Mergentheim/Mosbach Security Agency (MSA)

Zu Simulationszwecken wird ein Unternehmensnetzwerk als EventBus dargestellt. An dieses Unternehmensnetzwerk sind fünf Zweigstellen [i] BranchHKG, [ii] BranchCPT, [iii] BranchSFO, [iv] BranchSYD und [v] und BranchWUH angeschlossen.

Zwischen den Zweigstellen werden verschlüsselte Nachrichten ausgetauscht.

Für die Verschlüsselung stehen ein klassisches Verfahren und RSA zur Verfügung.
Die kryptographischen Algorithmen sind in austauschbaren Komponenten gekapselt.
Die Komponenten als jar sind digital signiert.
Nur Komponenten mit valider digitaler Signatur können geladen werden.

Der/die Schlüssel für die Ver-/Entschlüsselung sind in geeigneten Konfigurationsdateien im Format JSON zu speichern.

Bei dem Start der Applikation werden bereits definierte Channel automatisch aufgebaut.

Durch Drücken der **Taste [F3]** wird der **Debug-Modus** aktiviert oder deaktiviert. Bei aktiviertem Debug-Modus wird bei Ausführung der Befehle *encrypt message* und *decrypt message* im Verzeichnis log eine Logdatei [encrypt | decrypt]_[algorithm]_[unix_time_unixseconds].txt angelegt. Diese Logdatei zeichnet die Schritte detailliert und nachvollziehbar auf.

Durch Drücken der **Taste [F8]** wird das **Logfile** mit dem neuesten Zeitstempel in den Ausgabebereich der GUI geladen und angezeigt.

Die Applikation wird über **CQL** (Cryptographic Query Language) gesteuert.

Durch Mausklick auf den **Button execute** in der GUI oder Drücken der **Taste [F5]** wird der Befehl ausgeführt.

- **encrypt message "[message]" using [algorithm] and keyfile [filename]**
  Die zu dem Algorithmus korrespondierende Komponente [algorithm].jar wird dynamisch geladen und die Meldung mit dem key verschlüsselt.
  Die verschlüsselte Meldung wird im Ausgabebereich der GUI angezeigt.
- **decrypt message "[message]" using [algorithm] and keyfile [filename]**
  Die zu dem Algorithmus korrespondierende Komponente [algorithm].jar wird dynamisch geladen und die Meldung mit dem key entschlüsselt.
  Die entschlüsselte Meldung wird im Ausgabebereich der GUI angezeigt.
- **crack encrypted message "[message]" using shift**
  crack encrypted message “[message]” using rsa and keyfile [filename]
  Die zu dem Algorithmus korrespondierende Komponente [shift | rsa]_cracker.jar wird dynamisch geladen und versucht innerhalb von maximal 30 Sekunden die Meldung zu entschlüsseln. Wurde die Meldung innerhalb der Zeitvorgabe entschlüsselt, wird die entschlüsselte Meldung im Ausgabebereich der GUI angezeigt.
  Wurde die Meldung nicht innerhalb der Zeitvorgabe entschlüsselt, erfolgt die Meldung "cracking encrypted message "[message]" failed" im Ausgabebereich der GUI.

- **register participant [name] with type [normal | intruder]**
  Existiert kein Teilnehmer mit diesem Namen wird [i] ein Datensatz in der Tabelle participants und [ii] die Tabelle postbox_[participant_name] angelegt.
  Im Ausgabebereich der GUI wird die Meldung "participant [name] with type [normal | intruder] registered and postbox_[participant_name] created" angezeigt.

  Existiert ein Teilnehmer mit diesem Namen wird die Meldung "participant [name] already exists, using existing postbox_[participant_name]" im Ausgabebereich der GUI angezeigt.

  Für die **Simulation** werden folgende participants angelegt:

branch_hkg	normal
branch_cpt	normal
branch_sfo	normal
branch_syd	normal
branch_wuh	normal
msa	intruder

- **create channel [name] from [participant01] to [participant02]**
  Ein **Channel ist** als **dedizierter EventBus** – basierend auf **Google Guava** – realisiert.

  Für die bidirektionale Kommunikation bzw. das Senden von verschlüsselten Nachrichten zwischen Participant vom Typ normal ist ein Channel notwendig.

  Existiert bereits ein Channel mit dem Namen wird die Fehlermeldung "channel [name] already exists″ im Ausgabebereich der GUI angezeigt. Existiert bereits eine Kommunikationsbeziehung zwischen participant01 und participant02, wird die Fehlermeldung "communication channel between [participant01] and [participant02] already exists" im Ausgabebereich der GUI angezeigt.

  Sind participant01 und participant02 identisch, wird die Fehlermeldung "[participant01] and [participant02] are identical – cannot create channel on itself" im Ausgabebereich der GUI angezeigt.

  Existiert [i] kein Channel mit dem Namen und [ii] keine Kommunikationsbeziehung zwischen den beiden participant, wird ein Datensatz in der Tabelle channel angelegt und die Meldung ″channel [name] from [participant01] to [participant02] successfully created" im Ausgabebereich der GUI angezeigt.

  Nach erfolgreicher Erstellung eines channel erfolgt die Meldung "channel [name] from [participant01] to [participant02] created" im Ausgabebereich der GUI, in der Tabelle channel wird ein Datensatz angelegt.

  Für die **Simulation** werden folgende channel angelegt:

hkg_wuh	branch_hkg	branch_wuh
hkg_cpt	branch_hkg	branch_cpt
cpt_syd	branch_cpt	branch_syd
syd_sfo	branch_syd	branch_sfo

- **show channel**
  Im Ausgabebereich der GUI werden die Channel angezeigt.
  hkg_wuh | branch_hkg and branch_wuh
  hkg_cpt | branch_hkg and branch_cpt
  cpt_syd | branch_cpt and branch_syd
  syd_sfo | branch_syd and branch_sfo

- **drop channel [name]**
  Existiert der channel mit dem Namen, wird dieser Datensatz aus der Tabelle channel gelöscht und im Ausgabebereich der GUI die Meldung "channel [name] deleted" angezeigt. Existiert kein channel mit dem Namen, wird im Ausgabebereich der GUI die Fehlermeldung "unknown channel [name]" ausgegeben.

- **intrude channel [name] by [participant]**
  Existiert der channel mit dem Namen, wird der participant vom Typ intruder für den Channel registriert und erhält alle Nachrichten die über diesen Channel kommuniziert werden.

  Bei Erhalt einer Nachricht wird in der Tabelle postbox des intruder ein neuer Datensatz erstellt, das Attribut message wird auf den Wert unknown gesetzt.

  Der participant lädt dynamisch die zu dem Algorithmus korrespondierende Komponente [algorithm]_cracker.jar und versucht innerhalb von maximal 30 Sekunden die Meldung zu entschlüsseln. Wird innerhalb der Zeitvorgabe die Nachricht erfolgreich entschlüsselt, wird das Attribut message auf den Wert der Meldung im Klartext gesetzt.

  Im Fall einer erfolgreichen Entschlüsselung wird im Ausgabebereich der GUI die Meldung "intruder [name] cracked message from participant [name] | [message]" angezeigt.

  Im Fall einer nicht erfolgreichen Entschlüsselung wird im Ausgabebereich der GUI die Meldung "intruder [name] | crack message from participant [name] failed" angezeigt.

- **send message "[message]" from [participant01] to [participant02] using [algorithm] and keyfile [name]**

  Die Nachricht wird mit dem Algorithmus und keyfile verschlüsselt und von participant01 an participant02 über den Channel kommuniziert.

  Existiert zwischen participant01 und participant02 kein channel, wird die Fehlermeldung "no valid channel from [participant01] to [participant02]" im Ausgabebereich der GUI angezeigt.

  Existiert zwischen participant01 und participant02 ein channel wird die Nachricht mit dem Algorithmus und keyfile verschlüsselt und über den Channel kommuniziert.

  Das Versenden der Nachricht wird in der Tabelle messages protokolliert.

  Der Empfänger participant02 lädt dynamisch die zu dem Algorithmus korrespondierende Komponente [algorithm].jar und entschlüsselt die Nachricht. Nach Entschlüsselung wird in der Tabelle postbox_[participant02_name] ein Datensatz erstellt und Ausgabebereich der GUI die Meldung "[participant02_name] received new message" angezeigt.

algorithms		
id	TINYINT NOT NULL	**PK** Fortlaufende Nummerierung beginnend bei 1
name	VARCHAR(10) NOT NULL	unique

types		
id	TINYINT NOT NULL	**PK** Fortlaufende Nummerierung beginnend bei 1
name	VARCHAR(10) NOT NULL	unique

participants		
id	TINYINT NOT NULL	**PK** Fortlaufende Nummerierung beginnend bei 1
name	VARCHAR(50) NOT NULL	unique
type_id	TINYINT NOT NULL	FK

channel		
name	VARCHAR(25) NOT NULL	**PK**
participant_01	TINYINT NOT NULL	FK ► participants
participant_02	TINYINT NOT NULL	FK ► participants

messages		
id	TINYINT NOT NULL	**PK** Fortlaufende Nummerierung beginnend bei 1
participant_from_id	TINYINT NOT NULL	FK ► participants
participant_to_id	TINYINT NOT NULL	FK ► participants
plain_message	VARCHAR(50) NOT NULL	Meldung im Klartext
algorithm_id	TINYINT NOT NULL	FK ► algorithms
encrypted_message	VARCHAR(50) NOT NULL	Verschlüsselte Meldung
keyfile	VARCHAR(20) NOT NULL	Dateiname keyfile
timestamp	INTEGER	Unix-Zeitstempel in Sekunden

postbox_[participant_name]		
id	TINYINT NOT NULL	**PK** Fortlaufende Nummerierung beginnend bei 1
participant_from_id	TINYINT NOT NULL	FK ► algorithms
message	VARCHAR(50) NOT NULL	Entschlüsselte Meldung
timestamp	INTEGER	Unix-Zeitstempel in Sekunden

Komponente	Schnittstelle
shift.jar	String encrypt(String plainMessage, File keyfile) String decrypt(String encryptedMessage, File keyfile)
shift_cracker.jar	String decrypt(String encryptedMessage)
rsa.jar	String encrypt(String plainMessage, File publicKeyfile) String decrypt(String encryptedMessage, File privateKeyfile)
rsa_cracker.jar	String decrypt(String encryptedMessage, File publicKeyFile)

Anforderung	Bewertung
01 [S1 und S2] Basisarchitektur | 10 Punkte	**09** [S2] send message | 3 Punkte
02 [S1] Taste [F3], Debug-Modus | 2 Punkte	**10** [S1] encrypt message | 3 Punkte
03 [S2] Taste [F8], Logfile | 2 Punkte	**11** [S1] decrypt message | 3 Punkte
04 [S1] register participant | 2 Punkte	**12** [S2] crack message | 6 Punkte
05 [S1] create channel | 2 Punkte	**13** [S1] Komponente shift.jar | 5 Punkte
06 [S2] show channel | 2 Punkte	**14** [S2] Komponente rsa.jar | 5 Punkte
07 [S2] drop channel | 2 Punkte	**15** [S2] Komponente shift_cracker.jar | 10 Punkte
08 [S1] intrude channel | 3 Punkte	**16** [S1] Komponente rsa_cracker.jar | 10 Punkte

Aufgabenstellung | Wichtige Hinweise

- **Bearbeitung** der Komplexaufgabe erfolgt **im Team** durch zwei Studierende.
- **Programmiersprache** | Java 15.0.1 oder höher
- **IntelliJ IDEA** Community 2020.3 oder höher
- Nutzung **GitHub** und Build-Management mit **gradle** 6.7.1 oder höher.
- Datenbank: **HSQLDB** 2.5.1 oder höher.
- **Implementierung** einer technisch einwandfrei lauffähigen Applikation.
  Kommunikation der Nachricht "vaccine for covid is stored in building abc".
- Es sind **keine dedizierten Tests in JUnit** zu erstellen.
- Verwendung geeigneter **englische**r Begriffe für **Namen** und **Bezeichnungen**.
- Erstellung einer **7-Zip-Datei kpl_[matrikelnummern_teammitglieder].7z**
  mit dem vollständigen Projekt und **Upload in Moodle**.
  In der beigefügten **readme.txt** ist die Zuordnung Anforderung/Studierender dokumentiert.
- **Zeitansatz**: 40 Stunden je Studierender S1 und S2
- **Abgabetermin**: Sonntag, 04.04.2021
- **Bewertung**: 40 Punkte je Studierender S1 und S2