Kapitel 3

Angewandte Methoden

Hier werden in aller Kürze die Methoden beschrieben, die zur Anonymisierung der Daten von Helsana Gesundheitswissenschaften letztendlich für die finale Anonymisierungsstrategie verwendet wurden. Für eine detailiertere Beschreibung und weitere Methoden verweisen wir auf Templ (2017).

3.1 Pseudoanonymisierung

Direkte Identifizierungsvariablen (wie z.B. AHV Nummer, andere Personen-ID's, etc.) werden grundsätzlich entweder vor der Weitergabe der Daten gelöscht oder pseudoanonymisiert. Wie bereits erwähnt reicht eine Pseudoanonymisierung nicht aus um Daten vor der Re-Identifizierung von Personen ausreichend zu schützen. Die Pseudoanonymisierung ist aber unerlässlich für nicht vor der Datenweitergabe gelöschte direkte Identfizierungsvariablen.

Oft werden einfach andere ID's vergeben, z.B. für die Identifizierungsnummern einen Ersatz wie 796752604, wobei der Datenempfänger keinen Bezug zu dieser Nummer hat. Besser ist eine kryptologische Hashfunktion zur Pseudoanonymisierung zu verwenden. Im Folgenden wird eine Prozedur vorgeschlagen, die solche Funktionen user- und projekt-spezifisch verwendet.

3.1.1 Hashfunktion

Eine kryptologische Hashfunktion oder kryptographische Hashfunktion ist eine spezielle Form einer Hashfunktion. Es ist praktisch nicht möglich, zwei unterschiedliche Eingabewerte zu finden, die einen identischen Hashwert ergeben. Dies ist unter dem Schlagwort Kollisionssicherheit bekannt. Ausserdem gilt bei

einer kryptologische Hashfunktion die Einwegfunktion: aus dem Hashwert darf von *Aussenstehenden* der originale Inhalt nicht erzeugt werden können.

Der Hashwert (mit vorgegebener Länge) selbst stellt das Ergebnis dar, welcher mittels einer Hashfunktion berechnet wurde. Zumeist wird der Hashwert als eine hexadezimale Zeichenkette kodiert. Ein Hashcode des Namens "Matthias Templ" könnte so aussehen:

```
library(anonymizer)
name <- "Matthias Templ"
myhash <- hash(name, .algo = "sha256", .seed = 123)
myhash
```

```
## [1] "6f25de7af3ef4795a0d19f34eee7baa422f71463ab9bd244ab69540dc9625b85"
```

Hierbei kam der SHA-2 Algorithmus zum Einsatz, genauer, der SHA-256 welcher vom US-amerikanischen National Institute of Standards and Technology (NIST) standardisiert wurde. SHA-2 wird als sicher betrachtet und zur Benutzung empfohlen (Dang, 2002). Andere Algorithmen können gewählt werden, z.B. sha1, crc32, sha512, xxhash32, xxhash64 and murmur32.

3.1.2 Salt

Salt bezeichnet in der Kryptographie eine zufällig gewählte Zeichenfolge, die an einen gegebenen Klartext **vor** der Verwendung als Eingabe einer Hashfunktion angehängt wird. Der bei Pseudoanonymisierung verwendete Salt wird zusammen mit dem entstandenen Hashwert in einer Datenbank gespeichert.

Die Verwendung eines Salts erhöht den Aufwand dieser Angriffe deutlich, speziell Wörterbuchangriffe oder allgemein Angriffe mit bestehenden Registern an Namen, AHV-Nummern, etc. werden erschwert, da nicht mehr in einer Liste von Hashwerten (Hashtabelle) der zugehörige Klartext (Passwort) nachgeschlagen werden kann, sondern für jeden Klartext hash (Klartext + Salt) = bekannter Hash überprüft werden muss.

```
name_salt <- salt("Matthias Templ", .n_char = 5, .seed = 123)
name_salt
```

```
## [1] "hukwyMatthias Templhukwy"
```

```
myhash_salt <- hash(name_salt, .seed = 123)
myhash_salt
```

```
## [1] "08c4d795f622613c323e2e0533b955591d20efcf7ba62d3542775c64c89db6d5"
```

Tabelle 3.1: Datawarehouse mit unterschiedlichen Hashes für unterschiedliche Datenlieferungen. Es handelt sich bei den Kosten um erfundene Werte. Die Hashes sind aus Demonstrationszwecken nur 8 Zeichen lang.

names	hash_names1	hash_names2	kosten	geschlecht
Beat Bruengger	88e85240	cba7a110	36231	maennlich
Matthias Templ	f0dceabc	6ec25ff9	37845	NA
Eva Blozik	994a5642	2f8697ec	NA	weiblich

3.1.3 Unsalt

Gegeben den verwendeten Startwert des Zufallszahlengenerators, kann jederzeit bei Bekanntheit des Seeds ein Unsalt gemacht werden.

```
unsalt(salt(name_salt, .seed = 123), .seed = 123)
```

```
## [1] "Matthias Templ"
```

3.1.4 Zentraler Schlüssel mit Hashes

Die Idee ist empfängerspezifische oder/und projektspezifische Hashs zu erzeugen.

Hier ist ein sehr einfaches Beispiel gegeben, welches ohne Aufwand auf grosse Datensätze angewandt werden kann. Anhand von Tabelle 3.1 sieht man, dass zwei unterschiedliche Hash's erzeugt wurden. Dies geht einfach von statten, falls der Zufallszahlengenerator entsprechend verschieden initialisiert wurde.

Szenarios wären:

- Datenempfänger 1 braucht spezifische Variablen des Datawarehouse für Projekt 1. Er benötigt auch Daten für Projekt 2. Man möchte sicher gehen, dass Datenemfänger 1 die Daten nicht direkt zusammenführen kann.
- Datenempfänger 1 braucht spezifische Variablen des Datawarehouse für ein Projekt. Datenempfänger 2 braucht spezifische Variablen des Datensatzes für ein Projekt. Man möchte sicher gehen, dass sich Datenemfänger 1 und 2 die Daten nicht direkt zusammenführen können.

Wir wollen Szenario 1 betrachten (Szenario 2 ist ident bzgl folgendem Vorgehen).

Datenempfänger 1 bekommt folgende Datenlieferung für Projekt 1 (siehe Tabelle 3.2), ein Subset bestehend aus den Zeilen 1 und 2 des grossen Datensatzes im Datawarehouse. Natürlich geben wir ihm nicht die Namen, sondern die pseudoanonymisierten Namen.

Würde Datenempfänger 1 nun identische Hashs für jeweilige gleiche Personen bekommen, könnte er die Daten sofort eindeutig zusammenführen. Wir wollen

Tabelle 3.2: Datensatz 1 für Empfänger 1

names	kosten
88e85240	37845
f0dceabc	36231

Tabelle 3.3: Datensatz 2 für Empfänger 1

names	geschlecht
6ec25ff9	weiblich
2f8697ec	maennlich

dies vermeiden, indem er andere Hashs für selbige Personen bekommt. Für sein Projekt 2 bekommt der Datenempfänger 1 nun folgende Daten (siehe Tabelle 3.3), ein Subset des grossen Datensatzes bestehend aus Zeilen 2 und 3 und eines anderen hashs als bei Projekt 1.

Der Empfänger kann nun nicht eindeutig mergen, obwohl er beide Datensätze zur Verfügung hat die sich in ihren Merkmalen und Populationen überschneiden.

Die Zentrale kann jedoch auch später Daten zusammenführen. Die Zentrale kann bei gegebenen Anlass für Empfaenger 1 mergen.

3.1.5 Personen-, daten- und institutionsspezifische Kennzahl und individuelle Schlüssel mit Salt und Hash

Hier kann die gleiche Initialisierung des Hash verwendet werden. Es ergeben sich andere Hash's nur aufgrund anderen Salts pro Projekt oder Datenempfänger, siehe Tabelle 3.3.

Vorgehensweise bei zwei Datenlieferungen und einer Identifizierungsvariablen ID1 (die Verallgemeinerung auf mehrere Datenlieferungen und mehreren ID's ist trivial):

1. Salt der ID1 mit Initialisierung 1 (Zufallsgenerator) für Datenlieferung 1 → salt1ID1
2. Salt mit ID1 Initialisierung 2 (Zufallsgenerator) für Datenlieferung 2 → salt2ID1
3. Hash der gesalteten Information salt1ID1 → hashsalt1ID1. Weitergabe dieses Hashes (hashsalt1ID1) statt der ID1 bei Datenlieferung 1.
4. Hash der gesalteten Information salt2ID1 → hashsalt2ID1. Weitergabe dieses Hashes (hashsalt2ID1) statt der ID1 bei Datenlieferung 2.

Tabelle 3.4: Datawarehouse mit hashes aus gesaltetem Namen. (continued below)

names	salt_names1	salt_names2
Beat Bruengger	hukwyBeat Bruenggerhukwy	cqpqwBeat Bruenggercqpqw
Matthias Templ	hukwyMatthias Templhukwy	cqpqwMatthias Templcqpqw
Eva Blozik	hukwyEva Blozikhukwy	cqpqwEva Blozikcqpqw

hash_salt1	hash_salt2	kosten	geschlecht
e12307ae	ecc61634	36231	maennlich
4de7e72d	6ecf2190	37845	NA
441565f6	fe5ec3b1	NA	weiblich

Der Empfänger bekommt bei Datenlieferung 1 in unserem Szenario die pseudoanonymisierten Namen aus hash_salt1, bei Datenlieferung 2 die bezüglich hash_salt2 (siehe wiederum Tabelle 3.3.

Der Zentrale ist es weiterhin möglich eindeutig Daten zusammenzuführen, jedoch den Datenempfängern wird die eindeutige Datenverknüpfung abermals verunmöglicht. Für jedes Projekt ein User einen anderen Hash für dieselbe ID. Unterschiedliche User bekommen ebenfalls unterschiedliche Hashs für die gleiche ID.

3.2 Reidentifizierungsrisiko

Die Abwägung zwischen Re-identifizierungsrisiko und Datennutzen ist zentral. Je niedriger das Re-identifizierungsrisiko ist, desto mehr leidet die Qualität der Daten und desto grösser ist der Informationsverlust in den Daten. Die größte Herausforderung ist die feine Linie zu finden, bei der beide Masse auf einem akzeptablen Niveau sind.

Grundsätzlich sollte nur eine anonymisierte Version veröffentlicht werden (siehe auch Benschop et al., 2016). Die Veröffentlichung mehrerer anonymisierter Versionen derselben Daten könnte zu einer unbeabsichtigten Identifizierung von Personen führen. Wir nehmen an, dass vor dem Anonymisierungsprozess die Originaldaten einen Datennutzen von 100% aufweisen und daraus folgt, dass der Informationsverlust 0 ist.

Eine der grössten Herausforderungen der statistischen Geheimhaltung ist das Risiko und den Nutzen genau einzuschätzen (Hundepool et al., 2012b) und zwischen legitimen externen Datenempfängern (z.B. vertrauenswürdige Forscher),