das gesuchte Polynom und geben (0, D) an
9. Gratulatiere dem Nutzer und zeige Button für Wiki-Artikel oder QR-Codes an

4.2 Nichtfunktionale Anforderungen

/NA100/ Schnelle Reaktionszeit

/NA200/ Große Robustheit

/NA300/ Intuitive Benutzerführung

- Die Anwendung muss möglichst performant und verzögerungsfrei auf Benutzereingaben reagieren (soweit das die HW zulässt...)

- Die Anwendung muss leicht zu bedienen sein

5 Produktdaten

/PD10/ Nutzerverhalten: Für einen Logeintrag zur externen Analyse von anonymem Nutzerverhalten sind folgende Daten lokal zu speichern:

- Zeitstempel
- zugehörige Visualisierung
- Ereignis-Code
- Frei wählbare, textuelle Beschreibung des Ereignisses

6 Benutzerschnittstelle

6.1 Ablauf einer Visualisierung

Der Willkommensbildschirm besteht aus einem einleitenden Text, der das Ausstellungsstück kurz dem Publikum vorstellt, und einer Zeitleiste (Fig. 1). Auf der Zeitleiste werden mithilfe einer geeigneten Skala Jahreszahlen dargestellt. Die kryptografischen Visualisierungen werden auf dieser Zeitleiste mithilfe von "Meilensteinen"dargestellt. Weiterhin werden die Markierungen in grün, gelb oder rot eingefärbt, um den Schwierigkeitsgrad auf einen Blick erkennbar zu machen.

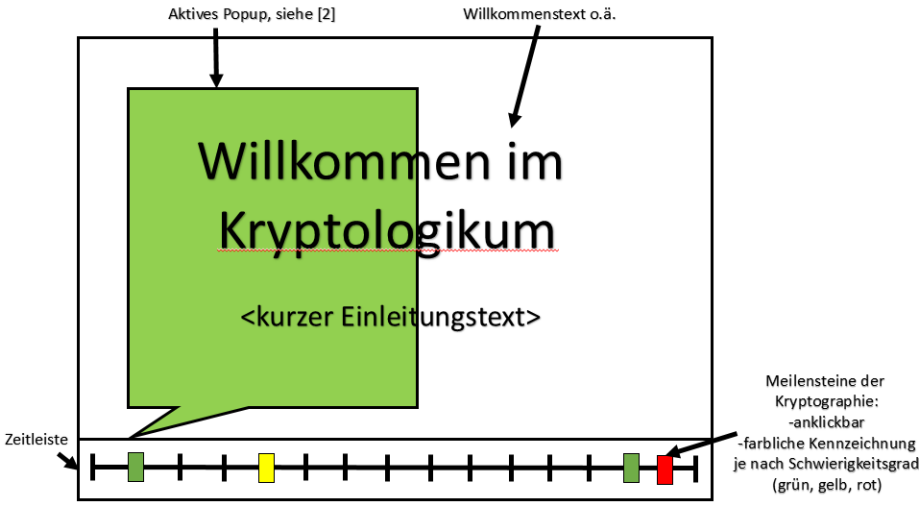

Abbildung 1: Willkommensbildschirm mit Zeitleiste.

Sobald ein Meilenstein ausgewählt wurde, erscheint ein Popover (Fig. 2), dass den Namen, die Schwierigkeit und den Zweck des gewählten kryptografischen Verfahrens noch einmal zusammenfasst. Weiterhin werden ein Screenshot der eigentlichen Visualisierung sowie ein Button, um diese zu starten, dargestellt.

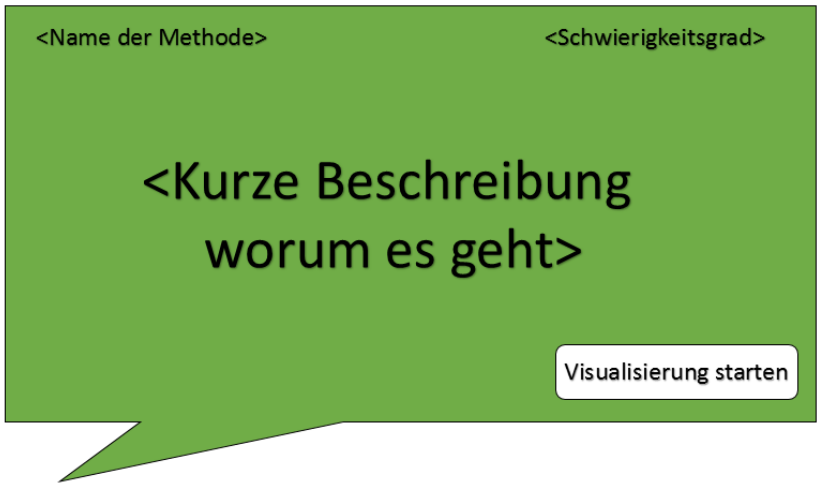

Abbildung 2: Detailansicht des Popovers in Fig. 1.

Nachdem der Benutzer eine Visualisierung startet, wird diese angezeigt (Fig. 3). Hierbei ist hervorzuheben, dass der Benutzer zu jedem Zeitpunkt die Visualisierung abbrechen kann, indem er auf den Button in der oberen linken Ecke klickt. Hilfe erhält er zu jedem Zeitpunkt im rechten oberen Eck.

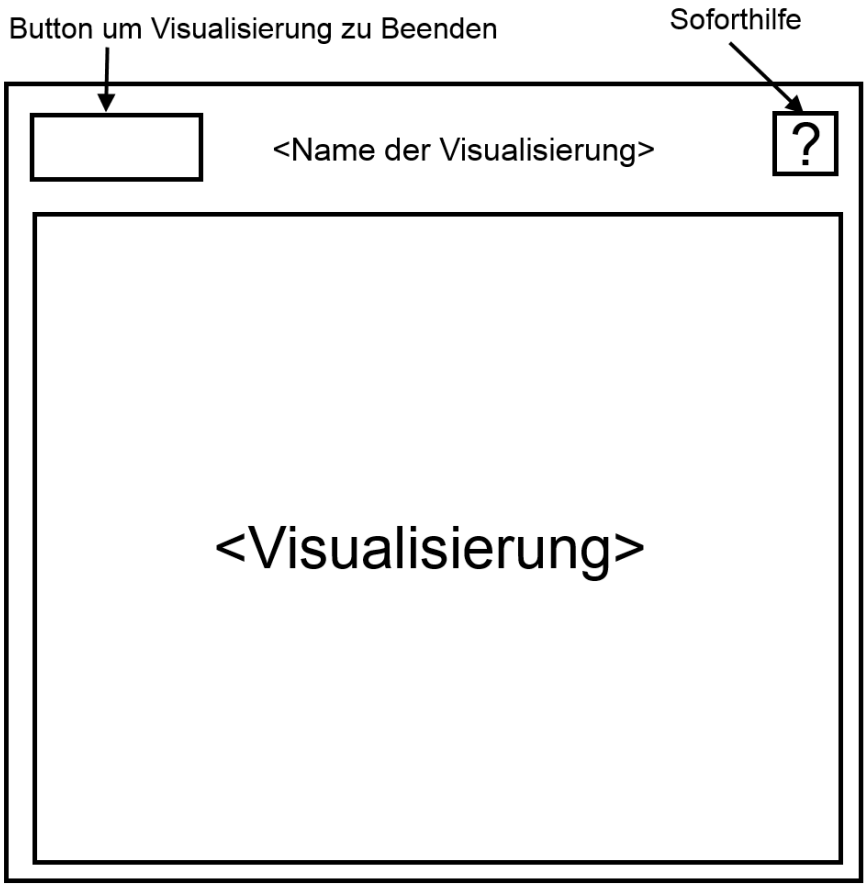

Abbildung 3: Darstellung einer Visualisierung.

Falls der Benutzer die Visualisierung erfolgreich beendet hat, wird (Fig. 4) angezeigt. Zunächst wird dem Benutzer zu seinem Erfolg gratuliert. Weiterhin besteht die prominente Möglichkeit, zum Willkommensbildschirm zurückzukehren. Falls das Interesse des Benutzers geweckt wurde, kann er zu (Fig. 5) wechseln.

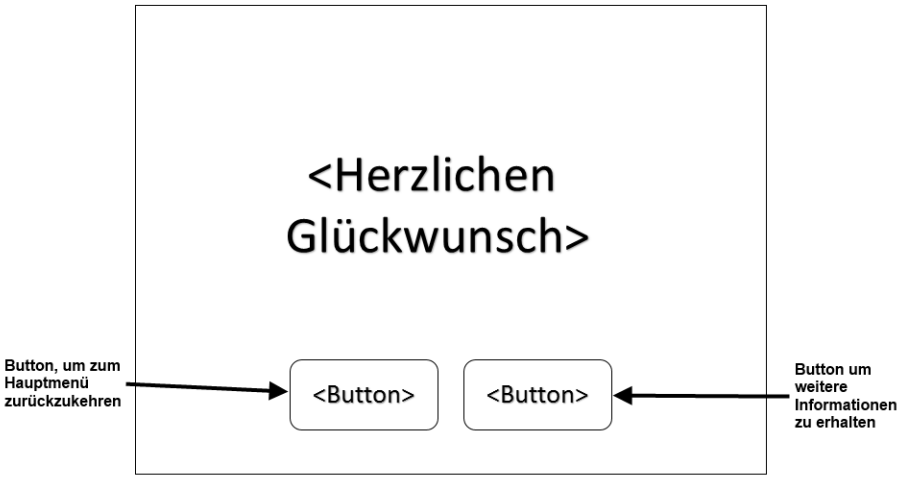

Abbildung 4: Erfolgreiches Beenden einer Visualisierung.

Hier bekommt der Benutzer weiterführendes Material über sein zuvor gewähltes Thema. Außer dem Text mit den Quellen, kann auch ein QR-Code angezeigt werden, welcher vom Nutzer gescannt werden kann. Auch hier bietet sich die Möglichkeit wieder zum Hauptmenü zurückzukehren.

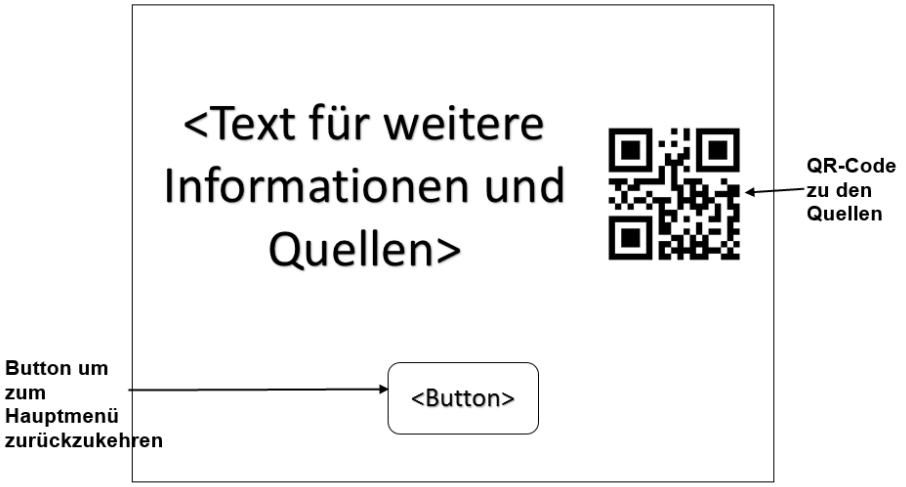

Abbildung 5: Weiterführende Informationen.

7 Systemmodelle

7.1 Systemübersicht

Das System verwendet das bekannte MVC-Entwurfsmuster. Weiterhin ist geplant, das System in zwei Schichten zu unterteilen. Von unten nach oben enthält Schicht 1 wiederverendbare und in sich abgeschlossene Bibliotheken. Zum einen handelt es sich hierbei um Code von Dritten (z.B. um QR-Codes zu generieren). Weiterhin ist eine Sammlung wiederverwendbarer Klassen (nachfolgen *GraphicsLib* genannt) geplant. *GraphicsLib* wird vor allem aus UI-Elemente bestehen, die für die Implementierung aller Visualisierungen wertvoll sind.

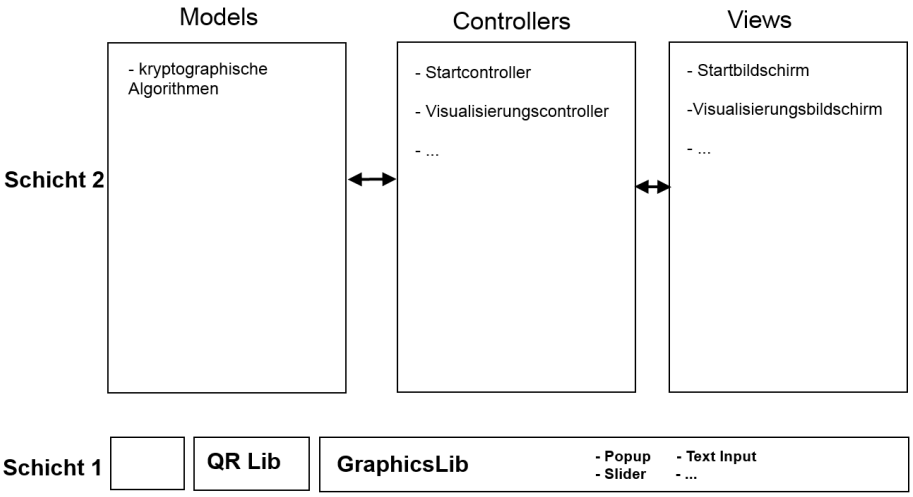

Abbildung 6: Schematische Darstellung des beschriebenen Systems.

8 Beispielszenarien

- Das Beispiel läuft schrittweise ab. Problem wird dargestellt: A möchte B eine Nachricht schicken, die nicht von einer anderen Person als B gelesen werden soll. C versucht die Nachricht zu bekommen. Dabei können A und B nur über ein unsicheres Medium kommunizieren (Analogie z.B. Schlüssel per Post verschicken), das von C abgehört wird.

- A und B generieren ihre Schlüsselpaare. Der genaue Schlüssel ist nicht relevant.
- A fragt B nach seinem public key, den B auch bereitwillig mitteilt. C hört den public key von B.
- A verschlüsselt die Nachricht mit dem public key von B und verschickt die Nachricht an B. C hört die verschlüsselte Nachricht ab.
- B entschlüsselt die Nachricht von A mit seinem private key und kann die Nachricht lesen
- C versucht mit der Nachricht irgendetwas anzufangen und versucht sie mit dem abgehörten public key von B zu entschlüsseln. Das scheitert natürlich.

  Beispiel einer Visualisierung anhand des Caesar-Algos:

- Es wird erklärt, dass vermutlich Caesar dieses Verfahren verwendet hat um geheime Nachrichten zu verschlüsseln
- Erklärung des Prinzips an einem Beispielsatz. Es werden schrittweise alle gleichen Buchstaben markiert und in ihr Äquivalent umgewandelt (z.B. "Hallo, wie geht's?> zuerst alle H zu K, dann alle a zu d, dann alle l zu o). Hier gibt es die Möglichkeit, schrittweise durchzugehen und noch mal zurück zu gehen. Außerdem kann man wenn man das Prinzip verstanden hat ans Ende springen.
- Als nächstes muss die Person ein Wort selbst verschlüsseln mit Vorschrift (z.B. a -> b).
- Am Ende wird mithilfe eines Diagrammes und der Kenntnis, das E der häufigste Buchstabe im Deutschen ist gezeigt, dass man diese Verschlüsselung leicht umgehen kann

Szenario 1: Bob ist interessiert in Kryptographie und besucht zu diesem Anlass das Kryptologikum, um mehr Informationen diesbezüglich zu erhalten. Er nähert sich dem Cryptographics, wo ein Bildschirmschoner verschiedene kryptographische Verfahren Teaser-ähnlich visualisiert. Eine Berührung des Bildschirmes zeigt die Willkommensoberfläche von Cryptographics: Ein Zeitstrahl, auf dem chronologisch angeordnet verschiedene kryptographische Verfahren, ihrem Erfindungsjahr zugeordnet, genannt werden und farblich durch ihre Komplexität von grün (leicht) bis rot (schwer) gekennzeichnet sind. Bob wählt durch eine Berührung die Cäsar-Chiffre aus, sie befindet sich nämlich ganz am Anfang des Zeitstrahls. Daraufhin erscheint eine kurze Einleitung als Popup, im Hintergrund baut sich die Oberfläche auf und der Zeitstrahl gleitet verkleinert als Gesamtübersicht an den unteren Bildschirmrand. Nach dem Durchlesen der Einleitung schließt Bob das Popup, und er sieht eine Demonstration der Cäsar-Chiffre als Schritt-für-Schritt-Beispiel auf einer speziell für das Verfahren ausgelegten Oberfläche. Auf genau dieser Oberfläche kann Bob nach dem Beispiel sein

neu erlerntes Wissen über das Verfahren anwenden, um Texte zu verschlüsseln, und durch einen gegebenen Schlüssel zu entschlüsseln. Nachdem er das Verfahren nun verinnerlicht hat, kann Bob das nächste Verfahren auswählen, entweder durch Drücken der "Weiter"-Taste, oder indem er ein Verfahren direkt am Zeitstrahl auswählt. Da er die weiteren Verfahren auch noch sehen möchte, arbeitet er sich am Zeitstrahl entlang durch das Programm, wobei er dann schließlich auf das Diffie-Hellman-Verfahren trifft, über welches er noch mehr Informationen erhalten möchte. Er berührt einen dafür vorgesehenen Button, welcher ein Popup mit einem QR-Code öffnet, dass letztendlich zu den ausführlichen Quellen weiterleitet. Alternativ ist dort auch eine gekürzte URL zu finden die man schnell abschreiben kann (Bsp.: http://goo.gl/abcd12).

Szenario 2: Die ungeduldige Alice wurde von ihrer Freundin überredet, an der Ausstellung teilzunehmen. Sie sieht das Cryptographics und versucht, es durch wiederholte zufällige Eingaben zum Absturz zu bringen. Das Programm zeigt sich jedoch schnell als zu robust, um einem solchen Angriff zu erliegen. Schließlich wählt sie das komplizierteste kryptographische Verfahren aus, um Besuchern nach ihr den Einstieg zu erschweren. Nachdem sie gegangen ist, muss sie jedoch enttäuscht feststellen, dass der nächste Besucher keine weitere Schwierigkeiten hatte, das Programm mit dem dafür vorgesehenen Button in den Grundzustand zurückzuversetzen.

Szenario 3: Nachdem Bob nun längere Zeit an Cryptographics verbracht hat, blickt er erschreckt auf die Uhr um festzustellen, dass es schon viel zu spät geworden ist. Er bricht auf, und lässt das Programm genau so zurück, wie er es gerade eben noch benutzt hat. Nach seiner letzten Interaktion erscheint nach x Sekunden eine Meldung, die Bob darauf hingewiesen hätte, dass sich das Programm bald zurücksetzt, sollte in den nächsten y Sekunden keine Eingabe mehr erfolgen. Da er jedoch schon an der Bushaltestelle steht, aktiviert Cryptographics nun den Bildschirmschoner, den Bob zuvor auch schon vorgefunden hatte, und setzt im Hintergrund das Programm wieder auf den Ursprungszustand zurück.

Szenario 4: Die Ausstellung ist für diesen Tag nun zu Ende, und die Kryptologikum-Administratorin Claudia schließt gerade noch die Tür hinter Bob ab, um sich jetzt den strombetriebenen Exponaten zu widmen. Als sie am Cryptographics ankommt, versucht sie zunächst, irgendwie zum Betriebssystem zurückzukehren, um den PC sicher herunterfahren zu können. Nach kurzer Zeit stellt sie fest, dass das ohne eine zusätzlich eingesteckte Hardware-Tastatur auf keinen Fall möglich ist. Sie holt also eine USB-Tastatur, steckt sie ein, kehrt über den Windows-Taskmanager zum Betriebssystem zurück, fährt den PC herunter und macht die Lichter aus.

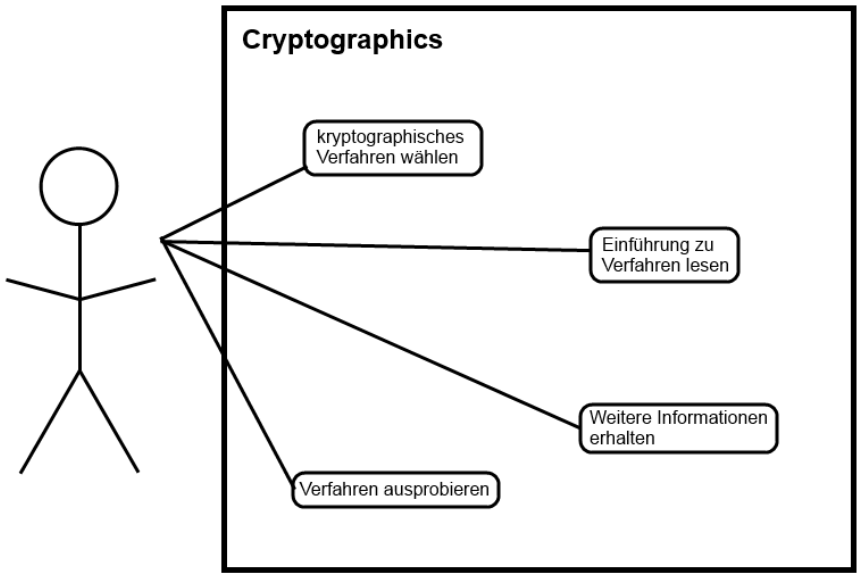

Abbildung 7: Anwendungsfalldiagramm zu beschriebenen Szenarien.

9 Globale Testfälle

- automatisiertes Testen mithilfe von JUnit vor allem der Krypto-Algorithmen - ggf. automatisiertes Testen der UI mithilfe von Szenarien und geeignetem Framework Stresstest mit möglichst zufälligen und willkürlichen Eingaben (wie es Kinder eben tun) Usability Tests mit passenden Testpersonen
Es ist sehr ratsam, dass die globalen Testfälle mit /Txy/ sich auf die Funktionalität /Fxy/ beziehen. So ist eine gewisse Konsistenz, Überdeckungsgrad und Übersichtlichkeit gewährleistet. Folgt demnächst, wenn die Funktionalen Anforderungen fertig sind!

10 Qualitätsbestimmung

11 Anhang

11.1 Zeichnungen, Skizzen, etc.

-Hier können wir ggf. die Zeichnungen und Skizzen anhängen (analog wie bei Handyverträge)...-

11.2 Glossar

Cryptographics

Name des zu entwickelnden Produkts.

IKS

Institut für Kryptographie und Sicherheit.

KIT

Karlsruher Institut für Technologie.

Kryptologikum

Ausstellung des IKS.

PSE

Praxis der Software-Entwicklung.