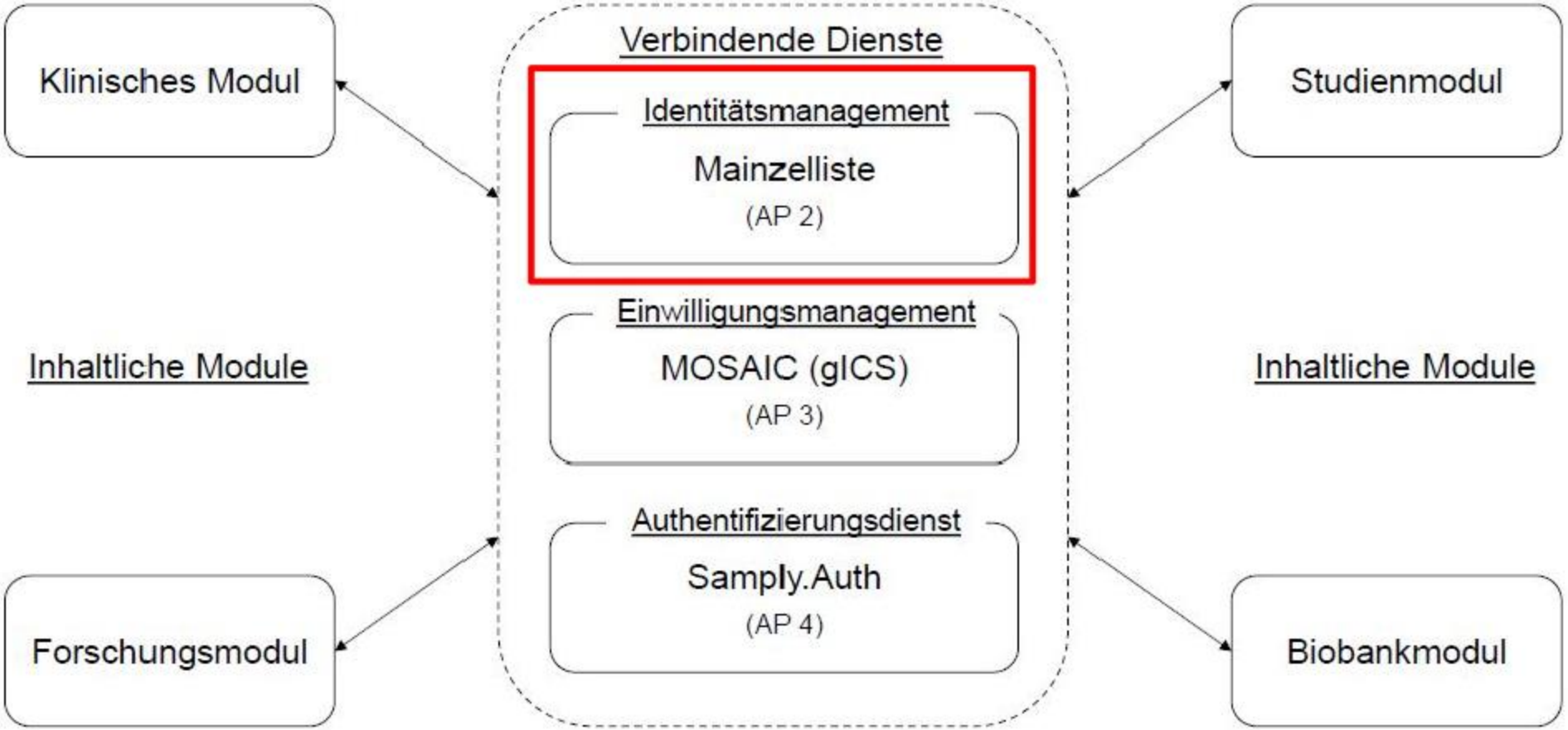
TMF – Technologie- und Methodenplattform
für die vernetzte medizinische Forschung e.V.

Mainzelliste Tutorial

TMF-Tutorial MAGIC | Bonn | 19.03.2019

Florian Stampe
Deutsches Krebsforschungszentrum
Verbundinformationssysteme

Einführung


Klinisches Modul
Verbindende Dienste
Identitätsmanagement
Mainzelliste
(AP 2)
Studienmodul
Einwilligungsmanagement
MOSAIC (gICS)
(AP 3)
Inhaltliche Module
Inhaltliche Module
Authentifizierungsdienst
Samply.Auth
(AP 4)
Forschungsmodul
Biobankmodul

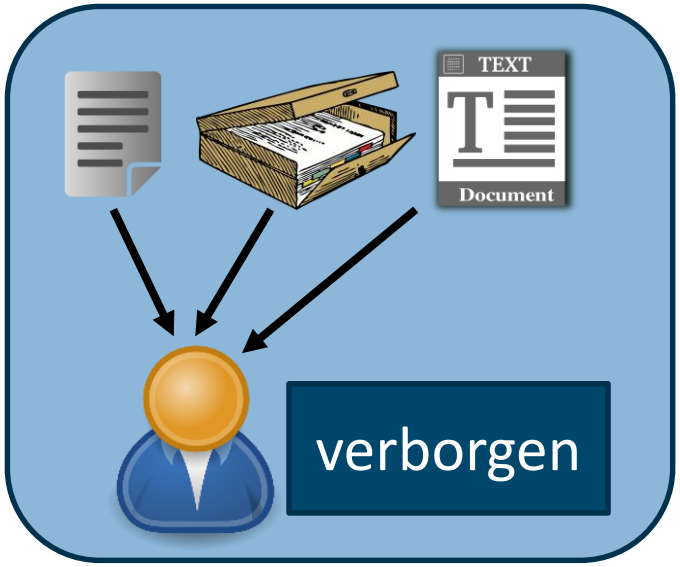
Gliederung

1. Einführung ID-Management – wozu, wie & womit
2. Funktionalitäten und Eigenschaften
3. Schnittstelle & Verwendung
4. Klassische Installation & Docker-Betrieb
5. Konfiguration
6. Funktionalitäten durch Erweiterungen
7. Fallbeispiel
8. Ausblick: Docker Secure Record Linkage Demo

Einführung – Wozu ein ID-Management?

Wozu dient ein ID-Management in Forschungsverbünden?

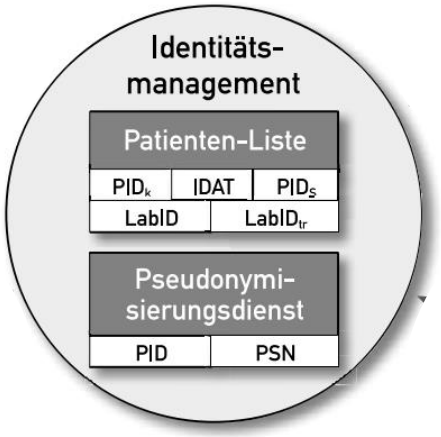
1. Daten, die zur selben Person gehören, zuzuordnen
2. Und dabei die Identität dieser Person vor Unberechtigten zu verbergen
3. Diese Ziele stehen in einem gewissen Spannungsverhältnis. Korrekte Zuordnung setzt eine Erkennbarkeit voraus.



Einführung – Wie funktioniert ein ID-Management?

- Etabliertes Vorgehen: IDAT von dritter Stelle pseudonymisieren zu lassen
- Dabei werden sogenannte Personenidentifikatoren (PID) erzeugt
- Mittels PIDs können Daten eines Patienten über Instititutsgrenzen hinweg verknüpft werden
  - Informationelle Gewaltenteilung
  - Pseudonymisierte Speicherung

Identitätsmanagement

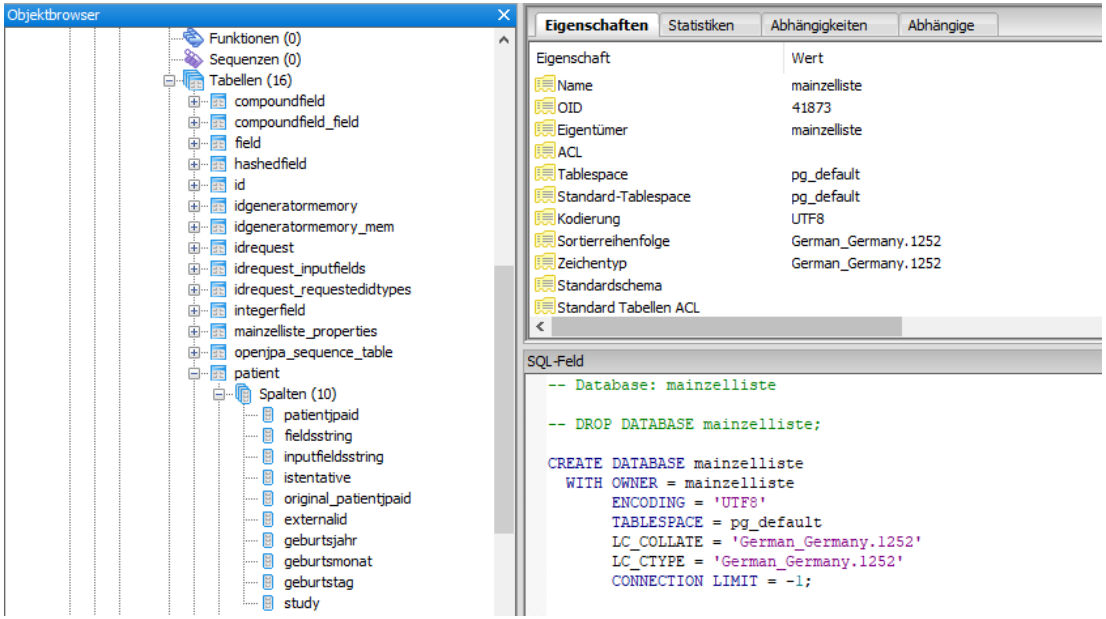


Aus: Leitfaden zum Datenschutz in medizinischen Forschungsprojekten
Generische Lösungen der TMF 2.0

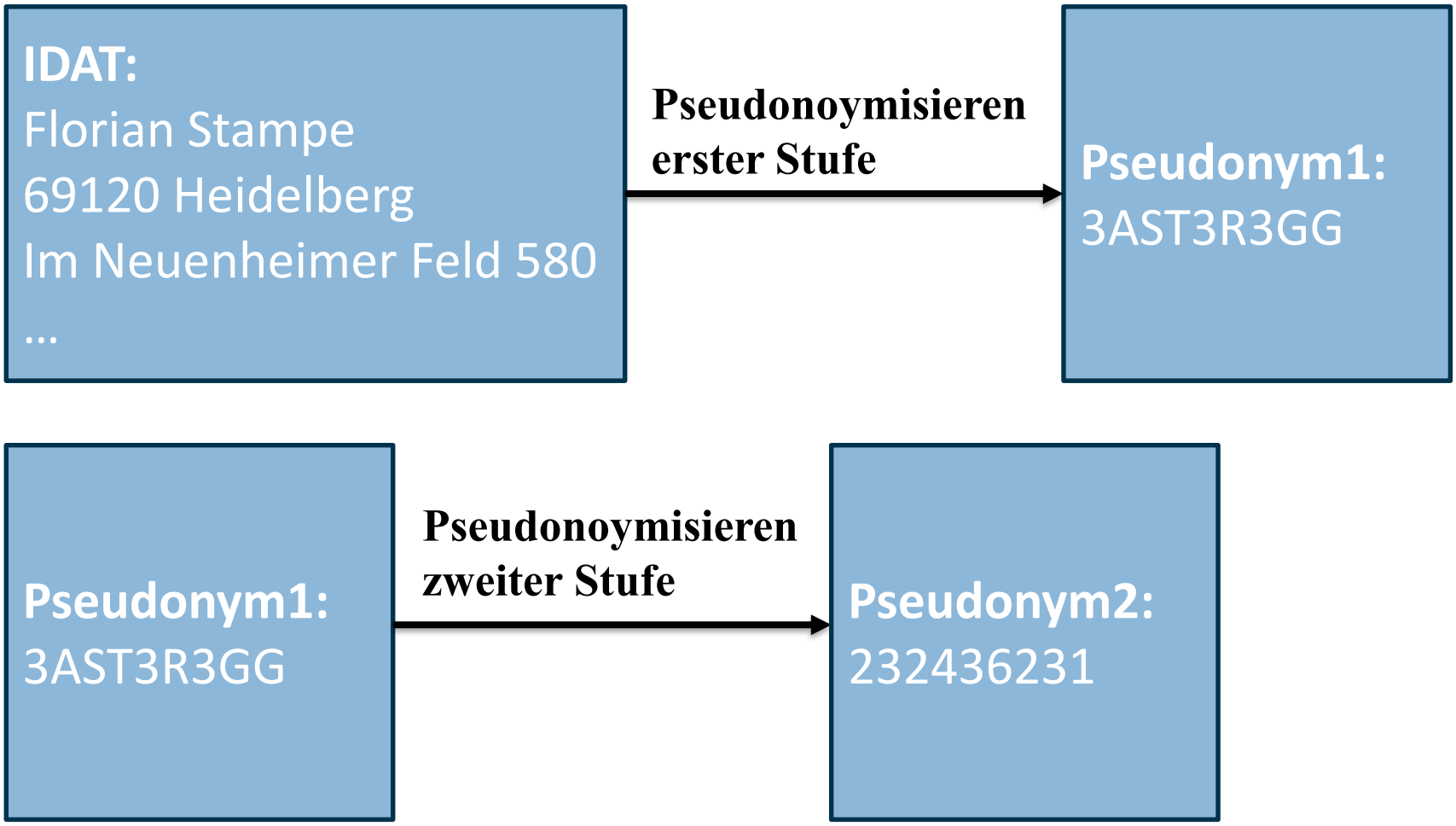
Einführung – Wodurch?

Das ID-Management wird in zwei funktionale Komponenten aufgeteilt:

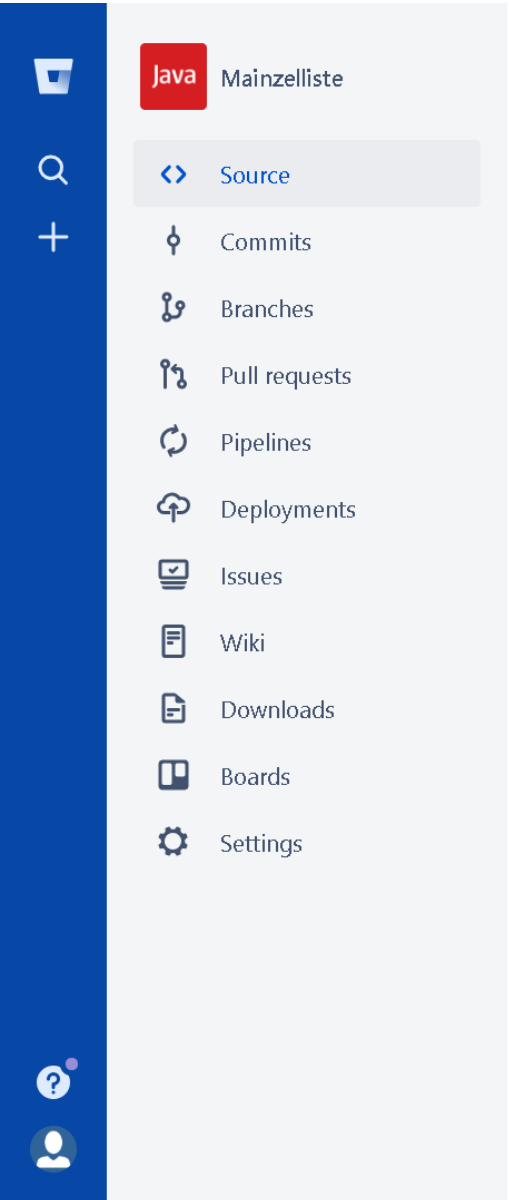
1. Patientenliste
2. Pseudonymisierungsdienst (1 und 2 Stufe)



Pseudonymisierung 1 & 2 Stufe



Einführung – Eigenschaften/Funktionalitäten



References

Mainzelliste is used in various medical joint research projects, including:

- German Mukoviszidose Register (Data protection concept)
- European chILD-EU register (Ethics/Data Safety)
- German Cancer Consortium
- Cluster for Individualized Immune Intervention (Ci3) (Meeting abstract on IT concept)
- Studies conducted by the LASER group
- The MIRACUM consortium

Another important use case is pseudonymization in central biobanks, for example:

- Comprehensive Biomaterial Bank Marburg
- Hannover Unified Biobank

The Mainzelliste API has been implemented in the following projects and software products:

- OSSE – Open Source Registry System for Rare Diseases in the EU
- OpenClinica (see presentation on integrating Mainzelliste and other software)
- secuTrial (see modules description)
- Semantic Clinical Registry System for Rare Diseases
- MOSAIC (the external interface of the "Trusted Third Party Dispatcher" is oriented towards the token-based concept of the Mainzelliste API, see Bialke et al., J Transl Med. 2015, 13:176)
- German Center for Cardiovascular Disease (DZHK) (see MOSAIC and the data protection concept)
- German National Cohort (see MOSAIC and the data protection concept)
- Electronic data capture system by Fraunhofer FOKUS
- CentraXX by Kairos GmbH
- Platform for medical research by Climedo Health GmbH (Description of electronic health record component)

We have compiled this list from the results of public search engines. If you use the Mainzelliste or its API, we would be glad to include your project in this list. Please don't hestitate to contact us.

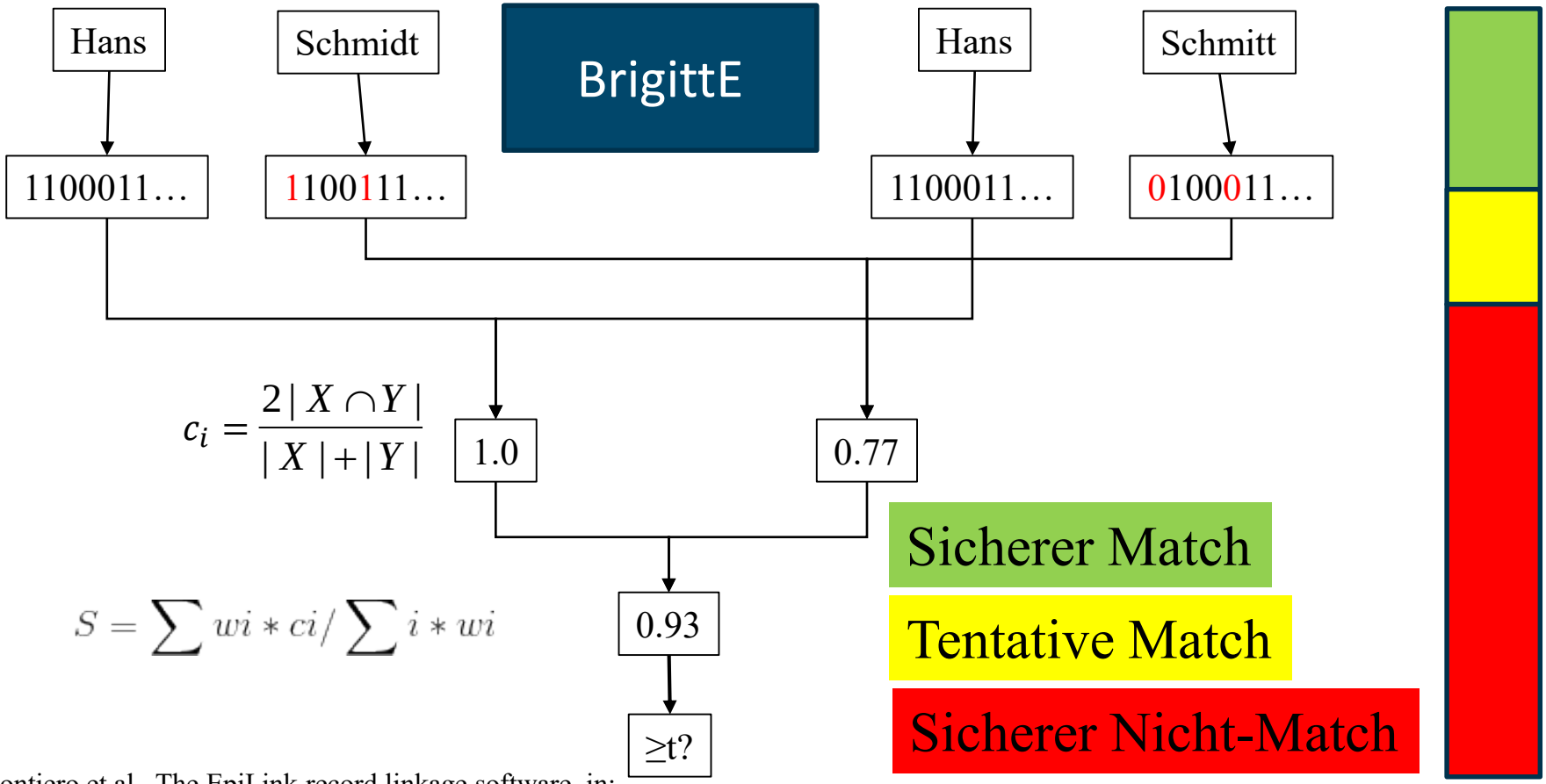
Contributing

Einführung – Eigenschaften/Funktionalitäten

Mainzelliste-Funktionalitäten

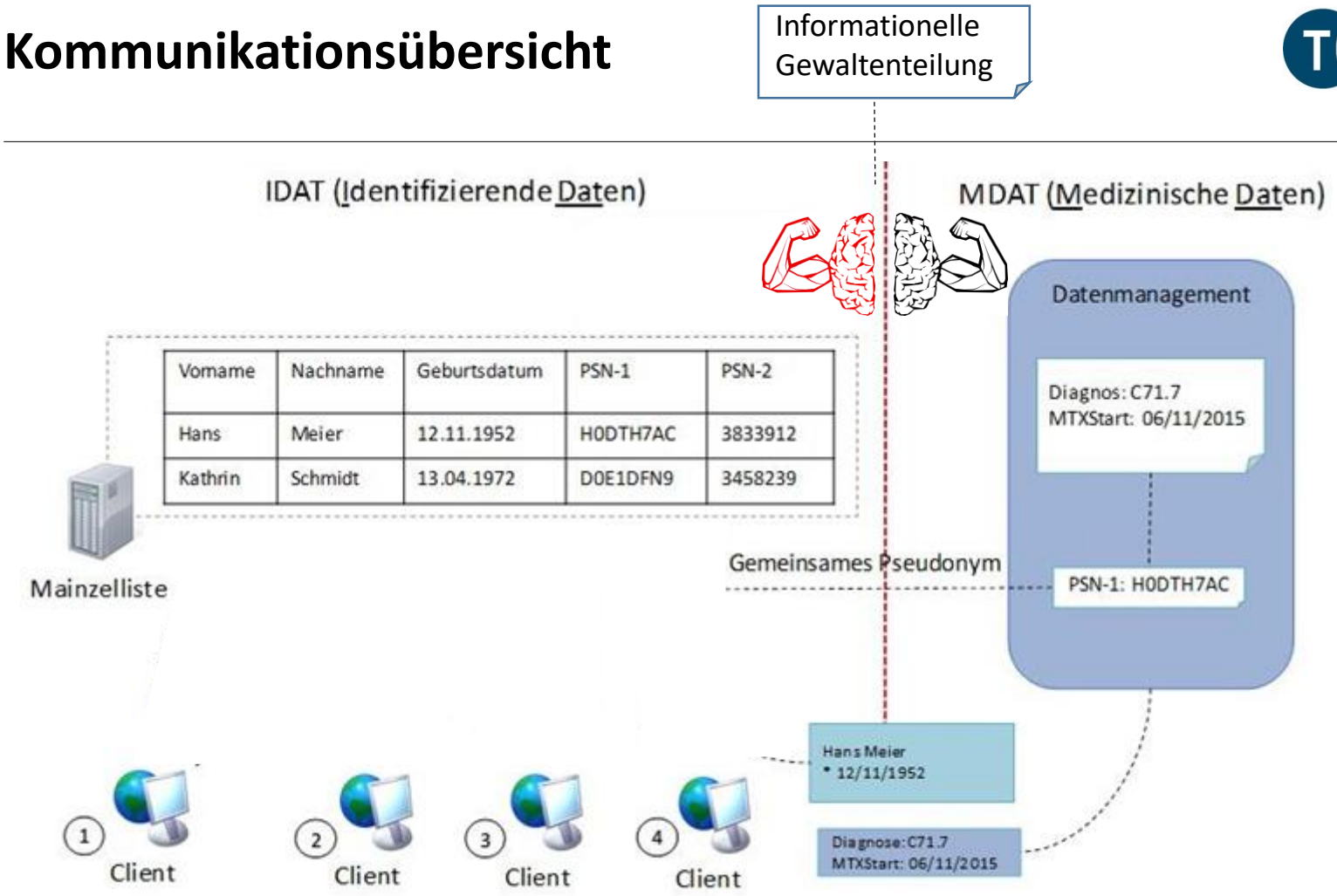
- Pseudonymisierung erster und zweiter Stufe
- PID-Generierung (nichtsprechend & fehlertolerant)
  - Alternative Generatoren (z.B. für Bioproben kürzere Pseudonym)
- REST-basierte Schnittstelle
- Webformulare zum Anlegen und Bearbeiten von Patienten
- Fehlertolerantes Record-Linkage

Record Linkage – EpiLink Algorithmus



*P. Contiero et al., The EpiLink record linkage software, in: Methods of Information in Medicine 2005, 44 (1), 66–71.

Kommunikationsübersicht



1 & 2
Pseudonymisierung
erster Stufe

3
Pseudonymisierung
zweiter Stufe

4
Depseudonymisierung

Schnittstelle der Mainzelliste

Jetzt etwas für die Techniker:

REST-Ressourcen

- **Sessions**
- **Tokens**
- **Patients**
- **HTML**

Prinzipieller Kommunikationsablauf

1. Session anlegen
   - Aufruf: POST https://mainzelliste.de/bsp/sessions ...
   - Antwort: SessionId - https://mainzelliste.de/bsp/sessions/{sessionId}
2. Token anfordern:
   - Aufruf: POST https://mainzelliste.de/bsp/sessions/{sessionId}/tokens ...
   - Antwort: TokenId - https://mainzelliste.de/bsp/sessions/{sessionId}/tokens/{tokenId}
3. Arbeiten mit Patienten in der Mainzelliste (CRUD)
   - Create
   - Read
   - Update
   - Delete

Schnittstelle der Mainzelliste

Patients – Anlagen eines Patienten

Methode: – POST

- Aufrufender: Beliebiger Tokeninhaber
- URL-Parameter: tokenID eines addPatient-Tokens
- Daten:
  - Felder (IDAT) als Schlüssel-Wert-Paare
  - Extern generierte IDs als Schlüssel-Wert-Paare
  - „sureness“ als Boolean (Gelten Daten als fehlerfrei)

Rückgabe

- 201 – Created: Der Patient wurde angelegt. Ids werden zurück gegeben.
- 400 – Eingabedaten sind ungültig
- 401 – Nicht autorisiert
- 409 – Conflict. Es wurde ein unsicherer Match gefunden

Schnittstelle der Mainzelliste

Patientenliste abrufen

Methode: – GET

- Parameter:
  - Token: Id eines Tokens vom Typ readPatients

Rückgabe

- Array von Patientenobjekten mit den angeforderten Daten

Schnittstelle der Mainzelliste

Bestehenden Patienten editieren

Methode: – PUT

- Parameter:
  - Token: Id eines Tokens vom Typ editPatient
- Daten:
  - Daten als Schlüsse-Wert-Paare
  1. Können Felder als auch externe IDs sein
  2. Felder die nicht gesetzt werden, werden nicht geändert
  3. Felder mit null oder Leerstring werden gelöscht

Rückgabe

- 200 – Es wird eine HTML-Seite mit erfolgreichen Änderungen ausgegeben
- 400 – Bad Request. Eingabedaten sind ungültig
- 401 – Unauthorized
- 404 – Not found (Angegebener Patient existiert nicht)

Schnittstellen Demonstration

Admin-GUI


localhost:8084/html/createPatient?tokenId=7ec65d11-4b8c-4ac6-9ce2-e76183090a38
PID anfordern
Hinweise zur Eingabe
Diese Anwendung gibt für die von Ihnen im Folgenden einzugebenden Stammdaten einen Personenidentifikator (PID) zurück. Dabei wird der bekannte Patientenbestand durchsucht; bei einem Treffer wird der bestehende PID zurückgegeben. Bitte beachten Sie bei Ihrer Eingabe die folgenden Punkte:
Geben Sie alle Ihnen bekannten Vornamen an, getrennt durch Leerzeichen.
Achten Sie bei Doppelnamen darauf, ob sie mit Bindestrich oder zusammen geschrieben werden (z.B. "Annalena" oder "Anna-Lena").
Geben Sie den Geburtsnamen nur an, falls er vom aktuellen Nachnamen abweicht (z.B. bei Namenswechsel durch Heirat).
Die mit * markierten Felder sind Pflichtfelder.
Stammdaten
Vorname:
(z.B. Anne-Marie Luise)
Nachname :
(z.B. Müller-Schulze)
Geburtsname :
(z.B. Schulze)
(falls abweichend)
Geburtsdatum :
Tag
Monat:
Jahr:
Wohnort :
(Postleitzahl / Ort)
#####
(z.B. Mainz)
PID anfordern

Admin-GUI

Ergebnis

Ihr(e) angeforderter/angeforderten Pseudonym(e) lauten:

- `pid: OOOCUOWP`

Bitte übernehmen Sie diese(s) in Ihre Unterlagen.

Der Text vor dem Doppelpunkt bezeichnet jeweils den Pseudonymtyp, der Text danach ist das Pseudonym selbst.

Eingegebene Daten:

Vorname:	BERND
Nachname :	BROT
Geburtsname :	
Geburtsdatum :	04. April 1955
Wohnort :	69120 Heidelberg

Installation/Betrieb Klassisch

- Z.B. Windows/Linux Betriebssystem
  - Betriebssystem muss WebServer und Datenbank ausführen
- Z.B. Tomcat 6, 7 oder 8
  - WebServer der Webanwendungen nach der Java-Servlet-Spezifkation ausführen kann. Geht auch eine Nummer größer mit JBoss oder Glassfish.
- Z.B. PostgreSQL
  - Auch andere Datenbanken wie MySQL möglich. Da JDBC verwendet wird, andere DB-Systeme möglich (untested).

Mainzelliste als Docker-Container

- Einfach möglich auf dem Stand der aktuell veröffentlichten Version zu bleiben/upzudaten
  - docker pull medicalinformatics/mainzelliste
- Kombination der verschiedenen Service, wie Web-Server und Datenbank als Docker-Compose verfügbar
- Einfache Konfiguration über Umgebungsvariablen möglich.
- Kombinierbar mit klassischer Instanz
  - Z.B. Mainzelliste in Docker und PostgreSQL auf klassischem Server
- Weitere/Komplexere Softwarekombinationen/Services einfach aufsetz- und konfigurierbar
  - Beispiel: Secure-Record-Linkage

Klassische Konfiguration

- Standard Pfad unter Unixoiden BS: /etc/mainzelliste/mainzelliste.conf
- Template mainzelliste.conf.default mitgeliefert
- Parameter unter anderem für: Datenbank, Zugriffsberechtigungen, Logging, IDAT Felder, Matcher (Record Linkage), ID Generatoren
- Gut gepflegtes Konfigurationshandbuch online verfügbar

Beispiele:

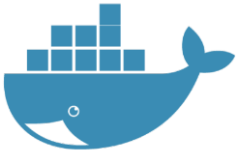
servers.0.apiKey = h0chK0mPleXeRKEY19032019
servers.0.allowedRemoteAdresses = 192.168.56.26;127.0.0.1
servers.0.permissions = createSession;createToken; tt_readPatients

log.filename = /var/log/mainzelliste/mainzelliste.log
log.level = DEBUG

Dokumentationen

- Anleitung für Entwickler
- Konfigurationshandbuch
- Installationsanleitung
- Anleitung für den MDAT-Admin
- Schnittstelle der Mainzelliste

Docker-Konfiguration



- Abstrahiert einen Teil der Konfiguration
- Konfigurationsparameter werden als Umgebungsvariablen mitgegeben
- Einfachster Weg ist über Docker-Compose Datei

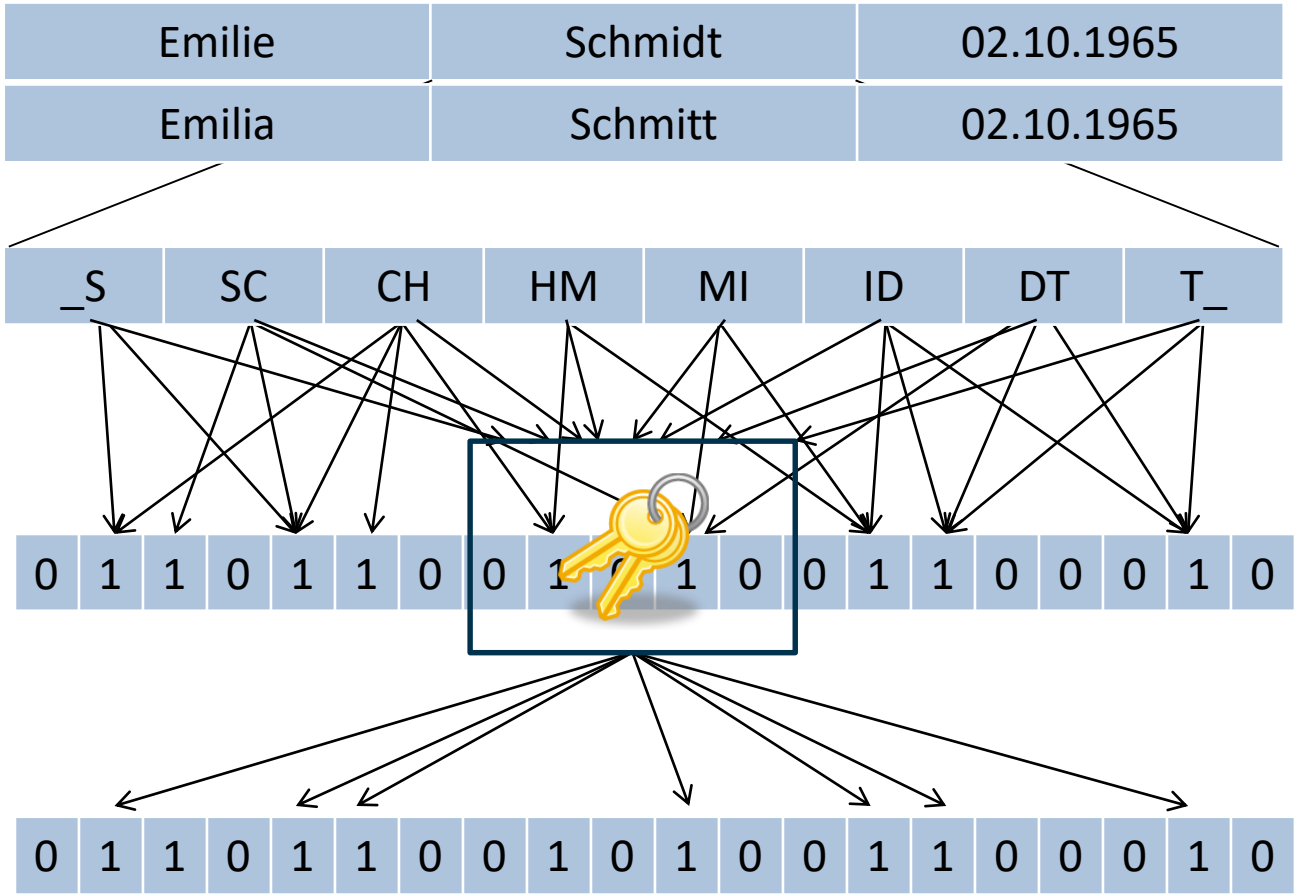
```
postgres:
    image: postgres:9.5-alpine
    ...
mainzelliste:
    container_name: mainzelliste
        - LOG_MODE=stdout #stdout=stdout everything else =logging in mainzelliste.log
        - LOG_LEVEL=DEBUG
        - ML_DB_HOST=postgres
        - SERVER_0_APIKEY=/run/secret/MainzellisteApiKey

    volumes:
        - ./mainzelliste/conf/:/etc/mainzelliste/
        - ./mainzelliste/logs:/usr/local/tomcat/logs/
    ...
secrets:
        - MainzellisteApiKey
```

Einführung – Eigenschaften/Funktionalitäten

Mainzelliste-Funktionalitäten durch Erweiterungen:

- Temporäre Identifikatoren
- Projektverwaltungsoberfläche
- JavaScript-Anbindung (Mainzelliste.Client)
- Privacy Preserving Record Linkage (Kontrollnummergeneratoren)
- Definition komplexer Pseudonymisierungsabläufe (MagicPL-Projekt)
- Ausblick: Secure Multiparty Computation

(Privacy-Preserving) Record Linkage

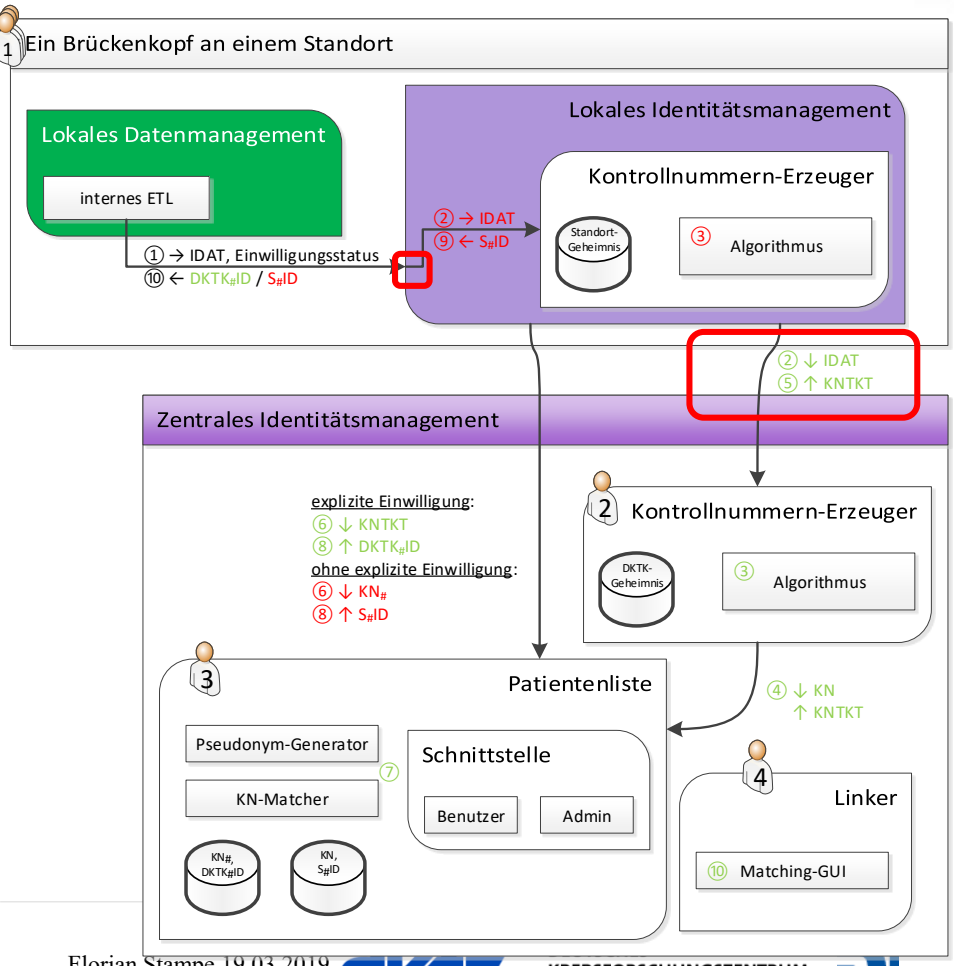


Rainer Schnell, Tobias Bachteler and Jörg Reiher: Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making* 2009, **9**:41

MAGICPL

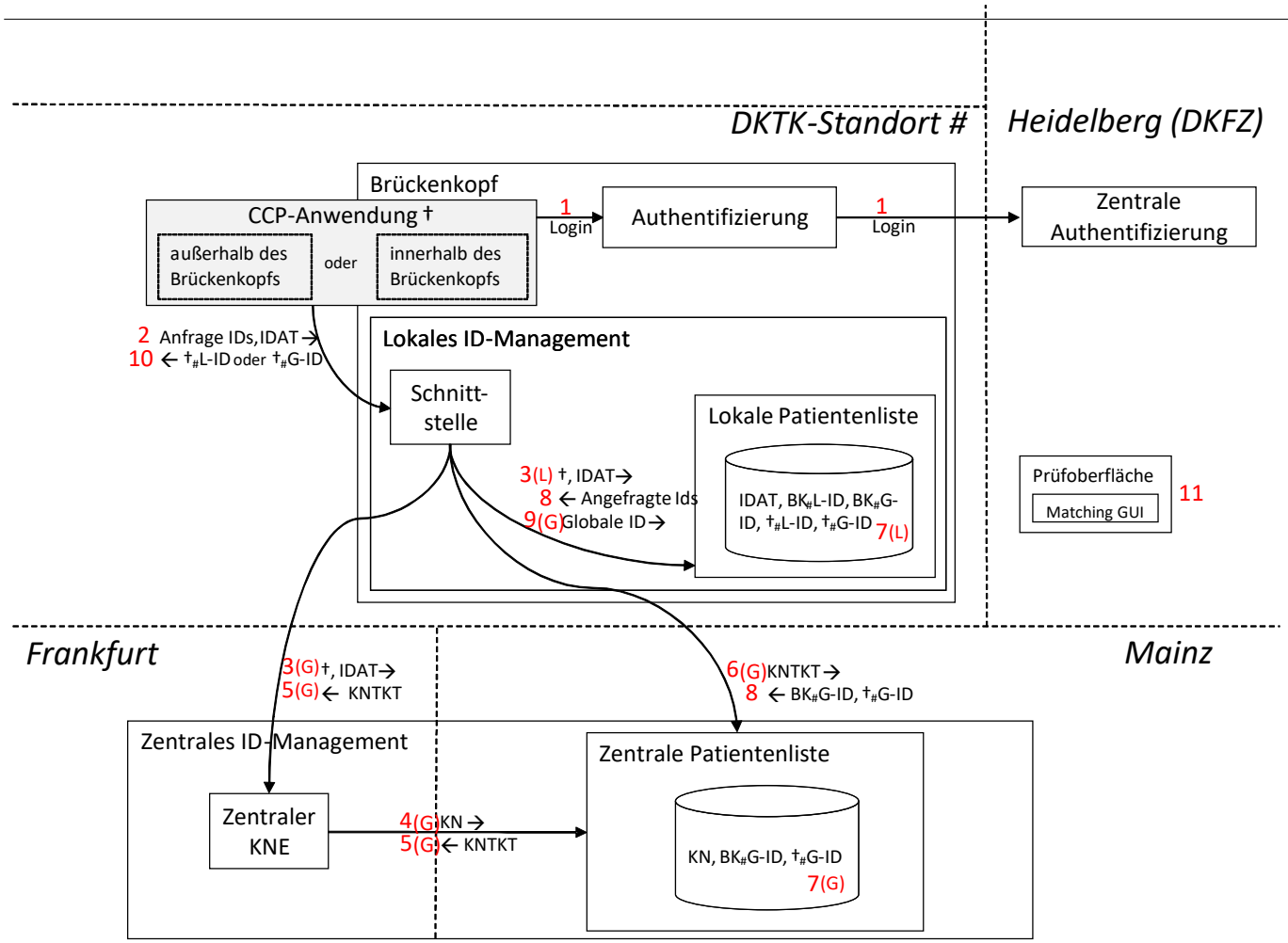
- Pseudonymisierungsprozesse sind komplex
  - Verwendung von „Multimainzellisten“
  - Es gibt keine „one size fits all“ Solution.
- MAGICPL beschreibt wie Mainzelliste miteinander kommunizieren
  - XML
  - Beschreibungssprache für Pseudonymisierungsprozesse
  - Es gibt Bausteine („Building Blocks“) die ja nach Projekt miteinander kombiniert werden können
  - Wiederverwendbar

MAGICPL




```
<?xml version="1.0" encoding="UTF-8"?>
<pathconfig xsi:schemaLocation="http://www.example.org/Pfade paths.xsd "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.example.org/Pfade">
  - <paths>
        <!-- /paths/getId * Schritte 1,10 im DSK * Zuständige Komponentente: Lokales ID-M
  - <path>
      <name>getId</name>
      <parameters/>
    - <input>
        <iorecord ref="IDAT"/>
        <iosingle ref="Einwilligungsstatus"/>
      </input>
    - <output>
        <iosingle/>
        <iorecord/>
      </output>
    - <switch evaluator="IsConsentedEvaluator">
      - <case value="true">
            <!-- * Schritte 2,5,6,8 GRÜN in DSK * Teilpfade: /paths/getKNTKT
            * /paths/getDKTKId -->
        - <multipath>
            <name>getDKTKId</name>
            <parameters/>
          - <input>
              <iorecord ref="IDAT"/>
              <iosingle ref="Einwilligungsstatus"/>
            </input>
          - <output>
              <iosingle ref="DKTK#ID"/>
            </output>
        - <step>
              <!-- /paths/getKNTKT * Schritt 2,5 GRÜN in DSK * Zuständige Ko
              Zentraler KNE -->
            <name>getKNTKT</name>
          - <input>
              <iorecord ref="IDAT"/>
            </input>
          - <output>
              <iosingle ref="KNTKT"/>
            </output>
            <implementation>de.mainzelliste.paths.processor.CngClient</im
        </step>
      - <step>
```

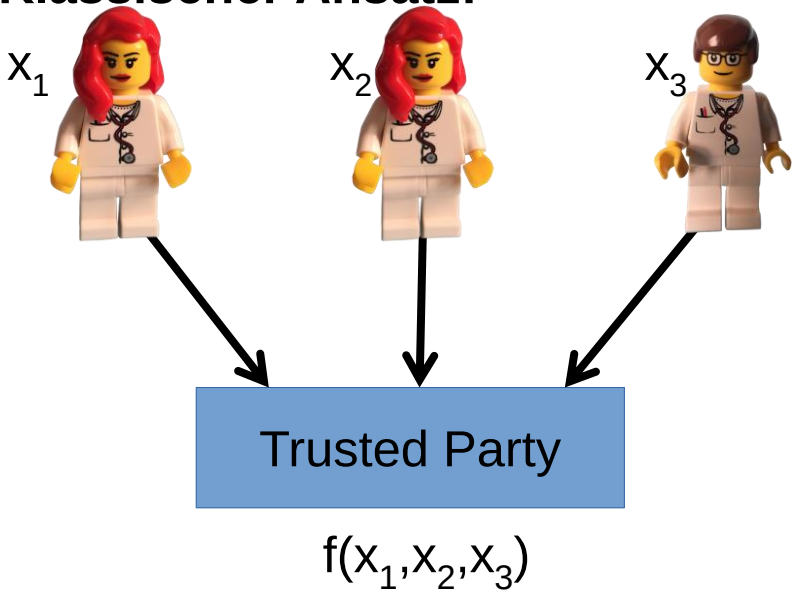
Verwendung Fallbeispiele



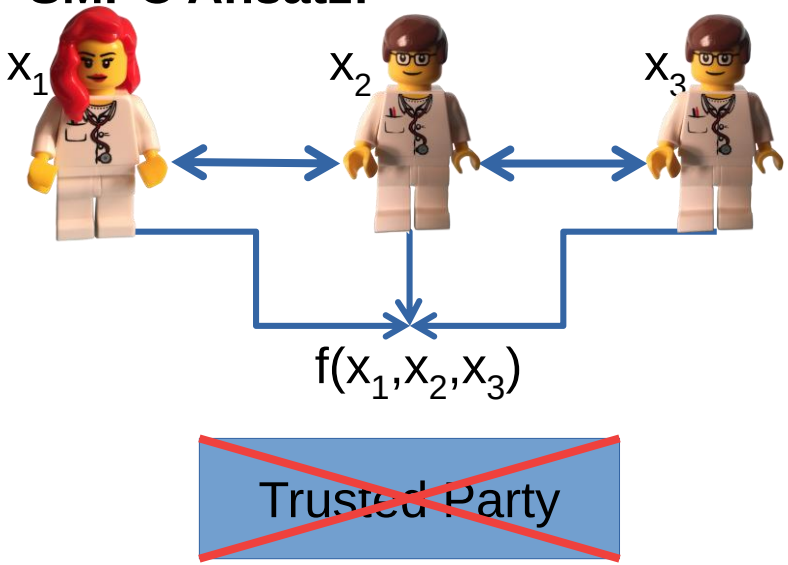
Quelle: DKTK Datenschutzkonzept

Secure Multi-party Computation (sMPC)

Klassischer Ansatz:



SMPC Ansatz:



Secure Multi-party Computation (sMPC)

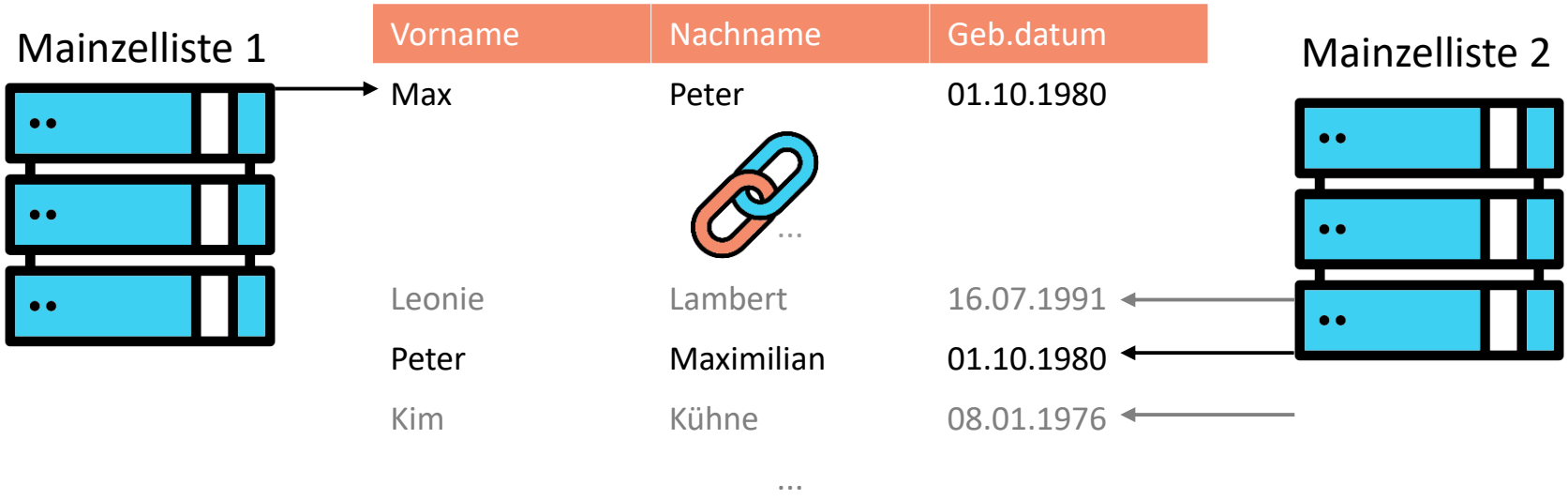
„[…] there's a very central theorem in crypto and it really is quite a surprising fact.
That says that **any computation** you'd like to do,
[...] that you can compute with a trusted authority,
you can also do **without a trusted authority**.“

- Prof. Dan Boneh, Stanford University

Informationstheoretische Sicherheit

- Auch perfekte Sicherheit genannt
- Beispiel One Time Pad (OTP)
  - Ein unendlich starker Angreifer mit unendlich viel Rechenleistung kann die Verschlüsselung nicht brechen

Ausblick: Secure Record Linkage



- Mittels **Secure Multi Party Computation** (sMPC) linken zwei Mainzellisten ihre Einträge, ohne die Daten der Gegenseite zu offenbaren.
- Fehlertoleranz wird durch Verwendung des **EpiLink-Algorithmus** garantiert
- Das sMPC Framework **ABY** (TU Darmstadt) garantiert kryptographische Sicherheit und eliminiert die Notwendigkeit einer Trusted Third Party
- Entwicklungsstand: **Prototyp / Proof-of-Concept** funktioniert

Docker Secure Record Linkage Demonstration

Kostenfreie Bereitstellung

Kostenfreie Nutzung der Mainzelliste durch AGPLv3-Lizenz

Mehr Informationen unter:

https://www.toolpool-gesundheitsforschung.de/produkte/mainzelliste

https://hub.docker.com/r/medicalinformatics/mainzelliste/ (BETA)

https://bitbucket.org/medicalinformatics/mainzelliste